International Journal of Innovative Technology and Interdisciplinary Sciences
www.IJITIS.org
ISSN:2613-7305
Volume 4, Issue 1, pp. 623-633, 2021
DOI: https://doi.org/10.15157/IJITIS.2021.4.1.623-633
Received January 24, 2021; Accepted March 10, 2021

A Unique Approach to Solve the System of Linear Equations

Md. Nur-E-Arefin
Department of Computer Science & Engineering, Royal University of Dhaka, Bangladesh
nur.arefin@royal.edu.bd

ABSTRACT

System of linear equations is a set of linear equations with same types of variables. Aside from mathematics, systems of linear equations are used in information theory, communication theory, and related fields. This study is aimed at analyzing the available methods and develops a new solution which does not involve direct matrix inversion for system of linear equations. By comparing the different test results with an existing and very well-known method called gauss elimination method, it has been seen that in terms of numerical accuracy and computing time the proposed approach achieves improve results. Furthermore, even very large systems can be solved by this proposed algorithm given a cluster with sufficient resources.

Keywords: System of linear equations; self-verifying method; indirect method; large linear interval systems; new algorithm.

1. INTRODUCTION

A system of linear equations means a set of linear equations. If these two or more linear equations intersect, that point of intersection is the solution to that two or more linear equations. Systems of linear equations can be solved in polynomial time using different methods [1]. In mathematics, system of linear equation is the basis of linear algebra [2]. There are lots existing algorithms for the system of linear equations, but the main target of this study is how to reduce the computational time as well as complexity. As solving a linear system is the main focus, there are many methods already discovered by the mathematicians. These techniques are Elimination of variables, Row reduction, Gaussian elimination, Cramer's rule, Matrix solution and other methods.

A new algorithm is introduced to solve the system of linear equations where at first generate solution X^* having randomly n components and Generate $n \times n$ matrix A. Then compute constant vector b_i. Generate initial point P_0 randomly from where computation starts and construct $P_i \perp \mathrm{H_i}$ from P_{i-1}. Generate new initial point P'_i and calculate distance between newly generated point $P_i(t)$ and initial solution X^* and try to minimize this distance. The simplest idea of the algorithm is shown by the following Figure 1.

Suppose there are two linear equations intersect each other in fig-1. So, the intersect point X is the solution. Now take a randomly selected initial point P_0 and make a perpendicular line from P_0 to another hyper-plane which is P_0P_1. Now calculate the distance between P_1 to solution X. Then again make a perpendicular line from P_1 to P_2 and the same way calculate the distance between P_2 to X. By this way the distance from the solution X to newly generated points (P_1, P_2 , P_3 etc.) will be minimized and every newly generated point will get close to the solution.

Solving the system of linear equation by implementing this algorithm and from the algorithm programming code is implemented which runs through computer and gives us output such that complexity and computation time can be measured. Actually, to find a solution is not the concern of this paper. Minimize the time and complexity is the aim, as a result a correct solution can be found in less time.

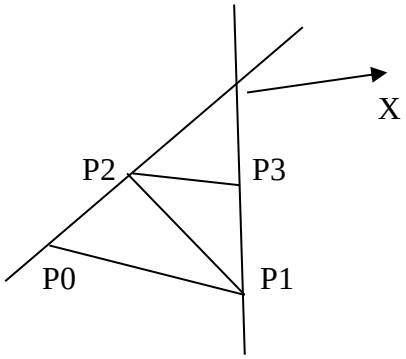

Figure 1. General idea of the proposed algorithm.

2. LITHERATURE REVIEW

Linear equations are in the form of $a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$, where $a_1 \ldots a_n$ are coefficients, $x_1 \ldots x_n$ are variables, and the constant term is b. A linear equation can have infinite number of solution (if it has 2 variables). But two system of linear equation can have one solution. The following equations are linear:

$3x + 4y = 6$ (1)

$2x_1 - x_2 + 5x_3 = 1$ (2)

$\prod (x + 2y) = z$ (3)

The following equations are not considered as linear:

$3x^2 = y$ (4)

$4xy=1$ (5)

$y=2/x$ (6)

A solution of a linear equation is the set of values $[s_1, \ldots, s_n]$, where when we can substitute $x_1 = s_1, \ldots, x_n = s_n$. One variable equation (example, $3x = 12$) can have only one solution ($x = 4$) and two variables linear equation form a line in the plane. For linear equations of more than three variables can have an infinite set of solutions [3]. Group of more than one linear equation is called a system of linear equations. Solving the system means finding a solution common to all the equations in the system. If the system has one or infinite solution then it's called consistent system otherwise it's called inconsistent system. In Homogeneous system the constant term in each equation is zero. The examples of homogeneous system of equations

$$3x + 2y + z = 0 \quad (7)$$

$$4x - y + z = 0 \quad (8)$$

$$-2x + 3y + 2z = 0 \quad (9)$$

In general, the equation $AX = B$ representing a system of equations is called homogeneous if B is zero. Otherwise, the equation is called non homogeneous. The examples of non-homogeneous system of equations:

$$2x + 3y = -8 \tag{10}$$

$$-x + 5y = 1 \tag{11}$$

2.1 Direct methods for solving system of linear equations:

Gaussian elimination consists of three-step to solve a system of linear equations [4]:

- Write down the system as an augmented matrix form.
- In order to reduce the matrix to row echelon form use row operations.
- Lastly use back substitution

Suppose there are some linear equations given below,

$$2x - 3y + z = -5 \tag{12}$$

$$3x + 2y - z = 7 \tag{13}$$

$$x + 4y - 5z = 3 \tag{14}$$

So, the augmented matrix of the above equations is:

$$\left|\begin{array}{ccc|c} 2 & -3 & 1 & -5 \\ 3 & 2 & -1 & 7 \\ 1 & 4 & -5 & 3 \end{array}\right| \tag{15}$$

The following matrix is the example of row echelon form:

$$\left|\begin{array}{ccc|c} 2 & 1 & 3 & 11 \\ 0 & 1 & 3 & 8 \\ 0 & 0 & 4 & 8 \end{array}\right| \tag{16}$$

Time complexity of Gaussian elimination is $O(n^3)$.

Gauss-Jordan elimination is similar to Gaussian elimination but there is one condition more in this method. In Gauss-Jordan method we have to find the row-reduced echelon form (RREF) augmented matrix. Jordan presents a numerical example, derived from a least squares application in geodesy, to illustrate the method that has come to be known as Gauss-Jordan reduction [5]. A matrix is said to be in RREF if:

- In each row the leading entry is one.
- There are no entries in the column above or below without any leading entry
- If a row contains a leading one then each row above contains a leading one further to the left.

The input of the Gauss-Jordan elimination algorithm is an $m \times n$ matrix (typically an augmented matrix). But the algorithm can also works for any matrix consisting of numerical entries.

Start with $i = 1, j = 1$.

a) If $a_{ij} = 0$ exchange the i^{th} row with another row below to guarantee that $a_{ij} \neq 0$. The non-zero entry in the (i, j) position is called a pivot. If all other entries in the column are 0 then increase j by 1.
b) To make the pivot entry 1 Divide the i^{th} row by a_{ij}.
c) In the j^{th} column eliminate all other entries.

d) To choose the new pivot element, Increase i by 1 and j by 1. Go to Step 1.

The algorithm stops after processing the last row or the last column of the matrix. The output of the Gauss-Jordan algorithm is the matrix in reduced row-echelon form. Gauss Jordan method has the time complexity of $O(n^3)$.

LU-decomposition method is based on Doolittle's method while the numerical structure approach is based on Cramer's rule [6]. The LU decomposition of a square real matrix A consists of computing two matrices L and U being respectively lower and upper triangular and satisfying $A = LU$ [7]. If A is a $n \times n$ matrix, L and U are also $n \times n$ matrices. The forms of L and U are given below:

$$L = \begin{bmatrix} l_{1,1} & 0 & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & 0 & & 0 \\ \vdots & & & \ddots & \vdots \\ l_{n,1} & l_{n,2} & l_{n,3} & \cdots & l_{n,n} \end{bmatrix} \tag{17}$$

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,n} \\ 0 & u_{2,2} & u_{2,3} & & u_{2,n} \\ 0 & 0 & u_{3,3} & & u_{3,n} \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{n,n} \end{bmatrix} \tag{18}$$

Lower triangular one has all zeroes above its diagonal and upper triangular has all zeroes below its diagonal.

Complexity analysis of LU Decomposition is given below:

To solve $Ax = b$,
Decompose A into LU – cost $\frac{2}{3}n^3$ flops
Solve $Ly = b$ for y by forward substitution – cost n^2 flops
Solve $Ux = y$ for x by backward substitution – cost n^2 flops

2.2 Iterative methods for solving system of linear equations:

Iterative methods of solving equations are more advantageous if a system of equations is large. For a large set of equations, elimination methods are prone to large round-off errors. Iterative methods give less round-off error. Iterative methods can be made more judiciously leading to faster convergence [8].

Gauss-Seidel method is an iterative method used to solve linear systems of equations. Given a system of linear equation $AX = b$, where A is a square matrix, X is vector of unknowns and b is vector of right-hand side values [9]. Suppose we have a set of n equations and n unknowns in the following form:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \tag{19}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \tag{20}$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \tag{21}$$

Each equation has to rewrite in generalized form for solving corresponding unknowns. After that initial guess have to be chosen to start Gauss Seidel method then substitute the solution in the above equation and use the most recent value. Iteration is

continued until the relative approximate error is less than pre-specified tolerance. Convergence is merely guaranteed just in case of Gauss Seidel method if matrix A is diagonally dominant. Matrix is said to be diagonally dominant if the absolute value of diagonal element in each row has been greater than or equal to summation of absolute values of rest of elements of that particular row.

The iterative process is ended when an intersection criterion is fulfilled. The computation can end when the difference between two successive iterations is less than the pre-specified tolerance. Advantageously, Gauss Seidel method is a very effective method in terms of storage and time. It is automatically adjusting to if error is made. It possesses less memory when programmed. If the coefficient matrix is sparse It is very fast and simple. It starts with an approximate answer. With each iteration, accuracy is improved. It has problem that it may not converge sometime even done correctly. Another problem of Gauss Seidel method is that it is not suitable for non-square matrices. Non-square matrices are converted into square matrices by taking pseudo inverse of the matrix. Non zero elements on the diagonals are necessary for the Gauss Seidel method.

The Jacobi method is the simplest iterative method for solving $Ax = b$. This method is both good and bad. Good, because it is relatively easy to understand. And bad because it is not typically used in practice. Still, it is a good starting point for learning about more useful, but more complicated, iterative methods [10].

The Jacobi method is easily derived by examining each of the n equations in the linear system of equations $Ax = b$ in isolation. If, in the i^{th} equation:

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \tag{22}$$

Solve for the value of x_i while assuming the other entries of x remain fixed. This gives

$$x_i^{(k)} = \frac{b_i - \sum_{j \neq i} a_{ij}x_j^{(k-1)}}{a_{ii}} \tag{23}$$

which is the Jacobi method [11].

In this method, the order in which the equations are examined is irrelevant, since the Jacobi method treats them independently. The definition of the Jacobi method can be expressed with matrices as

$$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b \tag{24}$$

where the matrices D, L, and U represent the diagonal, strictly lower triangular, and strictly upper triangular parts of A, respectively.

Conjugate Gradient Method is a very effective method. The un-preconditioned conjugate gradient method constructs the k^{th} iterate x^k as an element of $x^0 +$ span $\{r^0, \dots, A^{k-1}r^0\}$ such that $(x^k - \overline{x})Ax^k - \overline{x}$ is minimized, where $\overline{x}$ is the exact solution of $Ax = b$. This minimum is guaranteed to exist in general only if A is symmetric positive definite. The conjugate gradient iterates converge to the solution of $Ax = b$ in no more than n steps, where n is the size of the matrix. In every iteration of the method, two inner products are executed in order to calculate update scalars that are defined to make the sequences convince certain orthogonal conditions [12]. On a symmetric

positive definite linear system these criteria imply that the distance to the true solution is lessen in some norm.

Successive Over-relaxation method (SOR) is a method of solving a linear system of equations $Ax = b$ derived by deducting the Gauss-Seidel method [13]. This deduction takes the form of a weighted average between the previous iterate and the computed Gauss-Seidel iterate successively for each component,

$$x^{(k)} = \overline{\omega} x_k^i + (1 - \overline{\omega}) x_i^{(k-1)} \tag{25}$$

where x denotes a Gauss-Seidel iterate and $\overline{\omega}$ is the extrapolation factor. The idea is to choose a value for $\overline{\omega}$ that will accelerate the rate of convergence of the iterates to the solution. In matrix terms, the SOR algorithm can be written as

$$x^{(k)} = (D - \overline{\omega}\mathrm{L})^{-1}[\,\overline{\omega}\mathrm{U} + (1 - \overline{\omega})\mathrm{D}]\mathrm{x}^{(\mathrm{k}-1)} + \overline{\omega}(\mathrm{D} - \overline{\omega}\mathrm{L})^{-1}\mathrm{b} \tag{26}$$

where the matrices D, L, and U represent the diagonal, strictly lower-triangular, and strictly upper-triangular parts of A, respectively [14].

3. IMPLEMENTED METHOD

Verified solvers for dense linear (interval) systems require a lot of resources, both in terms of computing power and memory usage [15]. In this section a new algorithm to solve the system of linear equations which minimizes the use of resources both in terms of computing power and memory usage is introduced. In this algorithm we assume that system must have solution.

3.1 Proposed Algorithm

The pseudo code of the algorithm is given below,

Step 1:

Generate random solution X^*

Step 2:

Generate $n \times n$ matrix A, where $a'_{ij} = a_{ij} \div \sum a_{ij}^2$

Step 3:

Compute $b_i = \sum_{j=1}^{n} a_{ij} X^*$

Step 4:

Generate initial point P_0

Step 5:

Construct $P_i \perp \mathrm{H_i}$ from $P_{i-1}$where $P_i = P_{i-1} + a_i\alpha$ and $\alpha = b_i - a_i P_{i-1}$

Step 6:

Generate new initial point P'_i Where $P_i(t) = P_i + (P'_i - P_i)$ and

$t = \sum(a_i P_i - b_i) a_i (P'_i - P_i) \div \sum(P'_i - P_i)$

Step 7:

Minimize $\sum D_i^2(t)$. Find least value to t

Where $\sum D_i^2(t) = |a_i P_i(t) - b_i|$

Step 8:

Start from step 5 again until improvement is insufficient.

3.2 Functional definition of the proposed algorithm

Step 1:

Generate solution X^* having randomly n components. For example, if n=3 then $X^* = [X_1, X_2, X_3]$ where the value of X_1, X_2, X_3 are randomly generated.

Step 2:

Generate $n \times n$ matrix A, where coefficients a_{ij} of A are randomly generated. Then normalize matrix A using following rule $a'_{ij} = a_{ij} \div \sum a_{ij}^2$

Step 3:

Compute constant vector b_i using this rule, $b_i = \sum_{j=1}^{n} a_{ij} X^*$

Step 4:

Generate initial point P_0 randomly from where computation starts.

Step 5:

Construct $P_i \perp \mathrm{H_i}$ from P_{i-1} where $P_i = P_{i-1} + a_i \alpha$ and $\alpha = b_i - a_i P_{i-1}$. Draw perpendicular line from point P_{i-1} to point P_i of the hyper-plane $\mathrm{H_i}$ where α is a direction vector. This task can be accomplished by using the following pseudo code,

For $i = 1$ to n do

$\alpha = b_i - a_i P_{i-1}$

$P_i = P_{i-1} + a_i \alpha$

End-do

$\alpha = b_n - a_n P_{n-1}$

$P_1 = P_{n-1} + a_n \alpha$

Step 6:

Generate new initial point P'_i

Where $P_i(t) = P_i + (P'_i - P_i)t$ and

$t = \sum (a_i P_i - b_i) a_i (P'_i - P_i) \div \sum a_i (P'_i - P_i)$

Each time a new point is generated from the two points P'_i, P_i on the same hyperplane with direction vector t which can be calculated by using function $f(t)$.

Here $f(t)$ can be defined by,

$f(t) = \sum_{n}^{i=1} |a_i P_i(t) - b_i|^2$

Now we take $f'(t) = 0$ and get the value of t which is given below:

$t = \sum (a_i P_i - b_i) a_i (P'_i - P_i) \div \sum a_i (P'_i - P_i)$

Step 7:

Calculate distance between newly generated point $P_i(t)$ and initial solution X^* where distance defined by:

$\sum D_i^2(t) = |a_i P_i(t) - b_i|$ and try to minimize this distance.

Step 8:

Repeat algorithm from step 5 until improvement is insufficient.

3.3 Complexity Analysis of the proposed algorithm

1. For normalize matrix the required complexity order is $O(9n^2 + 9n + 2)$.
2. To construct perpendicular line to each hyper-plane and finding two points on this hyper-plane the required complexity order is $O(18n^2 + 21n + 6)$.
3. To determine new initial point the required complexity order is $O(9n^2 + 21n + 7)$.
4. To minimize distance between solution vector and new point on the hyper-plane the required complexity order is $O(6n + 7)$.
5. The total time complexity of the proposed algorithm is $O(36n^2 + 57n + 22)$ which can be approximately defined by $O(n^2)$.

4. SIMULATION RESULT AND DISCUSSION

4.1 Experiment Setup

To implement the proposed algorithm as well as gauss elimination method, the following files are used:

- .txt file
- .cpp file

Proposed algorithm technique is implemented by the help of Microsoft Visual Studio 2006 and the algorithm is implemented in the system which has the following properties:

Processor: Intel(R) Core (TM) i5 CPU
Memory (RAM): 4.00 GB
System Type: 32-bit Operating System

4.2 Result Analysis and Discussions

One of the existing algorithms to solve linear equations which is Gauss Elimination Method is a direct method whereas proposed algorithm is an indirect method. Now the objective is to compare the proposed algorithm with respect to Gauss Elimination method. For that purpose, both gauss elimination method and proposed algorithm are implemented and then run the codes with different dimension sizes. Let us assume that

N is the dimension size of the system of linear equation. Then different sizes of dimension of the system of linear equation are applied on both of the algorithm and compare their code execution time in seconds.

Form the Table 1 it can be seen that execution time of the proposed algorithm is little bit less than the execution time of gauss elimination algorithm. Though proposed Algorithm's execution time is very little bit less compare to the Gauss Elimination Method's execution time yet the effectiveness of solving system of linear equation is improved.

Table1. Comparisons between two methods

Dimension size, N	Gauss Elimination Method (Execution Time)	Proposed Algorithm (Execution Time)
N=15	0.001	0.001
N=20	0.001	0.001
N=25	0.002	0.002
N=30	0.004	0.003
N=50	0.013	0.011
N=80	0.027	0.023
N=100	0.031	0.030
N=500	0.154	0.143
N=800	0.181	0.177
N=900	0.198	0.193
N=1000	0.201	0.197

The proposed method is successful in achieving the aim of minimize the computing power and memory usage. There is a graph in Figure 2 which is made from Table 1 is given below:

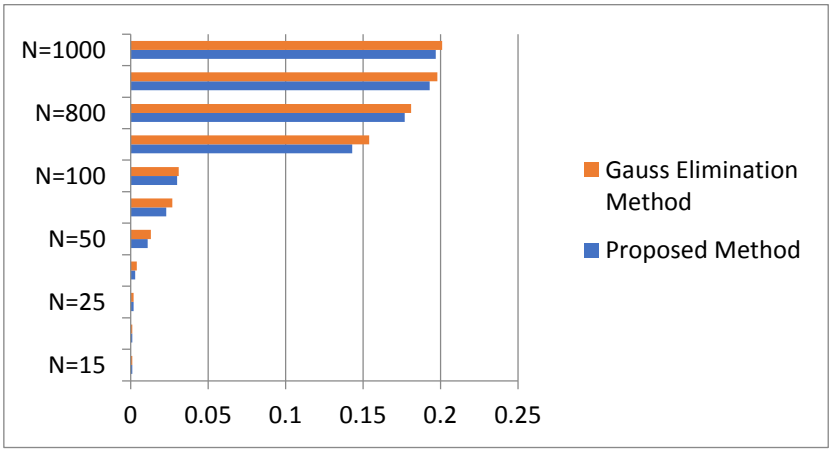

Figure 2: Comparison graph between 2 methods

Experimental results show that the proposed algorithm provide a cost-efficient solution of a system of linear equation. Because of the efficient solution overall time complexity of the algorithm decreases.

4.3 Analytical Comparison with Existing Algorithm

Time complexity of the proposed algorithm is $O(n^2)$. On the other hand, one of the existing algorithms Gauss Elimination method has time complexity $O(n^3)$ [16]. This proposed algorithm takes less time than Gauss Elimination Method. Moreover, this proposed algorithm solves problem of system of linear equations using indirect method whereas Gauss Elimination Method solves problem of system of linear equation using direct method. As a result, the solution of proposed algorithm has less round off and other errors than direct method. Besides direct methods lead to a very poor result. This is because of the various types of errors involved in numerical approximations.

5. CONCLUSION

Solving system of linear equations with minimum cost is well defined problem and studied from long times ago. This works goal is to solve the system of linear equations in a cost-effective way. This goal is successfully achieved by minimizing the execution time of the proposed algorithm in comparison with gauss elimination method. But in this proposed algorithm it has assumed that the system must have solution. Modifying the algorithm such that it will work in case of the system has no solution as well as lower the execution time will be the future work. Implement other methods and compare the proposed method with them can be done in future.

CONFLICT OF INTERESTS

The author confirms that there is no conflict of interests associated with this publication and there is no financial fund for this work that can affect the research outcomes.

REFERENCES

[1] Feige U., Reichman D. On Systems of Linear Equations with Two Variables per Equation. In: Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques. *Lecture Notes in Computer Science*. 2004; 3122; 117-127.

[2] Lay D.C. (2005) Linear Algebra and Its Applications. 3rd edition. Addison Wesley.

[3] Berman A. Plemmons R.J. (1994) Nonnegative Matrices in the Mathematical Sciences. SIAM.

[4] Atkinson K. (1989) An Introduction to Numerical Analysis. 2nd edition. New York: John Wiley & Sons.

[5] Althoen S. C, McLaughlin R. Gauss-jordan reduction: a brief history. *In: The American Mathematical Monthly (Mathematical Association of America)*, 1987; 94(2); 130-142.

[6] Mittal R.C., Al-Kurdi A. LU-decomposition and numerical structure for solving large sparse nonsymmetric linear systems. *In: Computers & Mathematics with Applications*. 2002; 43(1-2);131-155.

[7] Goldsztejn A., Chabert G. A Generalized Interval LU Decomposition for the

Solution of Interval Linear Systems. *In: Numerical Methods and Applications. Lecture Notes in Computer Science*. 2007; 4310; 312-319.

[8] Lipschutz S., Lipson M. (2001) Schaum's Outlines: Linear Algebra. Tata Mc-GrawHill edition, Delhi.

[9] Golub G.H., Van Loan CF. (1946) Matrix Computations (3rd ed.). Baltimore: Johns Hopkins.

[10] Helmke U., Hüper K. The Jacobi Method: A Tool for Computation and Control. In: Systems & Control: Foundations & Applications, 1997; 22; 205-228

[11] Rutishauser H. (1971) The Jacobi Method for Real Symmetric Matrices. In: Linear Algebra: Handbook for Automatic Computation.

[12] Hestenes H.R., Stiefel E. Methods of Conjugate Gradients for Solving Linear Systems. In: Journal of Research of the National Bureau of Standards. 1952; 49(6); 409-436.

[13] Hadjidimos A. Successive overrelaxation (SOR) and related methods. *In: Journal of Computational and Applied Mathematics*. 2000; 123; 177-199.

[14] Young D.M. (1971) Iterative Solution of Large Linear Systems. Academic Press.

[15] Kolberg M.L., Krämer W., Zimmer M. Efficient Parallel Solvers for Large Dense Systems of Linear Interval Equations. *Reliab. Comput.* 2011; 15(3); 193-206.

[16] Fang X.G., Havas G. (1997) On the worst-case complexity of integer Gaussian elimination. *In: Proceedings of the 1997 international symposium on Symbolic and algebraic computation (ISSAC '97).* Association for Computing Machinery. New York, NY, USA. p. 28–31.