SOLUTION OF LINEAR ALGEBRAIC SYSTEMS BY STATIONARY ITERATIVE METHODS: A SURVEY

Supported by tetfund TERTIARY EDUCATION TRUST FUND

A. A. Ibrahim[1]*, A. A. Alabi[2] and R. O. Osinowo[1]
[1]Department of Mathematical Sciences, Oduduwa University, Ipetumodu, Ile-Ife, Nigeria
[2]Department of Computer Science and Engineering, the Polytechnic, Ile-Ife, Nigeria
*Corresponding author: adebisiibrahim97@gmail.com

Received: November 26, 2019 **Accepted:** February 19, 2020

Abstract: We consider in this paper the application of stationary iterative methods; the Jacobi, the Gauss-Seidel and the SOR iterations for solving linear algebraic systems. The formulae of the three methods are derived from the general form of matrix decomposition. The rates of convergence of these methods are tested with examples and their execution times are examined in relation to the structure of their iteration matrices. The results show the differences in the rates of convergence and execution times.
Keywords: Jacobi iteration, Gauss-Seidel iteration, successive over-relaxation method, convergence

Introduction

Iterative methods are methods which find successive approximations from a given initial approximation to a given linear system of equations. Iterative methods were developed for solving large sparse linear systems arising from finite-difference discretization of partial differential equations, linear equations for linear least-squares problems as well as systems of linear inequalities arising from linear programming. They can as well be applied in the solution of nonlinear systems of equations (Kelly, 1995; Smith, 2004; Bamigbola & Ibrahim, 2014).

Similarly, in modern computing, the locality of a program's data can significantly affect its performance. For instance, some reordering transformation can improve the data locality for stationary iterative methods such as Gauss-Seidel method for solving linear systems with sparse coefficient matrix (Stout *et al*., 2004).

There are broadly two classes of iterative methods: stationary and non-stationary methods. The stationary methods include the Jacobi, the Gauss-Seidel the Successive Over-relaxation (SOR) and the Symmetric Successive Over-relaxation (SSOR) iterations while the non-stationary methods include the Conjugate Gradient method (CGM), the Bi-

Conjugate Gradient method (BCGM), the Minimal Residual method (MINRES), the Generalized Minimal Residual method (GMRES) and several other methods with their variants (Saad, 2000).

The Jacobi, the Gauss-Seidel the Successive-over-relaxation (SOR) and the Symmetric Successive Over-relaxation (SSOR) iterations are mostly applied in the solution linear algebraic system of equations.

Linear algebraic systems are usually of the form;

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots a_{2n}x_n &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots a_{3n}x_n &= b_3 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots a_{nn}x_n &= b_n \end{aligned}\right\} \quad (1)$$

Usually, arising from real life problems whose solution is of paramount importance. The above system (1.1) can be written more compactly as;

$$Ax = b \quad (2)$$

The method is best suited for a linear system whose coefficient matrix is a band matrix. Again, the choice of iterative method depends on the structure of the coefficient matrix A

The Jacobi, Gauss-Seidel and the SOR methods are guaranteed to converge to the exact solution if the coefficient matrix is diagonally dominant (Kelly, 1995; Saad, 2000).

Iterative Methods

The use of iterative methods requires that the coefficient matrix A be of the form $A = P - Q$ where P is invertible. Thus the linear system (1.1) becomes;

$$Px = Qx + b \quad (3)$$

which yields the iterative scheme

$$x^{(k+1)} = P^{-1}Qx^{(k)} + P^{-1}b \quad (4)$$

and $P^{-1}Q$ is called the iteration matrix.

Splitting A into its strictly lower triangular matrix L, diagonal matrix $D$and strictly upper triangular matrix U we have;

$$L = \begin{cases} a_{ij} & i > j \\ 0 & otherwise \end{cases}$$

$$\mathrm{D} = \begin{cases} a_{ij} & i = j \\ 0 & otherwise \end{cases}$$

$$U = \begin{cases} a_{ij} & i < j \\ 0 & otherwise \end{cases}$$

Thus (1.1) becomes;

$$(L + D + U)x = b \quad (5)$$

Definition 1: A square matrix is called singular if its determinant is zero (Brownson, 1989).

Lemma 1: The determinant of a square matrix is zero if it has at least a zero row or a zero column (Brownson, 1989).

Jacobi iteration

The Jacobi method involves using a current approximation $x^{(k)}$ **where** $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \ldots, x_n^{(k)})$ in each of the n equations to find the next approximation. That is, the current values $x_2^{(k)}, x_3^{(k)}, \ldots, x_n^{(k)}$ are used to find a new value $x_1^{(k+1)}$. Similarly the current values $x_1^{(k)}, x_3^{(k)}, \ldots, x_n^{(k)}$ are used in the second equation to obtain a new value $x_2^{(k+1)}$such that in the ith equation, current values of $x_1^{(k)}, x_2^{(k)}, \ldots x_{i-1}^{(k)}, x_{i+1}^{(k)} \ldots, x_n^{(k)}$ are used in the equation to find the new value $x_i^{(k+1)}$. That is given current values $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, \ldots, x_n^{(k)})$ we find new values for $x^{(k+1)} = (x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k+1)}, \ldots, x_n^{(k+1)})$.

In essence, Jacobi method is;

$$\left.\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}\left\{-\left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{14}x_4^{(k)} + \cdots + a_{1n}x_n^{(k)}\right) + b_1\right\} \\ x_2^{(k+1)} &= \frac{1}{a_{22}}\left\{-\left(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + a_{24}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right) + b_2\right\} \\ x_3^{(k+1)} &= \frac{1}{a_{33}}\left\{-\left(a_{31}x_1^{(k)} + a_{32}x_2^{(k)} + a_{34}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right) + b_3\right\} \\ &\vdots \\ x_n^{(k+1)} &= \frac{1}{a_{nn}}\left\{-\left(a_{n1}x_2^{(k)} + a_{n2}x_3^{(k)} + a_{n3}x_4^{(k)} + \cdots + a_{n,n-1}x_{n-1}^{(k)}\right) + b_n\right\} \end{aligned}\right\}$$

and simplifies to;

$$x^{(k+1)} = \frac{1}{a_{ii}}\left(-\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + \mathrm{b}\right) \quad (6)$$

which is expressed as

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

This is the Jacobi iterative method. This can be expressed as;

$$x^{(k+1)} = Tx^{(k)} + c \text{ (Kreyszig, 2011)}$$

Where $T = -D^{-1}(L+U)$ is the iteration matrix

Thus;

$$L + U = \begin{cases} a_{ij} & i \neq j \\ 0 & i = j \end{cases},$$

$$\mathrm{D} = \begin{cases} a_{ij} & i = j \\ 0 & otherwise \end{cases},$$

$$D^{-1} = \begin{cases} \dfrac{a_{ij}}{a_{ii}} & i \neq j \\ 0 & otherwise \end{cases}$$

Lemma 1: If a row or a column of a square matrix is zero, then its determinant is zero (Brownson, 1989).

Lemma 2: If the determinant is zero, then the matrix is singular (Brownson, 1989).

Theorem: The iteration matrix of the Jacobi scheme is non-singular (Strong, 2004).

The Jacobi iteration is guaranteed to converge when the coefficient matrix is diagonally dominant i.e.

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| < |a_{ii}| \quad (7)$$

Gauss-Seidel iteration

The Gauss-Seidel iterative scheme is an improvement over the Jacobi method. Unlike the Jacobi scheme, the computed $x_{i-1}^{(k+1)}$ values are used to obtain $x_i^{(k+1)}$ values since they are better approximations to the exact solution than the $x_{i-1}^{(k+1)}$ values. The Gauss-Seidel scheme is derived as which implies that taking i = 1, 2, 3, …, n, we have;

$$\left.\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}\left\{-\left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + a_{14}x_4^{(k)} + \cdots + a_{1n}x_n^{(k)}\right) + b_1\right\} \\ x_2^{(k+1)} &= \frac{1}{a_{22}}\left\{-\left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + a_{24}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right) + b_2\right\} \\ x_3^{(k+1)} &= \frac{1}{a_{33}}\left\{-\left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{34}x_4^{(k)} + \cdots + a_{2n}x_n^{(k)}\right) + b_3\right\} \\ &\vdots \\ x_n^{(k+1)} &= \frac{1}{a_{nn}}\left\{-\left(a_{n1}x_2^{(k+1)} + a_{n2}x_3^{(k+1)} + a_{n3}x_4^{(k+1)} + a_{n,n-1}x_{n-1}^{(k+1)}\right) + b_1\right\} \end{aligned}\right\}$$

That is;

$$x^{(k+1)} = \frac{1}{a_{ii}}\left(-\sum_{j=1}^{i-1} a_{ij}\,x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}\,x_j^{(k)} + b\right) \quad (8)$$

Thus we have;

$$x^{(k+1)} = D^{-1}\left\{-Lx^{(k+1)} - Ux^{(k)} + b\right\}$$

Since $DD^{-1} = I$, the identity matrix, equation (14) yield

$$(L+D)x^{(k+1)} = -Ux^{(k)} + b$$

$$Lx^{(k+1)} + Dx^{(k+1)} = -Ux^{(k)} + b$$

$$x^{(k+1)} = -(L+D)^{-1}Ux^{(k)} + -(L+D)^{-1}b$$

$G = -(L+D)^{-1}U$ and $\beta = -(L+D)^{-1}b$ then we

$$x^{(k+1)} = Gx^{(k)} + \beta \quad (9)$$

The SOR iteration

Consider the linear system (1.2) and from (2.3) we have;

$$\omega(L+D+U)x = \omega b$$

$$(L+D)x = -Ux + b$$

$$\omega(L+D)x = -\omega Ux + \omega b$$

$$0 = -\omega Lx - \omega Dx - \omega Ux + \omega b$$

Adding Dx to both sides and simplifying yields;

$$Dx = Dx - \omega Lx - \omega Dx - \omega Ux + \omega b$$

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

$$Dx^{(k+1)} = D(1-\omega)x^{(k)} + \omega\left((-L-U)x^{(k)} + b\right)$$

Multiplying through by D^{-1};

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega D^{-1}\left((-L-U)x^{(k)} + b\right)$$

$$x_i^{(k+1)} = (1-\omega)\,x_i^{(k)} + \frac{\omega}{a_{ii}}\left(-\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + \mathrm{b}\right) \quad (10)$$

This scheme is a generalization of the Gauss-Seidel iteration with the introduction of the relaxation parameter ω to speed up convergence. Indeed, if $\omega = 1$, (11) reduces to Gauss-Seidel iteration.

Numerical examples

In this section, we consider the application of the three iterative schemes discussed to linear algebraic systems. From the results obtained, the number of iterations and the computer execution time for each problem in Table 1 is shown in Table 2.

Table 1: The problems

S/N	Matrix *A*	Vector *b*	Dimension
1	$\begin{pmatrix} 7 & 2 & 0 & 0 & 0 \ldots & 0 \\ 1 & 7 & 2 & 0 & 0 \ldots & 0 \\ 0 & 1 & 7 & 2 & 0 \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0\ 1 & 7 \end{pmatrix}$	$\begin{pmatrix} 9 \\ 10 \\ \ldots \\ \ldots \\ 10 \\ 9 \end{pmatrix}$	$(50X50)$
2	$\begin{pmatrix} 8 & 1 & 0 & 0 & 0 \ldots & 0 \\ 1 & 8 & 1 & 0 & 0 \ldots & 0 \\ 0 & 1 & 8 & 1 & 0 \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0\ 1 & 8 \end{pmatrix}$	$\begin{pmatrix} 0.9 \\ 1 \\ \ldots \\ \ldots \\ 1 \\ 0.9 \end{pmatrix}$	$(75X75)$
3	$\begin{pmatrix} 4 & -1 & 0 & 0 & 0 \ldots & 0 \\ -1 & 4 & -1 & 0 & 0 \ldots & 0 \\ 0 & -1 & 4 & -1 & 0 \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 & -1 \quad 4 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 2 \\ \ldots \\ \ldots \\ 2 \\ 3 \end{pmatrix}$	$(125X125)$

Table 3.2: The solutions

S/N	Jacobi			Gauss-Seidel			SOR		
	Solution	**Iter. No.**	**CPU Time**	**Solution**	**Iter. No.**	**CPU Time**	**Solution**	**Iter. No.**	**CPU Time**
1	$\begin{pmatrix}0.9999\\0.9998\\0.9998\\...\\0.9998\\0.9999\\1.0000\end{pmatrix}$	10	0.0452	$\begin{pmatrix}1.0001\\1.0001\\1.0001\\...\\1.0001\\1.0001\\1.0001\end{pmatrix}$	7	0.0331	$\begin{pmatrix}1.0002\\1.0002\\1.0002\\...\\1.0002\\1.0002\\1.0002\end{pmatrix}$	7	0.0277
2	$\begin{pmatrix}0.1000\\0.1000\\0.1000\\...\\0.1000\\0.1000\\0.1000\end{pmatrix}$	7	0.0893	$\begin{pmatrix}0.1000\\0.1000\\0.1000\\...\\0.1000\\0.1000\\0.1000\end{pmatrix}$	5	0.0662	$\begin{pmatrix}0.1000\\0.1000\\0.1000\\...\\0.1000\\0.1000\\0.1000\end{pmatrix}$	5	0.0649
3	$\begin{pmatrix}0.9999\\0.9998\\0.9998\\...\\0.9999\\0.9999\\1.0000\end{pmatrix}$	10	0.1133	$\begin{pmatrix}1.0000\\0.9997\\0.9998\\...\\0.9999\\1.0000\\1.0000\end{pmatrix}$	7	0.0826	$\begin{pmatrix}0.9999\\0.9999\\0.9999\\...\\0.9999\\1.0000\\1.0000\end{pmatrix}$	7	0.0817

Discussion of Results

The three methods explored in this paper were tested with three different linear systems of equations of varying sizes. The results show that while the Jacobi method converges after ten iterations in example 1, seven iterations in example 2 and ten iterations in example 3, both the Gauss-Seidel and the SOR converge after seven iterations in example 1, five iterations in example 2 and seven iterations in example 3.

Thus, both the Gauss-Seidel and the SOR converge faster than the Jacobi method in terms of number of iterations and execution time. Again, though Gauss-Seidel and the SOR converge at the same number of iterations in the given examples, the SOR converges faster in terms of execution time.

Conflict of Interest

Authors declare there is no conflict of interest related to this study.

References

Bamigbola OM & Ibrahim AA 2014. On algebraic structure of improved Gauss-Seidel iteration. *Int. J. Math., Comput., Physical and Quantum Engr.*, 8(10).

Brownson R 1989 . Theory and Problems of Matrix Operations Schaum Outline Series. McGraw-Hill, New York, pp. 34-51.

Ibrahim AA 2014. Characterization and Application of Stationary Iterative Methods. University of Ilorin, Nigeria: Unpublished Ph.D. Thesis.

Carnahan, B. Applied numerical analysis. New York: John Wiley and Sons, 1980.

Kelly CT 1995. Iterative Methods for Linear and Nonlinear Equations. Philadelphia: SIAM, pp. 3-60.

Kreyszig E 2011. Advanced Engineering Mathematics. John Wiley and Sons, New York, pp. 34-51.

Meyer CD 2000. Matrix Analysis and Applied Linear Algebra. Philadelphia, PA: Society for Industrial and Applied Mathematics, pp. 34-51).

Saad Y 2000. Iterative Methods for Sparse Linear Systems, 2nd edition. New York: PWS Publishing, pp. 105-132.

Smith GD 1978. Partial Differential Equations. Oxford University Press, Oxford, pp. 48-75.

Strong DM 2004. Iterative methods for solving linear systems. *J. Math. Assoc. Am.*, 1-17.

Stout M, Carter J & Ferrante B Kreaseck 2004. Sparse tiling for stationary iterative methods. *Int. J. High Performance Comp. Applic.*, 18(1).

Stroud K & Booth DJ 2003. Further Engineering Mathematics. Palgrave, New York, pp. 34-51.