Scholars Journal of Physics, Mathematics and Statistics **ISSN 2393-8056 (Print)**
Abbreviated Key Title: Sch. J. Phys. Math. Stat. **ISSN 2393-8064 (Online)**

(An International Publisher for Academic and Scientific Resources)

On The Direct and Indirect Methods of Solving Systems of Linear Equations

Ogunlade Temitope Olu*
Department of Mathematics, Ekiti State University Ado-Ekiti, Nigeria

***Corresponding author**
Ogunlade Temitope Olu

Article History
Received: 22.10.2018
Accepted: 29.10.2018
Published: 30.11.2018

DOI:
10.21276/sjpms.2018.5.6.2

Abstract: In this paper various methods are compared for solving linear systems of equations. Both direct and iterative methods are considered. Different direct and indirect methods exist for the computation of linear system of equations. For direct methods, three methods are considered: Crammer's rule, Gaussian elimination and LU (lower and upper triangular matrices) Decomposition. Jacobi method, Gauss-Seidel method, Successive over-relaxation method will be considered for iterative methods. The results show that Successive over-relaxation method is more efficient than other methods considering convergence, number of iterations, memory requirements and accuracy.
Keywords: linear, direct, indirect, methods.

INTRODUCTION

A lot of problems occur in science, and engineering which may be modelled into a system of linear equations. For example, finding the current flowing in some electrical network, determination of stress in a building, characterization of connections in network of roads connecting various cities, etc. When the need to get solutions to these problems arises, economical solutions are sought.

A system of n-linear equations in n-unknowns has a general form:

$$\left.\begin{array}{l} b_{11}X_1 + b_{12}X_2 + b_{13}X_3 + \cdots + b_{1n}X_n = d_1 \\ b_{21}X_1 + b_{22}X_2 + b_{23}X_3 + \cdots + b_{2n}X_n = d_2 \\ \vdots \\ b_{n1}X_1 + b_{n2}X_2 + b_{n3}X_3 + \cdots + b_{nn}X_n = d_n \end{array}\right\} (1.1)$$

Where b_{ij} are constant coefficient $i \leq j \leq n$ and $d_1, d_2, \ldots, d_n$ are given real constants in the system of n- linear algebraic equation in n- unknowns with n- variable or unknown $X_1, X_2, \ldots, X_n$ the solution of the linear equation can be classified into the following ways; (i) if a system of linear equations solved and solved and unique value for $X_1, X_2, \ldots, X_n$ are obtained then, the system is consistent and independent (ii) if on the other hand the system has no definite solution then, it is said to be inconsistent and (iii) where there are infinitely many solutions to the linear system, it is said to be consistent but dependent. Different direct and indirect methods exist for the computation of linear system of equations. Many authors like Rajasekeran [1], Kalambi, [2 and Turner [3] investigate the solutions of linear equations by direct and indirect methods. Many scientific and engineering domains of computation may take the form of linear equations. Bakari1 & Dahiru [4] worked on iterative methods used for solving sparse and dense system of linear equation by considering Jacobi method and Gauss-Seidel methods. Their results show that Gauss-Seidel method is more efficient than Jacobi method by considering maximum number of iteration required to converge and accuracy. The equations in this field may contain large number of variables and hence it is important to solve these equations in an efficient manner [5].

DIRECT METHOD

In direct methods, consideration will be on Crammer's rule, Gaussian elimination and LU (lower and upper triangular matrices) Decomposition.

CRAMMER'S RULE

Crammer's rule is an explicit formula for the solution of a system of linear equations with as many equations as unknowns, valid whenever the system has a unique solution. It expresses the solution in terms of the determinants of from it by replacing one column by the vector of the equations.
Consider a two linear equation:

$$ax_1 + bx_2 = p \quad (2.1)$$
$$cx_1 + dx_2 = q \quad (2.2)$$

With condition that $ab - bc = \propto$, we can solve for the variable x_1 by eliminating the variable x_2. This is accomplished by multiplying the top equation by d and the bottom equation by b and then subtract.
i.e

$$\begin{array}{r} adx_1 + bdx_2 = pd \\ -bcx_1 + bdx_2 = qb \\ \hline adx_1 - bcx_2 = pd - qb \end{array} \qquad (2.3)$$

Hence,

$$(ad - bc)x_1 = pd - qb \qquad (2.4)$$

And we can solve for x_1 and obtain

$$x_1 = \frac{pd - qb}{ad - bc}$$

Similarly,

$$x_2 = \frac{aq - pc}{ad - bc}$$

Which can be expressed (using determinant) as

$$x_1 = \frac{\begin{vmatrix} p & b \\ q & d \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}}, x_2 = \frac{\begin{vmatrix} a & p \\ c & q \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}}$$

Where
$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ is the determinant of $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$
$\begin{vmatrix} p & b \\ q & d \end{vmatrix}$ is the determinant of $\begin{pmatrix} p & b \\ q & d \end{pmatrix}$
$\begin{vmatrix} a & p \\ c & q \end{vmatrix}$ is the determinant of $\begin{pmatrix} a & p \\ c & q \end{pmatrix}$

GAUSSIAN ELIMINATION

Gaussian elimination is an algorithm for solving system of linear equations. The process of Gaussian elimination has two parts. The first part (Forward Elimination) reduces a given system to either triangular or echelon form, or results in a degenerate equation, indicating the system has no unique solution but may have multiple solutions (rank<order). This is accomplished through the use of elementary row operations. The second step uses back substitution to find the solution of the system above.

Stated equivalently for matrices, the first part reduces a matrix to row echelon form using elementary row operations while the second reduces it to reduced row echelon form, or row canonical form.

Consider a linear equation AX = B which is an upper–triangle system and the diagonal element are non-zero.

$$\begin{array}{l} a_n X_1 + a_{12}X_2 + a_{13}X_3 + \ldots + a_{1N-1}X_{N-1} + a_{1N}NX_N = b_1 \\ \qquad a_{22}X_2 + a_{23}X_3 + \ldots + a_{2N-1}X_{N-1} + a_{2N}NX_N = b_2 \\ \qquad\quad a_{33}X_3 + \ldots + a_{3N-1}X_{N-1} + a_{3N}X_N = b_3 \\ \vdots \\ a_{N-1,N-1}X_{N-1} + a_{N-1,N}X_N = b_{N-1} \\ \qquad\quad a_{N,N}X_N = b_N \text{ (Upper-triangle system)} \end{array}$$

If $a_i, i \neq 0 \; for \; i = 1,2,3,\cdots,N$ then, there exists a unique solution.

The last equation involves only X_N, so we solve it first.

$$X_N = \frac{b_N}{a_N b_N}$$

Now X_N is known and it can be used in the next to last equation.

$$X_{N-1} = \frac{b_{N-1} - a_{N-1}NX_N}{a_{N-1,N-1}}$$

Now X_N and X_{N-1} are used to find X_{N-2}

$$X_{N-2} = \frac{b_{N-2} - a_{N-2,N-1}X_{N-1} - a_{N-2}NX_N}{a_{N-2,N-2}}$$

Once the value X_N, X_{N-1},..., X_{i-1} are known, the general step is

$$X_i = \frac{b_i - \sum_{j=i+1}^{N} a_{i,j}X_j}{a_{i,j}}$$

For $i = N-1, N-2, \ldots, 1$

The uniqueness of the solution is easy to be seen. The Nth equation implies ${}^{b_N}/_{a_N}$, N is the only possible value of X_N, so X_{N-1}, X_{N-2}, ..., X_1 are unique.

LU- DECOMPOSITION METHOD

In linear algebra, LU decomposition (also called LU factorization) factorizes a matrix as the product of a lower triangular matrix and an upper triangular matrix. The product sometimes includes a permutation matrix as well. The LU decomposition can be viewed as the matrix form of Gaussian elimination. Computers usually solve square systems of linear equations using the LU decomposition, and it is also a key step when inverting a matrix, or computing the determinant of a matrix.

Let A be a square matrix. An LU- decomposition is a decomposition of the form

$$A = LU \qquad (2.3.1)$$

L ⇒ lower triangular matrix

U ⇒ upper triangular matrix

This means that L has only zeros above the diagonal and U has only zeros below the diagonal.

$$L = \begin{pmatrix} 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 \\ a_{31} & a_{32} & 0 & 0 \\ a_{41} & a_{42} & a_{43} & 0 \end{pmatrix} \qquad (2.3.2)$$

$$U = \begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad (2.3.3)$$

Let

$$AX = P \qquad (2.3.4)$$

Substituting equation (2.3.1)

$$LUX = P \qquad (2.3.5)$$

If UX is set equal to Z,

$$LZ = P \qquad (1.2.3.7)$$

Equation (2.3.7) is solved by back substitution to give X.

ITERATIVE METHODS

An iterative method is a mathematical procedure that generates a sequence of improving approximate solutions for a class of problems.

Some well-known iterative schemes, which will be discussed in this paper, are Jacobi method, Gauss-Seidel method, Successive over-relaxation method.

JACOBI ITERATIVE METHOD

The Jacobi method is an algorithm for determining the solutions of a system of linear equations with largest absolute values in each row and column dominated by the diagonal element. Each diagonal element is solved for, and an approximate value plugged in. The process is then iterated until it converges. Consider the general system of linear algebraic equations, $Ax = b$, written in index

Notation

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

$(i = 1, 2 \ldots\ldots n)$ (2.4.1)

In Jacobi iteration, each equation of the system is solved for the component of the solution vector associated with the diagonal element, that is, x_i. Thus,

$$x_i = \frac{1}{a_{ij}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j - \sum_{j=i+1}^{n} a_{ij}\, x_j\right) \quad (2.4.2)$$

$(i = 1,2 \ldots\ldots n)$

An initial solution vector $X^{(0)}$ is chosen. The superscript in parentheses denotes the iteration number, with zero denoting the initial solution vector. The initial solution vector $X^{(0)}$ is substituted into Eq. (2.4.2) to yield the first improved solution vector $X^{(1)}$. Thus,

$$X_i{}^1 = \frac{1}{a_{ij}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j{}^{(0)} - \sum_{j=i+1}^{n} a_{ij}\, x_j{}^{(0)}\right)$$

$(i = 1,2 \ldots\ldots n)$ (2.4.3)

This procedure is repeated (i.e., iterated) until some convergence criterion is satisfied. The Jacobi algorithm for the general iteration step (k) is:

$$X_i{}^{(K+1)} = x_i{}^{(k)} + \frac{1}{a_{ij}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j{}^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j{}^{(k)}\right)$$

$(i = 1,2, n)$ (2.4.4)

GAUSS-SEIDEL METHOD

The Gauss-Seidel method of solving for a set of linear equations can be thought of as just an extension of the Jacobi method. Start out using an initial value of zero for each of the parameters. Then, solve for $X_1{}^{(1)}$ as in the Jacobi method. When solving for$X_2{}^{(1)}$, insert the just computed value for$X_1{}^{(1)}$. In other words, for each calculation, the most current estimate of the parameter value is used. Gauss-Seidel converges about twice as fast as Jacobi, but may still be very slow.

In the Jacobi method, all values of $X^{(k+1)}$ are based on $X^{(k)}$. The Gauss-Seidel method is similar to the Jacobi method, except that the most recently computed values of all x_i are used in all computations .In brief, as better values *of* x_i are obtained, use them immediately. Like the Jacobi method, the Gauss-Seidel method requires diagonal dominance to ensure Convergence. The Gauss-Seidel algorithm is obtained from the Jacobi algorithm, Eq. (2.4.4), by using $x_j{}^{(k+1)}$ values in the summation from j = 1 to $i - 1$ (assuming the sweeps through the equations proceed from $i = 1$ to n). Thus,

$$X_i{}^{(K+1)} = x_i{}^{(k)} + \frac{1}{a_{ij}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j{}^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j{}^{(k)}\right)$$

$(i = 1\ 2, .,n)$ (2.5.1)

Equation (2.50.1) can be written in terms of the residuals R_i by adding and subtracting $x_i{}^{(k)}$ from the right-hand side of the equation and rearranging to yield

$$X_i{}^{(K+1)} = x_i{}^{(k)} + \frac{R_i{}^{(k)}}{a_{ij}}$$

(i = 1,2 ..... n) (2.5.2)

$$R_i{}^{(k)} = b_i - \sum_{j=i+1}^{n} a_{ij}\, x_j{}^{(k)}$$

(i = 1,2 ..... n) (2.5.3)

The Gauss-Seidel method is sometimes called the method of successive iteration because the most recent values of all x_i are used in all the calculations.

SUCCESSIVE OVER-RELAXATION (SOR) METHOD

The Gauss-Seidel method can be modified to include over-relaxation simply by multiplying the residual $R_i^{(k)}$ in Eq. (2.5.2), by the over-relaxation factor, ω. Thus, the Successive – over –relaxation method is given by,

$$X_i^{(K+1)} = x_i^{(k)} + \omega \frac{R_i^{(k)}}{a_{ij}}$$

(= 1,2 ..... n) (2.6.1)

$$R_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k+1)} - \sum_{j=1}^{n} a_{ij}\, x_j^{(k)} \qquad (2.6.2)$$

When ω = 1 (2.6.1) yields the Gauss-Seidel method. When 1 < ω < 2, the system of equations is over-relaxed. Over-relaxation is appropriate for systems of linear algebraic equations. When ω < 1.0, the system of equations is under relaxed. Under-relaxation is appropriate when the Gauss-Seidel algorithm causes the solution vector to overshoot and move farther away from the exact solution. This behaviour is generally associated with the iterative solution of systems of nonlinear algebraic equations. The iterative method diverges if ω > 2.

ANALYSIS OF RESULTS

The efficiency of the three methods iterative methods will be compared in this section.

JACOBI ITERATION METHOD

To illustrate the Jacobi iteration method, let's solve the following system of linear algebraic equations:
Example 1:

$10X_1 - 8X_2 = -6$, $-8X_1 + 10X_2 - X_3 = 9$ and $-X_2 + 10X_3 = 28$

Table-1: Solution by the Jacobi Iteration Method

Iteration	X(1)	X(2)	X(3)
0	0	0	0
1	-0.6	0.9	2.8
2	0.12	0.7	2.89
3	-0.04	1.285	2.87
4	0.428	1.155	2.9285
5	0.324	1.53525	2.9155
6	0.6282	1.45075	2.953525
7	0.5606	1.697913	2.945075
8	0.75833	1.642988	2.969791
9	0.71439	1.803643	2.964299
⋮	⋮	⋮	
52	0.999981	1.999973	2.999998
62	0.999998	1.999997	3
63	0.999997	1.999998	3
64	0.999999	1.999998	3
65	0.999998	1.999999	3

Table-2: Solution by the Jacobi Iteration Method

Iteration	x(1)	x(2)	x(3)	x(4)	x(5)
0	0	0	0	0	0
1	0.25	0.5	0.5	0.5	0.5
2	0.375	0.6875	0.75	0.6875	0.75
3	0.421875	0.78125	0.84375	0.78125	0.84375
4	0.445313	0.816406	0.890625	0.816406	0.890625
5	0.454102	0.833984	0.908203	0.833984	0.908203
6	0.458496	0.840576	0.916992	0.840576	0.916992
7	0.460144	0.843872	0.920288	0.843872	0.920288
8	0.460968	0.845108	0.921936	0.845108	0.921936
9	0.461277	0.845726	0.922554	0.845726	0.922554
⋮	⋮	⋮	⋮	⋮	⋮
19	0.461538	0.846154	0.923077	0.846154	0.923077

Example 2: Consider another example;

$$4X_1 - X_2 = 1$$
$$-X_1 + 4X_2 - X_3 = 2$$
$$-X_2 + 4X_3 - X_4 = 2$$
$$-X_3 + 4X_4 - X_5 = 2$$
$$-2X_4 + 4X_5 = 2$$

The gauss-seidel iteration method

Let's rework the problem presented in Example 1 in (3.1.) using Gauss-Seidel iteration.

$$R_1 = -6 - 10X_1 + 8X_2$$
$$R_2 = 9 + 8X_1 - 10X_2 + X_3$$
$$R_3 = 28 + X_2 - 10X_3$$

To initiate the solution, let $X^{(0)T}$ = [0.0 0.0 0.0].

Table-3: Solution by the Gauss-Seidel Iteration Method

Iteration	X_1	X_2	X_3
0	0	0	0
1	-0.6	0.42	2.842
2	-0.264	0.973	2.8973
3	0.1784	1.33245	2.933245
4	0.46596	1.566093	2.956609
5	0.652874	1.71796	2.971796
6	0.774368	1.816674	2.981667
7	0.853339	1.880838	2.988084
8	0.904671	1.922545	2.992254
9	0.938036	1.949654	2.994965
⋮	⋮	⋮	⋮
31	0.999995	1.999996	3

Consider the example 2

$$4X_1 - X_2 = 1$$
$$-X_1 + 4X_2 - X_3 = 2$$
$$-X_2 + 4X_3 - X_4 = 2$$
$$-X_3 + 4X_4 - X_5 = 2$$
$$-2X_4 + 4X_5 = 2$$

Table-4: Solution by the Gauss-Seidel Iteration Method

ITERATION	X(1)	X(2)	X(3)	X(4)	X(5)
0	0	0	0	0	0
1	0.25	0.5625	0.640625	0.660156	0.830078
2	0.390625	0.757813	0.854492	0.921143	0.960571
3	0.439453	0.823486	0.936157	0.974182	0.987091
4	0.455872	0.848007	0.955547	0.98566	0.99283
5	0.462002	0.854387	0.960012	0.98821	0.994105
6	0.463597	0.855902	0.961028	0.988783	0.994392
7	0.463976	0.856251	0.961259	0.988913	0.994456
8	0.464063	0.85633	0.961311	0.988942	0.994471
9	0.464083	0.856348	0.961323	0.988948	0.994474
10	0.464087	0.856352	0.961325	0.98895	0.994475
11	0.464088	0.856353	0.961326	0.98895	0.994475
12	0.464088	0.856354	0.961326	0.98895	0.994475
13	0.464088	0.856354	0.961326	0.98895	0.994475

The successive-over relaxation method

Let's rework the problem presented in Example 1 in (3.1.) using Successive-Over Relaxation iteration.

$$R_1 = -6 - 10X_1 + 8X_2 \quad (3.10.1)$$
$$R_2 = 9 + 8X_1 - 10X_2 + X_3 \quad (3.10.2)$$
$$R_3 = 28 + X_2 - 10X_3 \quad (3.10.3)$$

To initiate the solution, let $X^{(0)T}$ = [0.0 0.0 0.0].

Table-5: Solution by the Successive-Over Relaxation Iteration Method

Iteration	X(1)	X(2)	X(3)
0	0	0	0
1	-0.72	0.3888	3.406656
2	-0.20275	1.216397	2.824636
3	0.488291	1.644437	2.992405
4	0.761001	1.840762	2.98241
5	0.894932	1.928871	2.994982
6	0.95273	1.968244	2.997193
7	0.978969	1.985824	2.99886
8	0.990597	1.993672	2.999469
9	0.995806	1.997175	2.999767
10	0.998127	1.998739	2.999895
11	0.999164	1.999437	2.999953
12	0.999627	1.999749	2.999979
13	0.999833	1.999888	2.999991
14	0.999926	1.99995	2.999996
15	0.999967	1.999978	2.999998
16	0.999985	1.99999	2.999999
17	0.999993	1.999996	3
18	0.999997	1.999998	3
19	0.999999	1.999999	3
20	0.999999	2	3

For example 2 in 3.1 we have

$$4X_1 - X_2 = 1$$
$$-X_1 + 4X_2 - X_3 = 2$$
$$-X_2 + 4X_3 - X_4 = 2$$
$$-X_3 + 4X_4 - X_5 = 2$$
$$-2X_4 + 4X_5 = 2$$

Table-6: Solution by the Successive-Over Relaxation Iteration Method

iteration	X(1)	X(2)	X(3)	X(4)	X(5)
0	0	0	0	0	0
1	0.275	0.625625	0.722047	0.748563	0.96171
2	0.419547	0.801376	0.904028	0.988222	0.997351
3	0.453424	0.843162	0.963228	0.990337	0.99495
4	0.461527	0.857491	0.96183	0.989081	0.994499
5	0.464657	0.856535	0.961361	0.988954	0.994475
6	0.464081	0.856343	0.961321	0.988948	0.994474
7	0.464086	0.856353	0.961326	0.98895	0.994475
8	0.464088	0.856354	0.961326	0.98895	0.994475
9	0.464088	0.856354	0.961326	0.98895	0.994475

CONCLUSION

The three main iterative methods for solving linear equation have been presented; these are Successive-Over Relaxation, the Gauss-Seidel and the Jacobi technique. In the examples considered the analysis of results shows that Jacobi method takes 65 iterations to converge for example 1 and 19 iterations to converge the example 2. The analysis of results also shows that Gauss-Seidel method takes about 31 iterations to converge for the example 1 and 13 iterations to converge for example 2. More so, the analysis of results also shows that Successive-Over Relaxation takes about 20 iterations to converge for the example 1 and 9 iterations to converge for example 2 to converge, as compared to other method, within the same tolerance factor. This shows that Successive-Over Relaxation requires less computer storage than the Jacobi method and Gauss - Seidel method. Thus, the Successive-Over Relaxation could be considered more efficient of the three iterative methods considered in this paper.

REFERENCES

1. Rajasekaran S. Numerical methods in Science and Engineering. S. Chand. 2003.
2. Kalambi IB. Solutions of Simultaneous Equations by Iterative Methods. Postgraduate Diploma in Computer Science Project. Abubakar Tafawa Balewa University, Bauchi. 1998.
3. Turner PR. 'Numerical Analysis'. Macmillian Press Ltd. Houndsmills. 1994.
4. Bakari AI and Dahiru IA. Comparison of Jacobi and Gauss-Seidel Iterative Methods for the Solution of Systems of Linear Equations. Asian Research Journal of Mathematics. 2018; 8(3): 1-7.
5. Suriya G, Syeda RA, Rabia K, Nargis M & Memoona K (2015). System of Linear Equations, Guassian Elimination. Global Journal of Computer Science and Technology(C). 15 (5):23-26.