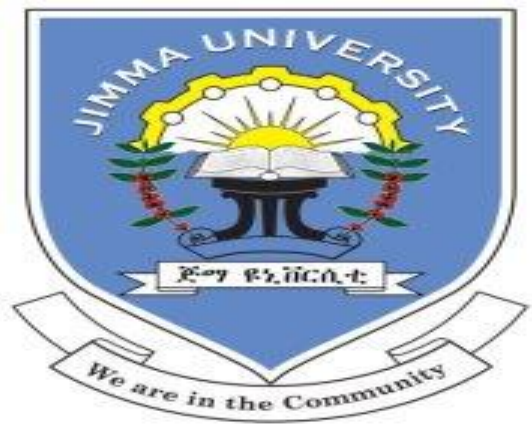
COMPARISON OF GENERALIZED ITERATIVE METHODS FOR SOLVING SYSTEMS OF LINEAR EQUATIONS



A THESIS SUBMITTED TO THE DEPARTMENT OF MATHEMATICS, JIMMA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE REQUIREMENTS OF DEGREE OF MASTERS SCIENCE IN MATHEMATICS (**NUMERICAL ANALYSIS STREAM**)

BY: MERGA GULUMA

ADVISOR: **GENANEW GOFE** (PhD)

CO-ADVISOR: **HAILU MULETA** (MSc)

OCTOBER, 2015

JIMMA, ETHIOPIA

Declaration

I hereby declare that **Comparison of Generalized Iterative Methods for Solving Systems of Linear Equations** is my own work for the M.Sc. degree in Mathematics and that, to the best of my knowledge, it contains no materials previously published by another person nor material which has been accepted for the award of any other degree of the university, except where due acknowledgement has been made in the study.

_____ _____

Merga Guluma (Candidate) Date

Certified by:

_____ _____

Dr. Genanew Gofe (Supervisor) Date

_____ _____

Ato. Hailu Muleta (Co-advisor) Date

ACKNOWLEDGEMENTS

First of all, I would like to thank the almighty God who is the source of everything in this world. Secondly, I would like to express my deepest gratitude to my advisor, Dr. Genanew Gofe and my co-advisor Ato Hailu Muleta for their genuine comments and supports for successful accomplishment of this research report.

Thirdly, my thanks reach to my colleagues and my darling brothers and sisters for their ideological and financial supports.

Lastly, my thanks go to my classmate who support me morally and psychologically to complete this research report.

Table of Contents

Contents

Acronyms

GJ-Generalized Jacobi method

GGS- Generalized Gauss-Seidel method

GESOR-Generalized Successive over Relaxation method

SPD-Symmetric Positive Definite

SDD-Strictly diagonally Dominant

List of Tables

Abstract

In this study we present a survey of three generalized iterative methods for the solution of the system of linear equations. The study used splitting of matrix, when the matrix satisfies strictly diagonally dominant, symmetric and positive definite property to guarantee the efficiency of the three methods. The thesis shows that the generalized successive over relaxation method is more efficient than the generalized Gauss-Seidel method and is much more efficient than the generalized Jacobi method taking into account the number of iterations, the computational running time and level of accuracy required to converges.

CHAPTER ONE

INTRODUCTION

1.1 Background

Nowadays the foundation of new innovations in the field of computational mathematics is due to the numerical methods that most widely being used to solve the equations arising in the fields of applied medical sciences, engineering and technology.

The essence of computational science is numerical algorithm and/or computational mathematics. In fact, substantial effort in computational sciences has been devoted to the development of algorithms, the efficient implementation in programming languages, and validation of computational results. A collection of problems and solutions in computational science can be found [13,17].

Numerical analysis is the branch of mathematics concerned with the theoretical foundation of numerical algorithms for the solution of problems arising in scientific applications. The subject addresses a variety of questions ranging from the approximation of system of linear algebraic equations, with particular emphasis on the convergence, stability, accuracy, efficiency and reliability of numerical algorithms. In numerical analysis, one of the most important topics that can be studied is systems of linear equations. The systems are relevant for modeling and solving in many branches of knowledge areas such as physics, engineering, health sciences, economics finance and even social sciences [10].

The major factors to be considered in comparing different numerical methods are the accuracy of the numerical solutions and its computational time. They further indicated that it is important to note that the comparison of numerical methods is not so simple because their performance may depend on the characteristic of the problem at hand. It should also be noted that there are other factors to be considered such as stability, versatility, proof against run-time error, and so on which are being considered in most of the MATLAB built-in routines [18].

Even though the various methods are being used to solve systems of linear equations, there is no unique method is best for all conditions. These methods are applied according to their speed and accuracy [12]. In addition to these, speed is a significant factor for the solution of system of linear equations as the volume of computation engaged is large so that the iterative methods very much effective regarding the time requirements and computer storage. These methods need fewer multiplications for huge system of equations as well as have fewer rounds off errors. Moreover, iterative methods are much more appropriate for the solution of system of linear equations when the number of equations in that system is large the iterative methods are fast and efficient [3]. This has encouraged many authors [3, 6, 12] to investigate the solutions of linear system of equations directly and indirectly. Systems of linear equations arise in a large number of areas directly in modeling physical situations and indirectly in the numerical solutions of the other mathematical models.

A general linear iterative method for the solution of the system of equations

$$Ax = b \tag{1}$$

(where A is the coefficient matrix, b is a column vector and x is solution vectors to be determined.), can be written in matrix form as

$$x^{k+1} = Hx^{k} + C \tag{2}$$

where $x^{(k+1)}$ and $x^{(k)}$ are the approximations for x at the $(k+1)^{th}$ and k^{th} iterations respectively. H is called the iteration matrix which depends on A and C is a column vector, which depends on A and b. To solve an nxn linear system of (1) A invertible matrix and an n-vector b of nonzero, then the problem is to find an n-vector x, starting with an initial approximation $x^{(0)}$ to the exact solution x and produces a sequence of approximation $\{x^{k}\}_{k=0}^{\infty}$ that converges to x. That means the exact solution can be obtained from (1) as follows:

$$x = A^{-1}b \tag{3}$$

Generalized iterative methods are defined as a process by which the first computed solution can sometimes be improved to yield a more accurate solution.

Sparse systems are (1) in which large numbers of elements in the coefficient matrix are zero and system of this type arises frequently in numerical solution of boundary value problem. The

numerical methods for the solution of system of linear equation (1) are broadly classified into two categories: Direct methods and iterative methods.

Generalized iterative methods [6] were continued until the residuals stabilize at or very near to zero. In practice one step of generalized iterative methods usually suffices if generalized iterative methods fail to stabilize. It is likely to have meaningful solutions cannot be obtained using conventional computing method.

The criteria considered are convergence rate, number of iterations required, memory requirements and accuracy. Therefore, the purpose of this study is to distinguish the fastest methods among generalized iterative methods that converge to the exact solution for solving large systems of linear equations by taking into account their iteration number, computational running time, accuracy and explain the results comparing among the three methods.

1.2 Statement of the Problem

Mathematical models in terms of system of linear equations with iterative method are common in many fields of science and engineering such as mechanics, electrodynamics, physics and others [11]. Thus, this shows the importance and application of iterative methods to solve problems in real life. It is possible to use an iterative method to find the approximate solution of linear system of large equation that has fewer round-off errors as compared to direct methods.

Thus the purpose of this study is to compare the methods that approximate linear system of equations with number of iteration required and computational running time has been compared for solving linear system of equations provide that the accuracy and efficient of numerical solutions. Therefore, the research tried to answer the following research questions:

1) What are the procedures and techniques to be used in order to verify the efficiency the methods?

2) Among the three methods, which one has the least computational running time?

3) Which one of the three methods gives the most accurate results?

1.3 Objective of the Study

1.3.1 General Objective

The general objective of this study is to compare the efficiency of the generalized iterative methods for solving system of linear equations.

1.3.2 Specific Objectives

The specific objectives of the study are

- To compute iteration numbers, computational running time and accurate numerical solutions of each schemes for solving systems of linear equations.
- To compare the computational time of each method for solving system of linear equations.
- To compare the accuracy of the three methods with exact solutions of system of equations.

1.4 Significance of the Study

The final results of this study may have the following importance:

- It plays prominent role in approximate solution of systems of linear equations using numerical schemes.
- It may give convenient results and information for students, faculty and others regarding iterative methods and which method is better in solving system of linear equations.
- It may provide some background information for other researchers who want to conduct a research on related topics.

Furthermore, this research would be useful for graduate program of the department and built the research skill and scientific communication of the researcher.

1.5 Delimitation of the Study

This study is solely emphasized on the numerical method for solving systems of linear equations that is subjected to generalized Jacobi, Generalized Gauss-Seidel and Generalized Successive over Relaxation method among many other indirect schemes for solving system of linear equations.

CHAPTER TWO

REVIEW OF RELATED LITERATURE

Authors like Turner [12] faced difficulty with Gauss Elimination approach because of round off errors and slow convergence for large systems of equations. To get rid of these problems many Authors like [8] and [12] were encouraged to investigate solutions of linear equations by indirect methods. Most researchers deal with the iterative methods for solving linear systems of equations and inequalities for sparse matrices. A sparse matrix is one whose entries are mostly zero. There are many ways of storing a sparse matrix whichever method is chosen some form of compact data structure is required that avoids storing the numerically zero entries in the matrix. It needs to be simple and flexible so that it can be used in a wide range of matrix operations [14,19].

Various methods converge to the root at different rates. That is, some methods are slow to converge and it takes a long time to arrive at the root, while other methods can lead us to the root faster. This is in general a compromise between ease of calculation and time. The rate at which an iterative method converges depends greatly on the spectrum of the coefficient matrix. Hence, iterative methods usually involve a second matrix that transforms the coefficient matrix into one with a more favorable spectrum. The transformation matrix is called a preconditioner. A good preconditioner improves the convergence of the iterative method, sufficiently to overcome the extra cost of constructing and applying the preconditioner. Indeed, without a preconditioner the iterative method may even fail to converge [15]. The term ‘iterative method’ refers to a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system at each step. There are two major types of iterative methods. Stationary methods are older, simpler to understand and implement; but usually not as effective. Non-stationary methods are a relatively recent development; their analysis is usually harder to understand, but they can be highly effective [15].

The natural idea to take advantage of the zeros of a matrix and their location was initiated by engineers in various disciplines. In the simplest case involving banded matrices, special techniques are straightforward to develop. Electrical engineers dealing with electrical networks in the 1960s were the first to exploit sparsity to solve general sparse linear systems for matrices

with irregular structure. The main issue, and the first addressed by sparse matrix technology, was to devise direct solution methods for linear systems. These had to be economical, both in terms of storage and computational effort. Sparse direct solvers can handle very large problems that cannot be tackled by the usual 'dense' solvers [22].
The matrix-by-vector product is an important operation which is required in most of the iterative solution algorithms for solving sparse linear systems.

The major factors to be considered in evaluating/comparing different numerical methods are the accuracy of the numerical solution and its computational time. They further indicated that it is important to note that the evaluation/comparison of numerical methods is not so simple because their performance may depend on the characteristic of the problem at hand. It should also be noted that there are other factors to be considered such as stability, versatility, proof against run-time error and so on which are being considered in most of the MATLAB built-in routines [18].

Iterative methods have traditionally been used for the solution of large linear systems with diagonally dominant sparse matrices. For such systems the methods of Gauss-Jacobi and Gauss-Seidel could be used with some success, not so much because of the reduction in computational work, but mainly because of the limited amount of memory that is required. Of course, reduction of the computational work was also a serious concern, and this led Jacobi (1846) to apply plane rotations to the matrix in order to force stronger diagonal dominance, giving up sparsity. Jacobi had to solve many similar systems in the context of eigenvalue computations [20].

Different sequential methods (derived mostly from Kalambi's) have been proposed [8]. These methods only consider one equality and inequality at a time and each iterate is obtained from the previous iterate. Various methods have been introduced to solve systems of linear equations. There is no single method that is best for all situations. These methods should be determined according to speed and accuracy [6].

The Jacobi method is one of the methods with a few computations but its rate of convergence is slow as compared to other iterative methods. It is also a method of solving a matrix equation which has non-zero element in its main diagonal. An approximate value can be obtained by

solving each main diagonal element. We shall continue the iteration process until it converges. Generalized Gauss-Seidel method is introduced by Davod K. Salkueh[4] and mentioned that this method is much faster than conventional Gauss-Seidel iterative method. We can use the most recent value in this method. It is fast and simple to use when the coefficient matrix is sparse as well as accuracy is developed in every iteration that is continue the iteration process until the relative error is less than the pre-defined tolerance Kalambi[8].

The successive over relaxation (SOR) algorithm is a stationary iterative method for solving linear system of equation [5 and 7]. In his research a generalization of the successive over relaxation say GESOR has been proposed and its convergence properties have been discussed. Some numerical examples have also been taken to show the efficiency of the proposed method. A couple of years ago, David Khojasteh Salkueh[5] introduced an equivalent splitting which led to the generalized Jacobi, Gauss-Seidel and Successive over relaxation methods.

In this research we have presented the indirect methods for solving large system of linear equations and try to find out the more efficient method for solving these systems of linear equations. The criteria considered are time to converge, number of iteration and memory requirements and accuracy for generalized Jacobi, generalized Gauss-Seidel and generalized Successive over Relaxation methods and compare the result of each method.

2.1. Generalized Jacobi (GJ) Method

It is an adjustment formed by splitting the coefficient matrix of $Ax = b$ from Jacobi iterative method [7]. It is the simplest technique to solve a system of linear equations with largest absolute values in each row and column dominated by the diagonal element. The Jacobi method is one of the methods with a few computations but its rate of convergence is slow as compared to other iterative methods. It is also a method of solving a matrix equation on a matrix which has non-zero element in its main diagonal. An approximate value can be obtained by solving each main diagonal element. We shall continue the iteration process until it converges. It is easily derived by examining each of the equation in the linear system $Ax = b$ in iteration.

As it is discussed by Ibrahim B.Kalambi [6] and Davod K. Salkuyeh [4], that Jacobi method is easier method to use for determination of the n-dimensional solution vector x of linear system

but slow to converge. As Author [4, 9], introduced generalized method of Jacobi, which is more efficient than conventional Jacobi method. More-importantly, the order in which the equations are examined is irrelevant as this method treated them independently. In this method it is not possible to use most recently available information. But in the next step, we can use the recently calculated value. We shall move on until residual difference is less than predefined tolerance. This method is convergent when the coefficient matrix of the system of linear equations is diagonally dominant and that is a necessary condition for this method.

2.2. Generalized Gauss -Seidel (GGS) Method

It is an alteration formed by splitting the coefficient matrix of $Ax = b$ from Gauss-Seidel iterative method and it is mainly an iterative method used to solve a system of linear equations [7]. It is also a method of solving a matrix equation on a matrix that has no zeroes along its main diagonal. In the same manner with Gauss Seidel method, it is also an iterative method used for the solution of linear system of equations. This method is introduced by Davod K. Salkueh [4] and mentioned that this method is much faster than conventional Gauss-Seidel iterative method. Each and every main diagonal element is solved and an approximate value got in. Proceeding with the Gauss Seidel method and supposing that the equations are examined in a sequence and also the previously computed results are used as soon as the process over.

Finally, we start with an initial approximation and substitute the solution in the given equation. We shall use the most recent value in this method. The iteration process is to be continued until the relative error is less than the pre-specified tolerance. It is fast and simple to use as well as accuracy is developed in all iteration. The Gauss-Seidel method is sometimes called the method of successive displacements to indicate the dependence of iterate on the order. It is a necessary condition for the Gauss-Seidel method to have nonzero elements on the main diagonal.

2.3. Generalized Successive over Relaxation (GESOR) Method

It is a decomposition formed by coefficient matrix of eq. (1) from successive over relaxation iterative methods [5].We can use the most recent value in this method. It is fast, simple and efficient to use when the coefficient matrix is sparse. The accurate result is obtained in all iteration that is continuing the iteration process until the residual difference is less than the pre-specified tolerance.

Choosing the value of ω

If $\boldsymbol{\omega}$**=1**, then SOR method simplifies to the Gauss-Seidel method. A theorem due to Kahan [20] shows that SOR fails to converge if $\boldsymbol{\omega}$ is outside the interval (0,2). Though technically the term under-relaxation should be used when $0<\boldsymbol{\omega}<1$, for convenience the term over-relaxation is now used for any value of $1<\boldsymbol{\omega}<2$.

In general, it is not possible to compute in advance the value of $\boldsymbol{\omega}$ that is optimal value for $\boldsymbol{\omega}$**,** the expense of such computation is usually prohibitive [15].

If the coefficient matrix A is symmetric and positive definite, the SOR iteration is guaranteed to converge for any value of $\boldsymbol{\omega}$ between 0 and 2, though the choice of $\boldsymbol{\omega}$ can significantly affect the rate at which the SOR iteration converges. Sophisticated implementations of the SOR algorithm employ adaptive parameter estimation schemes to try to home in on the appropriate value of $\boldsymbol{\omega}$ by estimating the rate at which the iteration is converging. In principle, given the spectral radius ρ of the Jacobi iteration matrix, one can determine a priori the theoretically optimal value $\boldsymbol{\omega}$ for SOR $\omega_{opt} = \frac{2}{1+\sqrt{1-\rho^2}}$ [15].

This is seldom done, since calculating the spectral radius of the Jacobi matrix requires an impractical amount of computation [7, 15].

It was noted that the method will converge if and only if the spectral radius of the iteration matrix $\rho(H)<1$, and the smaller the spectral radius, the faster the convergence. Analysis of the residual vectors of the Gauss-Seidel technique leads to the SOR iterative method which involves a parameter ω to speed convergence [16].

The linear system problem of the form (1) where A is an n×n nonsingular matrix, b is an n-vector and x is an n-vector expected to be found. As Davod K. Salkuyeh[5] stated that successive over relaxation method is convergent for the symmetric positive definite (SPD) matrices if $0<\omega<2$ and ω be a fixed parameter. The results shown by [6 and 8] proved that the successive over-relaxation method is faster than the Gauss Seidel and Jacobi methods because of its performance, number of iterations required to converge and level of accuracy.

CHAPTER THREE

METHODOLOGY

3.1 Study Area and Period

The study was conducted in Jimma University college of Natural Sciences Department of Mathematics in 2014/15 academic year. It focused on comparison of generalized iterative method for solving systems of linear equations.

3.2 Study Design

The study was used documentary analysis and experimental result that obtained by MATLAB software program for the linear system of equations. All algorithms were made in the same condition, which use the same type of processor, having the same memory size, the same operating system, and using the same equation .The processor used is Intel(R) core (TM) i3-4005U CPU @1.70GHZ 1.70GHz with 4GM memory (RAM), with the 64-bit operating system (windows 8.1 home premium) x64-based processor. The language program used is MATLAB version 7.6.0.324(R2008a).

3.3 Source of Information

The data was collected from the relevant source of information to achieve the objective of the study and experimental results were obtained by using MATLAB software program and using the input variables to compare the result of each method.

3.4 Study Procedures

Important materials and data for the study were collected using documentary analysis as an instrument. In order to achieve the intended objectives we followed the following mathematical steps.

- 1^{st} splitting the coefficient matrix of eq. (1) of order n and expressing it as

$$A = T_m - E_m - F_m \tag{4}$$

Where E_m and F_m are strictly lower and strictly upper triangular matrix of order n respectively and T_m= Diagonal matrix of order n.($a_{ij} \neq 0; i = j$)

- 2^{nd} deriving the iterative formula of each method.
- 3^{rd} deriving Iterative generalized formula of each method
- 4^{th} selecting the initial guess and inserting into the 3^{rd} formula.
- 5^{th} Continue the iteration process until residual difference is less than the pre-defined tolerance.
- 6^{th} possible conclusions and recommendations have been drawn.

3.5 Ethical Consideration

Regarding the ethical issue, official letters were taken from concerned body in order to get the existing materials such as books, journals and published articles from the library and lab of the nearby University

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Preliminary

Consider the linear system of equations

$$AX = b \tag{5}$$

where the matrix A$\in R^{nXn}$ and x,b$\in R^{n}$

Let A be a non-singular matrix with non-zero diagonal entries and $A = D - L - U$, where D is the diagonal of A, -L its strictly lower part and –U its strictly upper part of A [1].

Then the Jacobi, Gauss-Seidel and Successive over Relaxation methods for solving equation (5) are defined as follows:

$$X^{k+1} = D^{-1}(L+U)X^{k} + D^{-1}b \tag{6}$$

$$X^{k+1} = (D-L)^{-1}UX^{k} + (D-L)^{-1}b \tag{7}$$

$$X^{k+1} = (D-\omega L)^{-1}((1-\omega)D + \omega U)X^{k} + \omega(D-\omega L)^{-1}b \tag{8}$$

where $0 < \omega < 2$ is a fixed parameter, respectively.

Definition 4.1 An nxn matrix A is strictly diagonally dominant if the absolute value of each entry on the main diagonal is greater than the sum of the absolute values of the other entries. That is, $|a_{ii}| > \sum_{j=1, i\neq j}^{n} |a_{ij}|$, i=1, 2, 3, …, n[16].

Definition 4.2 The splitting matrix A=M-N is called i) regular if $M^{-1} \geq 0 \; and \; N \geq 0$.

ii) Weak regular if $M^{-1} \geq 0 \; and \; M^{-1}N \geq 0$.

iii) Convergent splitting if $\rho(M^{-1}N) < 1$[7].

Definition 4.3 Let A, M, N be three given matrices satisfying A=M-N. The pair of matrices M, N is a regular splitting of A, if M is nonsingular and $M^{-1}$and N is nonnegative.

Definition 4.4 The spectral radius ρ(A) of a matrix A is defined by the maximum modulus of λ, where λ is an eigenvalue of A. (For complex $\lambda = \alpha + i\beta$, we define $|\lambda| = (\alpha^2 + \beta^2)^{\frac{1}{2}}$).

Definition 4.5 A matrix A is said to be an M-matrix if it satisfies the following four properties:

i) $a_{ii} > 0$ for i=1,2,3,…,n.

ii) $a_{i,j} \leq 0$ for i ≠ j, i,j=1,2,3,…,n.

iii) A is nonsingular.

iv) $A^{-1} \geq 0$

Definition 4.6. An nxn matrix is called a band matrix if integers p and q, with 1<p,q<n, exist with the property that $a_{ij} = 0$ whenever $p \leq j - i$ or $q \leq i - j$. The band width of a band matrix is defined as w=p+q-1[16].

Definition 4.7 A matrix A is positive definite if it is symmetric and if $x^t Ax > 0$ for every n-dimensional vector $x \neq 0$.

Definition 4.8 An nxn matrix A convergent if $\lim_{k\to\infty}(A^k)_{ij} = 0$, for each i=1,2,3,…,n and j=1,2,3,…,n

Corollary If H is a weak diagonally dominant matrix, then the methods of generalized SOR and generalized Jacobi converges for $0 < \omega \leq 1$.

4.2 Generalized Iterative Methods

Let A=(a_{ij}) be an nxn matrix and T_m=(t_{ij}) be a banded matrix of bandwidth 2m+1 defined as

$$t_{ij}=\begin{cases} a_{ij}, if\ |i-j| \leq m \\ 0\ , otherwise \end{cases}$$

We consider the decomposition $A=T_m-E_m-F_m$ where $-E_m$ and $-F_m$ are the strict lower and strict upper part of the matrix $A-T_m$, respectively. In other words matrices T_m , E_m and F_m are defined as follows. $T_m=\begin{pmatrix} a_{1,1} & \cdots & a_{1,m+1} & & \\ \vdots & \ddots & & \ddots & \\ a_{m+1,1} & & \ddots & & a_{n-m,n} \\ & \ddots & & \ddots & \vdots \\ & a_{n,n-m} & \cdots & & a_{n,n} \end{pmatrix}$, $E_m=\begin{pmatrix} & & \\ -a_{m+2,1} & & \\ \vdots & \ddots & \\ -a_{n,1} & \cdots & -a_{n-m-1,n} \end{pmatrix}$,

$F_m=\begin{pmatrix} -a_{1,m+2} & \cdots & -a_{1,n} \\ \ddots & & \vdots \\ & & -a_{n-m-1,n} \end{pmatrix}$.

Then the Generalized Jacobi, Generalized Gauss-Seidel and Generalized Successive over relaxation methods are defined as

$$X^{K+1} = T_m^{-1}(E_m + F_m)X^K + T_m^{-1}b$$

$$X^{K+1} = (T_m - E_m)^{-1}F_m X^K + (T_m - E_m)^{-1}b$$

$$X^{K+1} = (T_m - \omega E_m)^{-1}((1-\omega)T_m + \omega F_m)X^K + \omega(T_m - \omega E_m)^{-1}b$$

Iterative methods are very important to solve systems of linear equations. But all methods are not suitable for some of the equations due to certain conditions. For these reason the study confirm by taking some counter examples that justify the pre-stated criteria. Under these section the study have found how to drive the formulae for stationary iterative methods and special emphasis on generalized iterative methods.

4.2.1 Generalized Jacobi Method

It can be derived from the concept of conventional Jacobi method through regular splitting of the coefficient matrix. The derivation of the method can be illustrated as follows: The coefficient matrix of equation (1) above can be written as $A = T_m - E_m - F_m$.

Then Ax=b becomes after substitution of A takes place

$$\begin{aligned} & (T_m - E_m - F_m)x = b \\ \Leftrightarrow\ & (T_m - (E_m - F_m))x = b \\ \Leftrightarrow\ & T_m x - (E_m + F_m)x = b \\ \Leftrightarrow\ & T_m x = (E_m + F_m)x + b \\ & x^{k+1} = T_m^{-1}(E_m + F_m)x^k + T_m^{-1}b \end{aligned} \qquad (9)$$

Where $T_m^{-1}(E_m + F_m)$ and $T_m^{-1}b$ are an iteration matrix and a column vector of the generalized Jacobi respectively.

4.2.2 Generalized Gauss-Seidel Method

It is formed from the stationary iterative method (which is called Gauss-Seidel) through regular splitting of the coefficient matrix and where the matrix is sparse.

We can derive the formula using either the forward or backward Gauss-Seidel methods. Now let use only the forward Gauss-Seidel method for the derivation of the method.

Here the equation Ax=b and then after substitution and certain manipulation the formula looks like as follows:

$$\begin{aligned} & (\mathrm{T_m} - \mathrm{E_m} - \mathrm{F_m})\mathrm{x} = \mathrm{b} \\ \Leftrightarrow\ & ((\mathrm{T_m} - \mathrm{E_m}) - \mathrm{F_m})\mathrm{x} = \mathrm{b} \\ \Leftrightarrow\ & (\mathrm{T_m} - \mathrm{E_m})\mathrm{x} - \mathrm{F_m x} = \mathrm{b} \\ \Leftrightarrow\ & (\mathrm{T_m} - \mathrm{E_m})\mathrm{x} = \mathrm{F_m x} + \mathrm{b} \end{aligned}$$

$$\Leftrightarrow x = (T_m - E_m)^{-1}F_m x + (T_m - E_m)^{-1}b$$

$$x^{k+1} = (T_m - E_m)^{-1}F_m x^k + (T_m - E_m)^{-1}b \quad (10)$$

where $(T_m - E_m)^{-1}F_m$ and $(T_m - E_m)^{-1}b$ are an iteration matrix and a column vector for Generalized Gauss-Seidel respectively.

4.2.3 Generalization of Successive over Relaxation Method

We can multiply the original equation (1) by a fixed parameter $\boldsymbol{\omega}$ to form the following equation:

$\omega Ax = \omega b$. After the decomposition of the left hand side term, we have

$$\omega A = \omega T_m - \omega E_m - \omega F_m$$

By back substitution, it becomes

$$(\omega T_m - \omega E_m - \omega F_m)x = \omega b$$

$$(T_m + \omega T_m - T_m - \omega E_m - \omega F_m)x = \omega b$$

$$\Leftrightarrow (T_m - \omega E_m + \omega T_m - T_m - \omega F_m)x = \omega b$$

$$\Leftrightarrow ((T_m - \omega E_m) - (T_m - \omega T_m + \omega F_m))x = \omega b$$

$$\Leftrightarrow ((T_m - \omega E_m) - ((1-\omega)T_m + \omega F_m))x = \omega b$$

$$\Leftrightarrow (T_m - \omega E_m)x - ((1-\omega)T_m + \omega F_m)x = \omega b$$

$$\Leftrightarrow (T_m - \omega E_m)x = ((1-\omega)T_m + \omega F_m)x + \omega b$$

$$\Leftrightarrow x = (T_m - \omega E_m)^{-1}((1-\omega)T_m + \omega F_m)x + \omega(T_m - \omega E_m)^{-1}b$$

$$x^{k+1} = (T_m - \omega E_m)^{-1}((1-\omega)T_m + \omega F_m)x^k + \omega(T_m - \omega E_m)^{-1}b \quad (11)$$

Where $(T_m - \omega E_m)^{-1}((1-\omega)T_m + \omega F_m)$ and $\omega(T_m - \omega E_m)^{-1}b$ are an iteration matrix and a column vector of the generalized successive over relaxation method respectively.

Theorem 4.1 Let A be strictly diagonally dominant (SDD) matrix. Then for any natural number $m \leq n$ the generalized Jacobi and generalized Gauss-Seidel methods are convergent for any initial guess x_0.

Proof: see [4]

Theorem 4.2 Let A and T_m be SPD matrices. Then for every $0 < \omega < 2$, the GESOR method converges with any initial guess x_0.

Proof: see [5].

4.3 Results

In order to compare the efficiency of the three methods, we were considering the three examples

Example 4.1: consider the 4x4 system given by:

$$
\begin{aligned}
&4x_1 + x_2 - x_3 = 7\\
&x_1 + 3x_2 - x_3 = 8\\
&-x_1 - x_2 + 5x_3 + 2x_4 = -4\\
&2x_3 + 4x_4 = 6
\end{aligned}
$$

Example 4.2 consider the 6x6 system given by:

$$
\begin{aligned}
&4x_1 - x_2 - x_4 = 1\\
&-x_1 + 4x_2 - x_3 - x_5 = 0\\
&-x_2 + 4x_3 - x_6 = 0\\
&-x_1 + 4x_4 - x_5 = 0\\
&-x_2 - x_4 + 4x_5 - x_6 = 0\\
&-x_3 - x_5 + 4x_6 = 0
\end{aligned}
$$

Example 4.3 :Consider the 20x20 system given by:

$$8x_1 - x_3 - x_7 = 0$$
$$8x_2 - x_4 - x_8 = -1$$
$$-x_1 + 4x_3 - x_5 - x_9 = 1$$
$$-x_2 + 4x_4 - x_6 - x_{10} = 0$$
$$-x_3 + 4x_5 - x_7 - x_{11} = -1$$
$$-x_4 + 8x_6 - x_8 - x_{12} = 0$$
$$-x_1 - x_5 + 8x_7 - x_9 - x_{13} = -1$$
$$-x_2 - x_6 + 8x_8 - x_{10} - x_{14} = 1$$
$$-x_3 - x_7 + 10x_9 - x_{11} = -1$$
$$-x_4 - x_8 + 5x_{10} - x_{12} - x_{16} = 0$$
$$-x_5 - x_9 + 5x_{11} - x_{13} - x_{17} = 1$$
$$-x_6 - x_{10} + 10x_{12} - x_{14} - x_{18} = 0$$
$$-x_7 - x_{11} + 8x_{13} - x_{15} - x_{19} = -1$$
$$-x_8 - x_{12} + 8x_{14} - x_{16} - x_{20} = 0$$
$$-x_9 - x_{13} + 8x_{15} - x_{17} = 1$$
$$-x_{10} - x_{14} + 4x_{16} - x_{18} = -1$$
$$-x_{11} - x_{15} + 4x_{17} - x_{19} = 0$$
$$-x_{12} - x_{16} + 4x_{18} - x_{20} = 1$$
$$-x_{13} - x_{17} + 8x_{19} = -1$$
$$-x_{14} - x_{18} + 8x_{20} = 0$$

Based on the above three examples, we have made analysis separately using tables below to compare the efficiency of the three generalized iterative methods that suits to find the solution of system of linear equations.

Table 4.1: Numerical solution of Example 4.1 for m=1 by GJ method.

K	X_1^k	X_2^k	X_3^k	X_4^k
0	0.0000000000000	0.00000000000000	0.0000000000000	0.00000000000000
1	1.3000000000000	1.80000000000000	-1.3000000000000	2.15000000000000
2	0.9100000000000	2.06000000000000	-0.9100000000000	1.95500000000000
3	1.0270000000000	1.98200000000000	-1.0270000000000	2.01350000000000
4	0.9919000000000	2.00540000000000	-0.9919000000000	1.99595000000000
5	1.0024300000000	1.99838000000000	-1.0024300000000	2.00121500000000
6	0.9992710000000	2.00048600000000	-0.9992710000000	1.99963550000000
7	1.0002187000000	1.99985420000000	-1.0002187000000	2.00010935000000
8	0.9999343900000	2.00004374000000	-0.9999343900000	1.99996719500000
9	1.0000196830000	1.99998687800000	-1.0000196830000	2.00000984150000
10	0.9999940951000	2.00000393660000	-0.9999940951000	1.99999704755000
11	1.0000017714700	1.99999881902000	-1.0000017714700	2.00000088573500
12	0.9999994685590	2.00000035429400	-0.9999994685590	1.99999973427950
Exact Solution	1.0000000000000	2.00000000000000	-1.0000000000000	2.00000000000000
Error	0.0000005314410	0.00000035429400	0.00000053144100	0.00000026572050

Number of iteration=12 **Elapsed Time is 0.002405 seconds**

Table 4.2: Numerical solution of Example 4.1 for m=1 by GGS method.

K	X_1^k	X_2^k	X_3^k	X_4^k
0	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
1	1.268292682926830	1.926829268292683	-0.95121951219512	1.975609756097561
2	1.013087447947650	1.996430696014278	-0.99762046400908	1.998810232004759
3	1.000638412095007	1.999825887610453	-0.99988392507363	1.999941962536818
4	1.000031142053415	1.999991506712705	-0.99999433780847	1.999997168904235
5	1.000001519124557	1.999999585693303	-0.99999972379553	1.999999861897768
6	1.000000074103637	1.999999979789918	-0.99999998652661	1.999999993263306
Exact Solution	1.000000000000000	2.000000000000000	-1.00000000000000	2.000000000000000
Error	0.000000074103637	0.000000020210082	0.000000013473388	0.000000006736694

Number of iteration=6 **Elapsed Time is 0.001748seconds**

Table 4.3: Numerical solution of Example 4.1 for m=1 and $\omega = 1.01$ by GESOR method.

K	X_1^k	X_2^k	X_3^k	X_4^k
0	**0**	**0**	**0**	**0**
1	1.280663252865155	1.947346988539381	-0.9572957815167	1.99364789075835
2	1.008831469548973	1.997105382472240	-0.9990523830343	1.99937619151715
3	1.000170578801981	1.999950465320743	-0.9999860252357	1.99999451261789
4	1.000002122119229	1.999999298229650	-0.9999999031918	1.99999993659591
5	1.000000005527705	1.999999997798376	-1.0000000018771	2.00000000108858
Exact Solution	1.000000000000000	2.000000000000000	-1.0000000000000	2.00000000000000
Error	0.000000005527705	0.000000002201624	0.00000000187716	0.00000000108858

Number of iteration=5 **Elapsed time is 0.001556 seconds**

Table 4.4: Numerical solution of Example 4.2 for m=1 by GJ method.

k=13	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	0.294823825778	0.29482401656	0.00000019079		
x_2^k	0.09316743276	0.09316770186	0.0000002691		
x_3^k	0.02815716010	0.02815734989	0.00000018979	0.003542 Seconds	13
x_4^k	0.08612787035	0.08612836439	0.00000049404		
x_5^k	0.04968874517	0.04968944099	0.00000069583		
x_6^k	0.01946120766	0.01946169772	0.00000049007		

Table 4.5: Numerical solution of Example 4.2 for m=1 by GGS method.

K=7	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	0.29482382577	0.294824016563	0.0000001908		
x_2^k	0.09316743276	0.093167701863	0.0000002691		
x_3^k	0.02815716010	0.028157349896	0.0000001898	0.001660 Seconds	7
x_4^k	0.08612829068	0.086128364389	0.0000000737		
x_5^k	0.04968933692	0.049689440994	0.0000001041		
x_6^k	0.01946162426	0.019461697723	0.0000000734		

Table 4.6: Numerical solution of Example 4.2 for m=1 and $\omega = 1.05$ by GESOR method·

K=5	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	0.29482413940709	0.294824016563147	0.000000122843949		
x_2^k	0.09316780786102	0.093167701863354	0.000000105997669		
x_3^k	0.02815738885926	0.028157349896480	0.000000038962781	0.001369 seconds	5
x_4^k	0.08612838704386	0.086128364389234	0.000000022654633		
x_5^k	0.04968946667164	0.049689440993789	0.000000025677854		
x_6^k	0.01946169962483	0.019461697722567	0.000000001902270		

Table 4.7: Numerical solution of Example 4.3 for m=1 by GJ method.

K=20	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	-0.00133333011505	-0.00133364508583	0.00000031497078	0.009624 seconds	20
x_2^k	-0.11532115886985	-0.11532109044523	0.00000006842462		
x_3^k	0.17111888081877	0.17111900750313	0.00000012668436		
x_4^k	-0.03118181450736	-0.03118201312238	0.00000019861502		
x_5^k	-0.2244205597255	-0.22442166064402	0.00000110091845		
x_6^k	0.01210809919268	0.01210819483583	0.00000009564315		
x_7^k	-0.18178824913392	-0.18178816818980	0.00000008094411		
x_8^k	0.10861342021945	0.10861328956049	0.00000013065896		
x_9^k	-0.08976822389022	-0.08976866425760	0.00000044036738		
x_{10}^k	-0.02151540340354	-0.02151515688014	0.00000024652339		
x_{11}^k	0.11298236287965	0.11298251811057	0.00000015523092		
x_{12}^k	0.01943441318971	0.01943428224860	0.00000013094111		
x_{13}^k	-0.13878085966321	-0.13878137553097	0.00000051586776		
x_{14}^k	-0.00636576065796	-0.00636563102650	0.00000012963145		
x_{15}^k	0.09866672194731	0.09866678139961	0.00000005945230		
x_{16}^k	-0.20444106200084	-0.20444134308744	0.00000028108660		
x_{17}^k	0.01788507713670	0.01788429098550	0.00000078615119		
x_{18}^k	0.21011521060115	0.21011541555686	0.00000020495570		
x_{19}^k	-0.14011217999367	-0.14011213556818	0.00000004442549		
x_{20}^k	0.02546880398538	0.02546872306629	0.00000008091909		

Table 4.8: Numerical solution of Example 4.3 for m=1 by GGS method.

K=11	Approximate solution	Exact solution	Error	CPU time	Iteration number
x_1^k	-0.00133310687998	-0.00133364508583	0.0000005382058		
x_2^k	-0.11532136080792	-0.11532109044523	0.000000270362691		
x_3^k	0.171120038756722	0.171119007503133	0.00000103125358		
x_4^k	-0.03118248665115	-0.03118201312238	0.000000473528769		
x_5^k	-0.22442078790579	-0.22442166064402	0.00000087273823		
x_6^k	0.012107998420790	0.012108194835839	0.000000196415049		
x_7^k	-0.18178782074355	-0.18178816818980	0.00000034744625		
x_8^k	0.108613053525004	0.108613289560494	0.000000236035490		
x_9^k	-0.08976838895831	-0.08976866425760	0.00000027529929		
x_{10}^k	-0.02151549280641	-0.02151515688014	0.000000335926270	0.004641 seconds	11
x_{11}^k	0.112982957798164	0.112982518110579	0.00000043968758		
x_{12}^k	0.019434158347163	0.019434282248605	0.000000123901442		
x_{13}^k	-0.13878121286431	-0.13878137553097	0.00000016266666		
x_{14}^k	-0.00636576105504	-0.00636563102650	0.000000130028535		
x_{15}^k	0.098666903888315	0.098666781399615	0.00000012248870		
x_{16}^k	-0.20444153705961	-0.20444134308744	0.000000193972163		
x_{17}^k	0.017884464354587	0.017884290985503	0.00000017336908		
x_{18}^k	0.210115314004927	0.210115415556863	0.000000101551936		
x_{19}^k	-0.14011209356371	-0.14011213556818	0.00000004200446		
x_{20}^k	0.025468694118736	0.025468723066295	0.000000028947559		

Table 4.9: Numerical solution of Example for m=1 and $\omega = 1.10$ by GESOR method.

K=9	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	-0.0013335292030	-0.0013336450858	0.000000115882736		
x_2^k	-0.1153212872656	-0.1153210904452	0.00000019682041		
x_3^k	0.17111924071816	0.17111900750313	0.00000023321502		
x_4^k	-0.0311820265037	-0.0311820131223	0.00000001338134		
x_5^k	-0.2244216855862	-0.2244216606440	0.00000002494219		
x_6^k	0.01210818674904	0.01210819483583	0.00000000808679	0.003115 seconds	9
x_7^k	-0.1817881080576	-0.1817881681898	0.00000006013213		
x_8^k	0.10861315942108	0.10861328956049	0.00000013013940		
x_9^k	-0.0897686272108	-0.0897686642576	0.00000003704679		
x_{10}^k	-0.0215151576599	-0.0215151568801	0.00000000077977		
x_{11}^k	0.11298251882914	0.11298251811057	0.00000000071856		
x_{12}^k	0.01943426176103	0.01943428224860	0.00000002048757		
x_{13}^k	-0.1387813659355	-0.1387813755309	0.00000000959542		
x_{14}^k	-0.0063656720146	-0.0063656310265	0.00000004098817		
x_{15}^k	0.09866679015170	0.09866678139961	0.00000000875209		
x_{16}^k	-0.2044413264383	-0.2044413430874	0.00000001664907		
x_{17}^k	0.01788429192279	0.01788429098550	0.00000000093729		
x_{18}^k	0.21011540580160	0.21011541555686	0.00000000975526		
x_{19}^k	-0.1401121348467	-0.1401121355681	0.00000000072139		
x_{20}^k	0.02546871832848	0.02546872306629	0.00000000473781		

Table 4.10: The consolidation of the above result for each iterative method can be summarized as follows:

Methods	Order of matrices					
	Example 4.1		Example 4.2		Example 4.3	
	Number of iteration	CPU time (seconds)	Number of iteration	CPU time (seconds)	Number of iteration	CPU time (seconds)
GJ	12	0.002405	13	0.003542	20	0.009624
GGS	6	0.001748	7	0.001660	11	0.004641
GESOR	5	0.001556	5	0.001369	9	0.003115

Table 4.11: As varying ω for Generalized Successive over Relaxation method

Parameter (ω)	Example 4.1		Example 4.2		Example 4.3	
	Number of iteration	CPU time (sec)	Number of iteration	CPU time (sec)	Number of iteration	CPU time (sec)
1.01	5	0.01213	7	0.00717	11	0.0102
1.05	10	0.01306	5	0.01291	10	0.0136
1.1	8	0.00728	6	0.01239	9	0.0241
1.25	11	0.00580	9	0.00842	12	0.0146
1.5	22	0.00651	18	0.00708	20	0.0106
1.9	143	0.03871	110	0.01954	120	0.0524

In the above tables we took m=1 for the generalized iterative methods while in the next tables we have considered m=2 for the first and the third examples to compare the generalized iterative methods.

Table 4.12: Numerical solution of Example 4.1for m=2 by GJ method.

K=20	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	0.9999997914591	1.0000000000000	0.000000208540		
x_2^k	2.000000900410	2.000000000000	0.00000090041	0.011943 seconds	20
x_3^k	-0.999997650535	-1.000000000000	0.00000234946		
x_4^k	1.999998759486	2.000000000000	0.000001240513		

Table 4.13: Numerical solution of Example 4.1 for m=2 by GGS method.

K=11	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	0.999999598820704	1.00000000000000	0.0000004011792		
x_2^k	2.000000265837072	2.00000000000000	0.000000265837	0.002185 seconds	11
x_3^k	-0.99999994780206	-1.0000000000000	0.000000052197		
x_4^k	1.99999997390103	2.00000000000000	0.000000026098		

Table 4.14: Numerical solution of Example 4.1 for m=2 and w=1.05 by GESOR method.

K=10	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	1.0000002116546	1.000000000	0.000000211654		
x_2^k	2.0000000468825	2.000000000	0.000000046882	0.001827 seconds	10
x_3^k	-0.999999876757	-1.00000000	0.000000123242		
x_4^k	1.9999999420243	2.000000000	0.000000057975		

Table 4.15: Numerical solution of Example 4.3 for m=2 by GJ method.

K=12	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	-0.0006862618400	-0.000686315734	0.0000000538946		
x_2^k	-0.1153209800858	-0.115321090445	0.0000001103593		
x_3^k	0.1745024793015	0.1745021816595	0.0000002976420		
x_4^k	-0.031181567382	-0.031182013122	0.0000004457400		
x_5^k	-0.222379923148	-0.222380528546	0.0000006053979		
x_6^k	0.0121083093752	0.0121081948358	0.0000001145393		
x_7^k	-0.179992538592	-0.179992707536	0.0000001689443		
x_8^k	0.10861333261270	0.1086132895604	0.00000004305221		
x_9^k	-0.0789243949337	-0.0789244290809	0.00000003414719		
x_{10}^k	-0.0215151030750	-0.0215151568801	0.00000005380508		
x_{11}^k	0.11596841267469	0.11596841169204	0.00000000098265		
x_{12}^k	0.01943432353755	0.01943428224860	0.00000004128894	0.023709 seconds	12
x_{13}^k	-0.1379503503902	-0.1379503869322	0.00000003654196		
x_{14}^k	-0.00636549282870	-0.00636563102650	0.000000138197804		
x_{15}^k	0.100277964148688	0.100277823375814	0.000000140772874		
x_{16}^k	-0.20444085911538	-0.20444134308744	0.000000483972065		
x_{17}^k	0.019097961385033	0.019097403019697	0.000000558365336		
x_{18}^k	0.210115655185792	0.210115415556863	0.000000239628930		
x_{19}^k	-0.13985648530596	-0.13985662298906	0.000000137683100		
x_{20}^k	0.025468770087164	0.025468723066295	0.000000047020870		

Table 4.16: Numerical solution of Example 4.3 for m=2 by GGS method.

K=7	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	-0.0013336033244	-0.001333645085	0.0000000417613		
x_2^k	-0.1153209919656	-0.115321090445	0.0000000984795		
x_3^k	0.17111907732570	0.17111900750313	0.00000006982257		
x_4^k	-0.0311816827500	-0.0311820131223	0.00000033037230		
x_5^k	-0.2244215388420	-0.2244216606440	0.00000012180198		
x_6^k	0.01210828468823	0.01210819483583	0.00000008985239		
x_7^k	-0.1817881318420	-0.1817881681898	0.00000003634772		
x_8^k	0.10861335655591	0.10861328956049	0.00000006699542		
x_9^k	-0.0897686483193	-0.0897686642576	0.00000001593823		
x_{10}^k	-0.0215149929446	-0.0215151568801	0.00000016393552		
x_{11}^k	0.11298257132265	0.11298251811057	0.00000005321207	0.030295 Seconds	7
x_{12}^k	0.01943432888820	0.01943428224860	0.00000004663960		
x_{13}^k	-0.1387813600971	-0.1387813755309	0.00000001543386		
x_{14}^k	-0.0063656042465	-0.0063656310265	0.00000002677999		
x_{15}^k	0.09866678728988	0.09866678139961	0.00000000589027		
x_{16}^k	-0.2044412886729	-0.2044413430874	0.00000005441446		
x_{17}^k	0.01788430673558	0.01788429098550	0.00000001575008		
x_{18}^k	0.21011544249920	0.21011541555686	0.00000002694234		
x_{19}^k	-0.1401121316701	-0.1401121355681	0.00000000389799		
x_{20}^k	0.025468729781587	0.025468723066295	0.000000006715292		

Table 4.17: Numerical solution of Example 4.3 for m=2 and w=1.03 by GESOR method.

K=6	Approximate solution	Exact solution	Error	CPU time	Number of iteration
x_1^k	-0.0006863075062	-0.0006863157346	0.00000000822839		
x_2^k	-0.1153210326422	-0.1153210904452	0.00000005780301		
x_3^k	0.17450222042481	0.17450218165952	0.00000003876528		
x_4^k	-0.0311817956431	-0.0311820131223	0.00000021747925		
x_5^k	-0.2223804860301	-0.2223805285463	0.00000004251619		
x_6^k	0.01210825303437	0.01210819483583	0.00000005819853		
x_7^k	-0.1799926979330	-0.1799927075367	0.00000000960368		
x_8^k	0.10861332560843	0.10861328956049	0.00000003604793		
x_9^k	-0.0789244219632	-0.0789244290809	0.00000000711769		
x_{10}^k	-0.0215150754251	-0.0215151568801	0.00000008145496		
x_{11}^k	0.11596842695200	0.11596841169204	0.00000001525996		
x_{12}^k	0.0194343057474	0.0194342822486	0.00000002349886		
x_{13}^k	-0.1379503831359	-0.1379503869322	0.00000000379629	0.014545 seconds	6
x_{14}^k	-0.00636561889635	-0.00636563102650	0.000000012130142		
x_{15}^k	0.100277824726188	0.100277823375814	0.000000001350374		
x_{16}^k	-0.20444132201148	-0.20444134308744	0.000000021075960		
x_{17}^k	0.019097406575542	0.019097403019697	0.000000003555845		
x_{18}^k	0.210115425720413	0.210115415556863	0.000000010163550		
x_{19}^k	-0.13985662220482	-0.13985662298906	0.000000000784247		
x_{20}^k	0.025468725531536	0.025468723066295	0.000000002465242		

4.4 Discussion

As Ibrahim B. Kalambi[6] result shows that the Successive Over-Relaxation method is more efficient than the other two iterative methods(Jacobi and Gauss-Seidel), considering their performance, using parameters as time to converge, number of iterations required to converge, storage and level of accuracy.

According to Davod Khostajeh Salkuyeh[4 and 5] illustrated in his work the generalized Jacobi, generalized Gauss-Seidel and generalized Successive Over-Relaxation method are more efficient than the conventional Jacobi, Gauss-Seidel and Successive Over-Relaxation methods respectively for solving the solution of system of linear equations.

In this research, the researcher have obtained the result based on three practical examples to compare the efficiency of the three generalized iterative methods taking into account their performance, number of iteration required to converge, computational running time it takes to converge(in seconds) and level of accuracy having the same tolerance factor. The results obtained in the Tables above have been discussed as follows: in Table 4.1 generalized Jacobi has registered the number of iteration required (12) and time it takes (0.002405) for 4x4 system of linear equation considered. In Table 4.2 generalized Gauss-Seidel has registered the number of iteration required (6) and time it takes to converge (0.001748) for 4x4 system of linear equation. In Table 4.3 generalized Successive over Relaxation has registered the number of iteration required (5) and time it takes to converge (0.001556) for 4x4 system of linear equation. In Table 4.4 generalized Jacobi has registered the number of iteration required (13) and time it takes to converge (0.003542) for 6x6 system of linear equation. In Table 4.5 generalized Gauss-Seidel has registered the number of iteration required (7) and time it takes to converge (0.001660) for 6x6 system of linear equation. In Table 4.6 generalized Successive over Relaxation has registered the number of iteration required (5) and time it takes to converge (0.001369) for 6x6 system of linear equation. In Table 4.7 generalized Jacobi has registered the number of iteration required (20) and time it takes to converge (0.009624) for 20x20 system of linear equation. In Table 4.8 generalized Gauss-Seidel has registered the number of iteration required (11) and time it takes to converge (0.004641) for 20x20 system of linear equation. In Table 4.9 generalized Successive over Relaxation has registered the number of iteration required (9) and time it takes to converge (0.003115) for 20x20 system of linear equation. If we observe table 4.1, table 4.2 and table 4.3 for m=1 and if we compare these with table 4.12, table 4.13 and table 4.14 for m=2 the result shows the efficiency of the generalized iterative method is good when m=1. On the

other hand table 4.7, table 4.8 and table 4.9 which is the approximation value of the generalized iterative methods for m=1 and if we compare respectively with the results of table 4.15, table 4.16 and table 4.17 which is for the value of m=2 it shows that the efficiency of the generalized iterative method is more efficient when m=2.

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

We have presented the three generalized iterative methods for solving system of linear equations and these are generalized Jacobi method, generalized Gauss-Seidel method and generalized Successive over Relaxation method. Three model examples are considered: 4x4, 6x6 and 20x20 systems of linear equation. The study was treated the three methods in different perspective. The first mechanism to compare their efficiency is in terms of the iteration number required to converge. The second way of comparing the three methods are based on the error obtained from the difference between exact solution and approximate solution having in mind pre-specified tolerance ($5x10^{-6}$) and finally the three methods have been compared in computational running time while calculating the given system of linear equations. The number of iteration required and the time it takes to converge for the generalized Jacobi method, the generalized Gauss-Seidel method and the generalized Successive over Relaxation method respectively are 12 and 0.002405 seconds, 6 and 0.001748seconds and 5 and 0.001556seconds for order of 4x4, 13 and 0.003542seconds, 7 and 0.001660seconds and 5 and 0.001369seconds for order of 6x6, and 20 and 0.009624seconds, 11 and 0.004641seconds and 9 and 0.003115seconds for order of 20x20. The obtained results shows that the generalized successive over relaxation method is more efficient than generalized Gauss-Seidel method and is much more efficient than the generalized Jacobi method with their performance, number of iteration required, computational running time and level of accuracy should be taken into account for each examples.

5.2 Future Works

The results under Table 4.11 show that for different value of ω GESOR gives the approximation at different iteration steps, different computational time. In the future work it is recommendable to calculate its optimal parameter.

References

[1] A. J. Hughes Hallett: *The Convergence of Accelerated Over relaxation iterations*. Mathematics of Computation vol.47 no.175 July 1986 pages 219-223.

[2] A. Hadjidimos and A. Yeyios : *On some Extensions of the Accelerated Over relaxation (AOR) Theory*. Internat .J. math. And math. Scie. Vol.5 no.1 (1982) 49-60.

[3] Anamul Haque Laskar and Samira Behera : *Refinement of iterative methods for the solution of system of equation Ax=b*; IOSR Journal of Mathematics (IOSR-JM).

[4] Davod K. Salkuyeh, *Generalized Jacobi and Gauss-Seidel Methods for Solving Linear Systems of Equations, Numer. Math. J. Chinese Uni (Englisher) issue* 2, Vol.16: 164-170, 2007.

[5] D. Khojasteh Salkuyeh : A Generalization of the SOR method for solving linear system of equations. Journal of Applied mathematics Islamic Azad University of Lahijan; vol.4 no. 15, winter 2007.

[6] Ibrahim B. Kalambi,et al *A Comparison of Three Iterative Methods for the Solution of Linear Equations, J. Appl. Sci. Environ*. Vol. 12, 2008, no.4, 53-55.

[7] Jae Heon Yun and Sang Wook kim : Generalized Stationary Iterative method for solving linear systems. Koean J. Comput. And appl. Math. Vol.5 (1998) no.2,pp 341-349.

[8] Kalambi I. *Solutions of simultaneous equations by iterative methods*. Postgraduate Diploma in computer science project. Abubakar tafawa Balewa University. Bauchi 1998.

[9] Kumer Vatti V.B. et al and Genanew Gofe Gonfa, *Refinement of Generalized Jacobi (RGJ) Method for Solving System of Linear Equations. Int. J. Contemp. Math. Sciences*, Vol.6, 2011, no.3,109116.

[10] Lazim Abdullah and Nurhakimah Ab Rahman: *An investigation of the relationship between extrapolation parameters and approximate solutions in solving fuzzy linear systems using refinement of Jacobi Over Relaxation method*. Journal of physics conference series 435(2013) 012038.

[11] M. T. Darvishi *, P. Hessari: *On Convergence of the Generalized AOR method for linear systems with diagonally dominant coefficient matrices*. Applied Mathematics and computation 176(2006) 128-133.

[12] Noreen Jamil, *A Comparison of Direct and Indirect Solvers for Linear Systems of equations*. Int. J. Emerg. Scie. 2(2), 310-321, June 2012.

[13] National science Foundation, Division of mathematical science, program description PD 06-888

Computational mathematics 2006.

[14] P.G. Fisher: *Iterative techniques for solving simultaneous equation systems*: a view from the economics literature. Journal of computational and Applied Mathematics 24(1988) 241-255 North –Holland.

[15] Richard Barrett[2], et al: *Templates for the solution of linear systems*: Building Blocks for iterative methods.

[16] Richard L.Burden and J. Douglas Faires (2011): *Numerical Analysis*. Ninth Edition Brooks/cole, United States, United Kingdom

[17]Steeb W.H., Hardy Y., Hardy A. and Stoop R. 2004. *Problems and solutions in scientific computing with C++ and Java simulations, world scientific publishing*. ISBN 981-256-112-9

[18] Steven T.Karris (2004), *Numerical analysis using MATLAB and spreadsheets* 2nd Edition.

[19] Timothy. A. Davis: Fundamentals of algorithms.

[20] W. Kahan: Gauss-Seidel methods of solving large systems of linear equations. PhD Thesis, University of Toronto, 1958.

[21] Yousef Saad* Henk A.Vander Vorst+: Iterative solution of linear systems in the 20th century.

[22] Yousef Saad (2000): *Iterative methods for sparse linear systems* 2nd Edition with corrections.