NUMERICAL SOLUTION TO SYSTEM OF LINEAR EQUATIONS: A PASCAL PROGRAMMING APPROACH

BY

BASHIR LATEEF EKUNDAYO
PGD/MCS/99/2000/936

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS/COMPUTER SCIENCE, IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR AWARD OF A POST GRADUATE DIPLOMA IN COMPUTER SCIENCE, FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGER STATE, NIGERIA.

SEPTEMBER 2001

CERTIFICATION

This is to certify that this project work was carried out by **BASHIR LATEEF EKUNDAYO (PGD/MCS/1999/2000/963)** in the department of Mathematics/Computer Sciences, School of Science and Science Education, Federal University of Technology, Minna, Niger State, Nigeria.

_____ _____

DR. Y. AIYESIMI Date

PROJECT SUPERVISOR

_____ _____

DR. S. A. REJU Date

HEAD OF DEPARTMENT

_____ _____

EXTERNAL EXAMINER Date

DEDICATION

To

- My Late Grandmother : Mrs Aduni
- Sisters and brothers

ACKNOWLEDGEMENT

Special thanks to Almighty God for given me this opportunity to complete this project successfully; and to my supervisor Dr. Yomi Aiyesimi for his accurate supervision throughout this project. I remain indebted to his constant guidance. May Almighty God bless him throughout his lifetime.

I wish to express my sincere gratitude to my Dad BASHIR AJIBOLA; he had been in the forefront of my Education, I pray he lives long to reap the fruit of his labour.

My appreciation goes to Mr. And Mrs Oke for their parental support given to me and my friends; may God regard them for their kindness, bless them and make them keep up their good deeds.

I am also grateful to my freidns, Alabi Olusola John, Beky Dawood, Ogunbiyi Tunde (GSM), Lekan ,Uncle Tajudeen (Computer) and the entire members of PGD computer 2000. May God let us reap the Good fruit of life.

Finally, for those who were helpful to me at one time or the other, I thank you all, God bless.

BASHIR LATEEF EKUNDAYO

ABSTRACT

The methods of solving systems of linear algebraic equations are divided into two: The first group comprises of the so-called exact or direct methods and the second is the numerical method. The exact method enables us to obtain solution of a linear system after a finite number of arithmetic operations have been performed: Among these are Crammer's rule, Gausian Elimination and sweep method. The Numerical methods involve carrying out series of iterations to obtain an approximate solution. This project uses iteration methods of Jacobi and Siedel to test the convergency rate of both Jacobi and siedel's method of solution; and number of iterations involves.

TABLE OF CONTENT

CHAPTER FOUR

CHAPTER ONE

1.0 GENERAL INTRODUCTION

The numerical method of liner algebra includes the numerical methods of solving systems of linear algebraic equations, matrix inversion, computing determinants, and findings the Eigen values and Eigen vectors of matrices.

Methods of solving systems of linear algebraic equations are sub-divided into two groups. The first group comprises so-called exact or direct methods that is, algorithms enabling us to obtain the solution of a system after a finite number of arithmetic operations. Among these are: Cramer's rule for finding the solution of a system with the aid of determinants, the Gauss Elimination and the sweep methods of solving systems of linear algebraic equations, in particular.

This project discusses the numerical solutions to system of linear equation using iterative methods. Emphasis is laid on gauss Jacobi and Gauss Siedel's method; and the efficiency of Siedel over Jacobi using Pascal programming.

1.1 AIMS

(i) To show how fast is it in convergence

(ii) The number of iteration involves

(iii) How accurate is the computed results

(iv) The efficiency of Gauss Siedel using Pascal Programming

1.2 SCOPES AND LIMITATION

The project discusses solutions to system of linear equations using iterative methods. Emphasis is on Gauss Siedel and Jacobi's methods of solution only.

1.3 DEFINITION OF TERMS

(a) Matrices- A matrix is defined as a rectangular array of numbers enclosed in a bracket.

(b) Square matrix – A matrix with equal number of rows and columns

(c) Singular matrix – a matrix is said to be singular if and only if the rows or columns of the matrix are linearly dependent. Also if its determinant is zero.

(d) Sparse matrix – A matrix is said to be sparse if most of its elements aij {i = 1,…,n , j = 1, 2…n} are zero

(e) Dense matrix – a matrix is said to be dense if most of its elements aij are non-zero.

(f) Symmetric matrix – a square matrix P defined by [Pij} of order m is called symmetric matrix if every i,j we have that [P ij] = [Pji]

(g) Truncation – Is the dropping of any digits to the right of the decimal point.

(h) Label – is a positive integer used to prefix a statement of instruction within a Pascal program.

(i) Constants – are objects whose values cannot change during the running of a program.

(j) Variables are objects whose value can change during program execution.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 SYSTEM OF LINEAR EQUATIONS

Any arbitrary system of m linear equations in n unknowns will be written as

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n &= b_2 \\ a_{31}x_1 + a_{32}x_2 + \ldots\ldots + a_{3n}x_n &= b_3 \\ \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots\ldots + a_{mn}x_n &= b_m \end{aligned}\right\} \qquad (1.0)$$

where $x_1, x_2, \ldots\ldots, x_n$ are the unknowns and the subscripted a's and b's denote constants.

The above system is a linear algebraic system of $m \times n$.

2.2 Solutions

To solve the system (1.0) above, it is first abbreviated to an augmented matrix of the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots\ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \ldots\ldots & a_{2n} & b_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & a_{m3} & \ldots\ldots & a_{mn} & b_n \end{bmatrix}. \qquad (1.1)$$

The method of solution of (1.1) depends on the density of the matrix form above.

If the matrix is dense, then we use direct method of solution, mostly favoured by Gaussian Elimination. If on the other hand, the matrix is sparse the appropriate method of solution is the iterative procedure.

2.2.1 ITERATIVE METHOD OF SOLUTION

Although Gaussian Elimination (or the text version of Gauss-Jordan elimination) is generally the method of choice for solving a linear system of n equations in n unknowns, there are other approaches to solving linear systems, called iterative or indirect methods, which are better in certain situations. These methods start with an initial approximation to a solution and the generate a succession of better and better approximations that tend toward an exact solution.

2.2.2 GAUSS-JACOBI OR METHOD OF SIMULTANEOUS DISPLACEMENT

This method applied to linear systems of m equations in n unknowns. Suppose that the system has exactly one solution and that the diagonal entries

$a_{11}, a_{22}, \ldots, a_{mn}$ are non-zero.

To start, we rewrite system (1.0) by solving the first equation for x_1 in terms of the remaining unknowns; solving the second for x_2 in terms of the remaining unknowns, and so on.

This yield

$$x_1^1 = \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^0 - a_{12}x_3^0 - \ldots\ldots - a_{1n}x_n^0\right)$$

$$x_2^1 = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^0 - a_{23}x_3^0 - \ldots\ldots - a_{2n}x_n^0\right) \qquad (1.2)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$x_n^1 = \frac{1}{amn}\left(b_{m1} - a_{m2}x_2^0 - a_{m3}x_3^0 - \ldots\ldots - a_{mn-1}x_{n-r}^0\right)$$

The above procedure gives the steps taken to obtain the first iteration. If an approximation to the solution of (1.0) is known, and these approximate values are substituted into the right hand side of (1.2), it is often the case that the values of $x_1, x_2, \ldots x_n$ that result on the left hand side form an even better approximation to the solution. This is a key to the Jacobian method.

To obtain the second iteration, we substitute the various values of x_k obtained by the processes above into (1.0) and repeat the procedure.

Thus

$$x_1^2 = \frac{1}{a_{11}}\left[b_1 - a_{12}X_2^{(1)} + a_{13}X_3^{(1)} + \ldots + a_{1n}X_n^{(1)}\right]$$

$$x_2^{(2)} = \frac{1}{a_{22}}\left[b_1 - a_{11}X_1^{(1)} - \sum_{k=1}^{n} a_{2k}X_k^{(1)}\right]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$x_m^{(2)} = \frac{1}{a_{mn}}\left[b_m - \sum_{r=1}^{m-1} a_m X_r^{(1)} - \sum_{r=m+1}^{n} a_{mr}X_r^{(1)}\right]$$

In general, we have the r^{th} iteration

$$X_m^{(n)} = \frac{1}{a_{mn}}\left[b_m - \sum_{n=1}^{m-1} a_{mp} X_p^{(r-1)} - \sum_{p=m+1}^{n} a_{mp} X_p^{(r-1)}\right]$$

2.2.3 GAUSS-SIEDEL OR SUCCESSIVE DISPLACEMENT METHOD

In this method, a minor modification of the Jacobi method often reduces the number of iterations needed to obtain a given degree of accuracy.

If each iterations of the Jacobi, the new approximation is obtained by substituting the previous approximation into the right side of (1.2) and solving for new values of $x_1, x_2, \ldots$

i.e.

$$x_1^{(1)} = \frac{1}{a_{11}}\left[b_1 - \sum_{n=0}^{m=2} a_{1m} X_m^{(0)}\right]$$

$$x_2^{(1)} = \frac{1}{a_{22}}\left[b_2 - a_{21} X_1^{(1)} - \sum_{m=3}^{m=n} a_{2m} X_m^{(0)}\right]$$

$$x_3^{(1)} = \frac{1}{a_{33}}\left[b_3 - \sum_{m=1}^{m=2} a_{3m} X_n - \sum_{m=3}^{m=n} a_{3m} X_m^{(0)}\right]$$

.....

$$x_n^{(1)} = \frac{1}{a_{nm}}\left[b_n - \sum_{m=1}^{m=1} a_{nm} X_m^{(0)}\right]$$

also

$$x_1^{(2)} = \frac{1}{a_{11}}\left[b_1 - \sum_{k=2}^{m} a_{12} X_k^{(1)}\right]$$

$$x_2^{(2)} = \frac{1}{a_{22}}\left[b_2 - a_{21} X_1^{(2)} - \sum_{k=3}^{m} a_{2k} X_k^{(1)}\right]$$

$$x_3^{(2)} = \frac{1}{a_{33}}\left[b_3 - \sum_{k=1}^{m} a_{3k} X_k^{(2)} - \sum_{k=3}^{m} a_{3k} X_k^{(2)}\right]$$

.....

$$x_p^{(2)} = \frac{1}{a_{pp}}\left[b_p - \sum_{k=1}^{p-1} a_{pk} X_k^{(2)} - \sum_{k=p+1}^{m} a_{pk} X_k^{(1)}\right]$$

Hence, in general we have the Gauss-Siedel iteration as

$$x_r^{(s)} = \frac{1}{a_{rr}}\left[b_r - \sum_{m=1}^{r-1} a_{rm} X_m^{(s)} - \sum_{m=r+1}^{n} a_{rm} X_m^{(s-1)}\right]$$

These new values are not al computed simultaneously, first x_1, is obtained from the top equation, then x_2 is obtained from the second equation, then x_3, and so on. Since the new values are generally closer to the exact solution, this suggests that better accuracy might be obtained by using the new x values as soon as they are known.

$$x_1^{(1)} = \frac{1}{a_{11}}\left[b_1 - \sum_{n=0}^{n=2} a_{1m} X_m^{(0)}\right]$$

$$x_2^{(1)} = \frac{1}{a_{22}}\left[b_2 - a_{21} X_1^{(1)} - \sum_{m=3}^{m=n} a_{2m} X_m^{(0)}\right]$$

$$x_3^{(1)} = \frac{1}{a_{33}}\left[b_3 - \sum_{m=1}^{m=2} a_{3m} X_n - \sum_{m=3}^{m=n} a_{3m} X_m^{(0)}\right]$$

.....

$$x_n^{(1)} = \frac{1}{a_{mn}}\left[b_n - \sum_{m=1}^{n-1} a_{nm} X_m^{(0)}\right]$$

CHAPTER THREE

PASCAL

3.1 PASCAL PROGRAMMING LANGUAGE

Pascal programming language was invented in 1970 by Professor Niklaus Wirth of Zurich, Switzerland. It was named after the Seventeenth century French Mathematician who invented one of the earliest mechanical calculating machines.

It is a development of an earlier language; ALGOL (Algorithm Language) whose name implies that it is based on a more organized and mathematically oriented approach to programming than other languages. The mathematical aspect refers to the ideas relating to the proof of theorems rather than the mathematics of computations. The whole concept of the language is the structural approach to the solution of a problem. Such an approach not only makes it easier to write but also improves the clarity for an outsider who may have to take over the development of program from the original author.

Pascal program is built with lexical tokens that is, a program header and the body. The token are either language symbols or basic entities constructed by programmers.

Word symbols are reserved words with predefined meanings in Pascal programs and cannot be redefined by the user.

Identifiers are sequence of letters and digits beginning with a letter(s) used for naming the various objects defined.

Numbers are represented in decimal notation only and they denote integers or real values. A number with a decimal point and or letter 'e' is a real number; otherwise it is an integer number.

Labels are unsigned integers in the close interval from 0 to 9999.

They are used for prefixing statement, if necessary.

Strings are character whose value is denoted by a character sequence endorsed within quotes. However, if quotation marks must appear within a string, the marks are duplicates.

Comments are character sequence occurring outside character standard alternative representation for curly braces.

3.2 ARITHMETIC OPERATOR

Operators operate on one or two operand and perform a specific operation. The standard built in arithmetic operators in Pascal include the following:

(i) + Addition

(ii) - Subtraction

(iii) * Multiplication

(iv) / Division to obtain real number result

(v) DIV Division to yield a truncated integer result.

(vi) MOD Modulus to yield the remainder of a division

3.3 PUNCTUATION

The following punctuation rules are considered in Pascal Program.

(i) Semicolon: This terminates the program heading, declaration and each statement from the next.

(ii) Commas: They are used to separate items in lists such as the names in a real variable declaration or the items in a WRITE and READ statement.

(iii) Colon: This is used in several situations but in elementary program. The main function is to separate list of items in a declaration from the corresponding data type name.

(iv) Period: This is used to terminate programs and always appear after the last end.

(v) Procedure and Function: Procedures and functions are two kind of subprogram available in Pascal. They are like subroutine which when called upon within the Program; they will perform the task they are design to do. Hence, for a procedure or a function to be useful in a program, they need to have been declared in the procedure and function declaration part.

The difference between these two forms of subprogram in Pascal is that function operates to yield a single result while a procedure can yield more than one result.

(vi) Procedure: It allows more than one value to be returned into the main program during execution. Ion case where no value is returned to the main program, In/Out specific operations are performed like arranging a list of numbers in a particular order etc. The format for declaring a procedure in the procedure and function part takes the form:

PROCEDURE Name (I1,I2,I3:Real; VAR R1, R2:Integer);

After the name of the procedure, a list of formal parameters are enclosed in parenthesis. This list consists of two parts; one sublist is I_1, I^2,.. and the other is R_1, R_2,...

The parameters I_1, I_2, I_3, in the first sub list are called value parameters and the other parameters R_1, R_2, prefixed with VAR are called reference parameters. The Input value (arguments) to be processed by the procedure are copied into the value parameters. After the procedure has processes the inputs, the results or outputs are stored in the reference parameters.

However, a procedure having being declared in the appropriate declaration part can be used in the body of the program by typing the procedure name which may be followed by parameters (both value and reference or variable parameters). If the procedure is not meant for any computation, it can be called without starting any parameters along the name of the procedure.

(vii) Function: The use of function allows the introduction of programmer (users defined function as against the built-in functions, which are predefined within the compiler.

However, the format of these two forms of subprogram is the same. It commences with the name of subprogram followed by the parameter of this format is given as:

FUNCTION Power (Number, Index: Integer): Integer;

Where power represents the name of the function with which it will be called. Number and index represents local variable to be operated upon within the function.

The integer inside the parenthesis represents local variable types while the one outside the parenthesis represent the function type.

(viii) Array:

3.4 PROGRAM STRUCTURE

In Pascal, a program is composed of a program heading (its identification), that is followed by a block (definition and declarations of all objects, used in the program and statements). The program heading contains the identification details of the program. The format of this is given below as:

PROGRAM Identifier;

The identifier may be followed by parameter lists in parenthesis which represents the input and output devices to be used by the program.

The second section of a Pascal is made up of the declaration and definition and statement part. This section is further subdivided into six parts; the parts are listed in the required order as follow:

LABEL DECLARATION PART

CONSTANT DEFINITION PART

TYPE DEFINITION PART

VARIABLE DECLARATION PART

PROCEDURE AND FUNCTION

The above listed parts are discussed as follow:

Label declaration: This part consists of all labels defined in the block. It is used with

2nd Iteration

$$X_1^{(2)} = \frac{\left[16 + X_2^{(1)} - X_3^{(1)}\right]}{3}$$

$$= \frac{[16 + 1.6 - 0]}{3}$$

$$= 17.6/3$$

$$= 5.866667$$

$$X_2^{(2)} = \frac{\left[8 - X_1^{(1)} - 3X_3^{(1)}\right]}{5}$$

$$= \frac{[8 - 5.33333 - 3(0)]}{5}$$

$$= 0.533333$$

$$X_3^{(2)} = \frac{\left[4X_1^{(1)} + X_2^{(1)}\right]}{2}$$

$$X_3^{(0)} = \frac{[4 \times 5.333333 + 0.533333]}{2}$$

$$= 21.866653/2$$

$$= 10.933333$$

3rd iteration

$$X_1^{(3)} = \frac{16 + X_2^{(2)} - X_3^{(2)}}{3}$$

$$= \frac{[16 + 0.533333 - 10.933333]}{3}$$

$$= \frac{5.600003}{3}$$

$$= 1.866667$$

$$X_2^{(3)} = \frac{\left[8 - X_1^{(2)} - 3X_3^{(2)}\right]}{5}$$

$$= \frac{[8 - 5.866667 - 3 \times 10.93333]}{5}$$

$$= \frac{8 - 5.866667 - 32.79999}{5}$$

$$= \frac{[-30.666667]}{5}$$

$$= -6.133333$$

$$X_3^{(3)} = \frac{\left[4X_1^{(2)} + X_2^{(2)}\right]}{2}$$

$$= \frac{4 \times 5.866667 + 0.533333}{2}$$

$$= 24.000001/2$$

$$= 12.000001$$

4th iteration

$$X_1^{(4)} = \frac{16 + X_{21}^{(3)} - X_3^{(3)}}{3}$$

$$= \frac{[16 - 6.133333 - 12.000001]}{3}$$

$$= \frac{[9.866667 - 12.000001]}{3}$$

$$= -2.133334/3$$

$$= -0.711111$$

$$X_2^{(4)} = \frac{\left[8 - X_1^{(3)} - 3X_3^{(3)}\right]}{5}$$

$$= \frac{8 - 1.866667 - 3 \times 12.000001}{5}$$

$$= \frac{6.133333 - 36.000003}{5}$$

$$= -29.86667/5$$

$$= -5.973334$$

$$X_3^{(4)} = \frac{\left[4X_1^{(3)} + X_2^{(3)}\right]}{2}$$

$$= \frac{[4 \times 1.866667 - 6.133333]}{2}$$

$$= \frac{[7.466668 - 6.133333]}{2}$$

$$= \frac{[1.333335]}{2}$$

$$= 0.666668$$

5th iterations

$$X_1^{(5)} = \frac{\left[16 + X_2^{(4)} - X_3^{(4)}\right]}{3}$$

$$= \frac{[16 - 5.973334 - 0.666668]}{3}$$

$$= \frac{[16 - 6.640002]}{3}$$

$$= \frac{[9.359998]}{3}$$

$$= 3.119999$$

$$X_1^{(5)} = \frac{\left[8 - X_1^{(4)} - 3X_3^{(4)}\right]}{5}$$

$$= \frac{[8 - 0.711111 - 3 \times 0.666668]}{5}$$

$$= \frac{[8 - 2.711115]}{5}$$

$$= \frac{[5.288885]}{5}$$

$$= 1.057777$$

$$X_3^{(5)} = \frac{\left[4X_1^{(4)} + X_2^{(4)}\right]}{2}$$

$$= \frac{[4 \times -0.711111 - 5.973334]}{2}$$

$$= \frac{[-8.817778]}{2}$$

$$= -4.408889$$

Summary of Results

m	$X_1^{(m)}$	$X_2^{(m)}$	$X_3^{(m)}$
0	0	0	0
1	5.33333	1.60000	0.00000
2	5.866667	0.533333	10.933333
3	1.866667	-6.133333	12.000001
4	-0.711111	-5.973334	0.666668
5	3.119999	1.057777	-4.408889

4.2 GAUSS SIEDEL'S SOLUTION

Compute the approximate solution of the following linear algebraic system using Gauss Siedel's method

i.e. $3x_1 - x_2 + x_3 = 16$

$x_1 + 5x_2 + 3x_3 = 8$

$4x_1 + x_2 - 2x_3 = 0$

Using Gauss Siedel's method of solution

Algorithm: For the m^{th} iteration, the solution for X_r is given as

$$X_r^m = \left[b_r - \sum_{s=1}^{n} a_{rs} X_s^{(m)} - \sum_{s=r+1}^{n} a_{rs} X_s^{(m-1)} \right] \Big/ a_{rr}$$

Taking the initial solution as $X^0 = (0, 0, 0)$

1st iteration

$$X_1^{(1)} = \frac{\left[16 + X_2^{(0)} - X_3^{(0)}\right]}{3}$$

$$= 5.333333$$

$$X_2^{(1)} = \frac{\left[8 - X_1^{(1)} - 3X_3^{(0)}\right]}{5}$$

$$= \frac{8 - 5.333333 - 0}{5}$$

$$= \frac{2.666667}{5}$$

$$= 0.5333334$$

$$X_3^{(0)} = \frac{\left[4X_1^{(1)} + X_2^{(1)}\right]}{2}$$

$$= \frac{[4 \times 5.333333 + 0.5333334]}{2}$$

$$= \frac{21.333332 + 0.5333334}{2}$$

$$= 10.9333327$$

2nd iteration

$$X_1^{(2)} = \frac{[16 + X_2^{(1)} - X_3^{(1)}]}{3}$$

$$= \frac{[16 + 0.5333334 - 10.933333]}{3}$$

$$= \frac{5.6000004}{3}$$

$$= 1.8666668$$

$$X_2^{(2)} = \frac{[8 - X_1^{(2)} - 3X_3^{(1)}]}{5}$$

$$= \frac{[8 - 1.8666668 - 3 \times 10.933333]}{5}$$

$$= -\frac{26.666666}{5}$$

$$= -5.3333332$$

$$X_3^{(2)} = \frac{[4X_1^{(2)} + X_2^{(2)}]}{2}$$

$$X_3^{(0)} = \frac{[4 \times 1.8666668 - 5.3333332]}{2}$$

$$= 2.1444440 / 2$$

$$= 1.072222$$

3rd iterations

$$X_1^{(3)} = \frac{\left[16 + X_2^{(2)} - X_3^{(2)}\right]}{3}$$

$$= \frac{[16 - 5.3333332 - 1.072222]}{3}$$

$$= \frac{9.5944448}{3}$$

$$= 3.1781483$$

$$X_2^{(3)} = \frac{\left[8 - X_1^{(3)} - 3X_3^{(2)}\right]}{5}$$

$$= \frac{[8 - 3.1781483 - 3 \times 1.072222]}{5}$$

$$= \frac{8 - 6.3948113}{5}$$

$$= \frac{1.6051887}{5}$$

$$= 0.32103772$$

$$X_3^{(3)} = \frac{\left[4X_1^{(3)} + X_2^{(3)}\right]}{2}$$

$$= \frac{4 \times 3.1781483 + 0.32103772}{2}$$

$$= \frac{13.03361092}{2}$$

$$= 6.51680546$$

4th Iteration

$$X_1^{(4)} = \frac{\left[16 + X_2^{(3)} - X_3^{(3)}\right]}{3}$$

$$= \frac{[16 + 0 - 6.51680546]}{3}$$

$$= \frac{9.48319454}{3}$$

$$= 3.16106485$$

$$X_2^{(4)} = \frac{\left[8 - X_1^{(4)} - 3X_3^{(3)}\right]}{5}$$

$$= \frac{[8 - 3.12773151 - 3 \times 6.51680546]}{5}$$

$$= -\frac{14.67814789}{5}$$

$$= -2.935629572$$

$$X_3^{(4)} = \frac{\left[4X_1^{(4)} + X_2^{(4)}\right]}{2}$$

$$= \frac{4 \times 3.12773151 - 2.935629572}{2}$$

$$= \frac{19.575297468}{2}$$

$$= 4.787648734$$

5th Iteration

$$X_1^{(5)} = \frac{\left[16 + X_2^{(4)} - X_3^{(4)}\right]}{3}$$

$$= \frac{[16 - 2.935629572 - 4.787648734]}{3}$$

$$= \frac{8.276721694}{3}$$

$$= 2.75890723$$

$$X_2^{(5)} = \frac{\left[8 - X_1^{(5)} - 3X_3^{(4)}\right]}{5}$$

$$= \frac{[8 - 2.75890723 - 3 \times 4.787648734]}{5}$$

$$= -\frac{9.121853332}{5}$$

$$= -1.8243706664$$

$$X_3^{(5)} = \frac{\left[4X_1^{(5)} + X_2^{(5)}\right]}{2}$$

$$= \frac{4 \times 2.75890723 - 1.8243706664}{2}$$

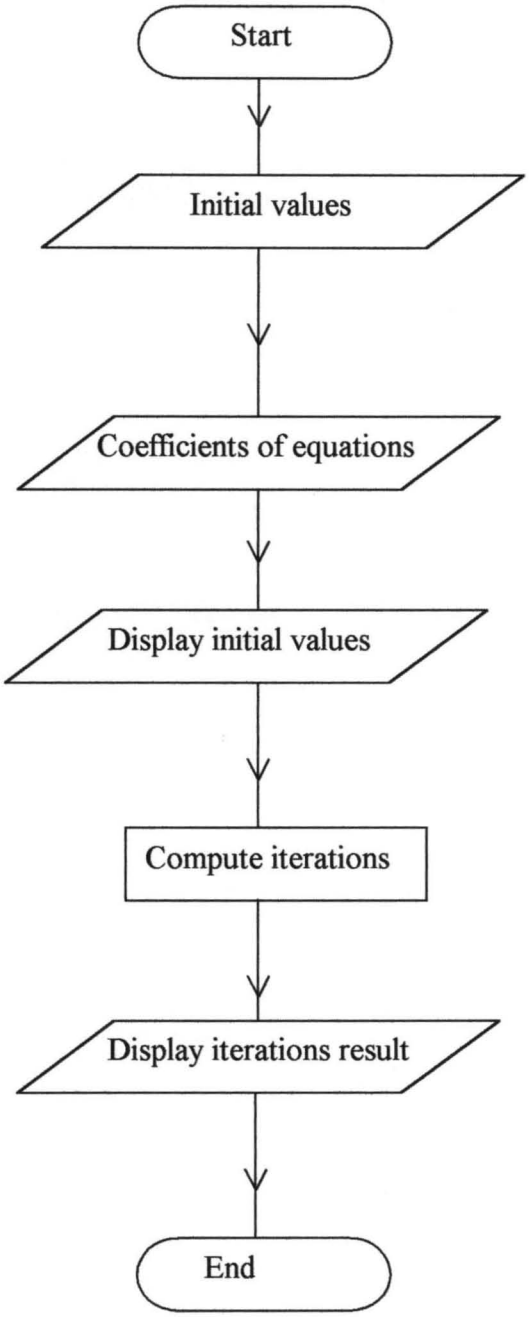
$$= \frac{9.2112582536}{2}$$

$$= 4.6056291268$$

Summary of Results

m	$X_1^{(m)}$	$X_2^{(m)}$	$X_3^{(m)}$
0	0.000000	0.000000	0.000000
1	5.333333	0.5333334	10.9335327
2	1.8666668	-5.3333332	1.072222
3	3.1781483	0.32103772	6.51680546
4	3.1277315	-2.935629572	4.787648734
5	2.75890723	-1.87243706664	4.6056291268

4.3 FLOWCHART



4.4 APPLICATION OF PASCAL TO JACOBI

```
Program JACOBI (Input, Output);
      Type
           store2=packed array[1..3,1..4] of real;
           store4= packed array[1..5,1..5] of real;
      Var
           Initial:store2;
           S:store4;
           c,k,j,i,B:Integer;

Begin
      writeln ('Enter your initial values');
          For c:= 1 to 3 do
              Readln(S[1,c]);

      Begin
          Writeln('Enter the coefficients your equation
line by line');
          For k:= 1 to 3 do
          Begin
              For J:= 1 to 4 do
                 Readln(Initial[k,J]);
          end;
      end;
      for c:= 1 to 3 do
         write ('':2,S[1,c]:9);writeln;
      Begin
         For k:= 1 to 7 do
         Begin
               S[2,2]:=(Initial[1,4] -
(Initial[1,3]*S[1,3])-
(Initial[1,2]*S[1,2]))/Initial[1,1];
               S[3,3]:=(Initial[2,4] -
(Initial[2,3]*S[1,3])-
(Initial[2,1]*S[1,1]))/Initial[2,2];
               S[4,4]:=(Initial[3,4] -
(Initial[3,3]*S[1,2])-
(Initial[3,2]*S[1,2]))/Initial[3,3];

               write
('':2,S[2,2]:9,'':5,S[3,3]:9,'':5,S[4,4]:9);writeln;
```

```
                S[1,3]:=S[4,4];
                S[1,2]:=S[3,3];
                S[1,1]:=S[2,2];
            end
        end
end.
```

.5 APPLICATION OF PASCAL TO SIEDEL

```
rogram Siedel (Input, Output);
     Type
          store2=packed array[1..3,1..4] of real;
          store4= packed array[1..5,1..5] of real;
     Var
          Initial: store2;
          S: store4;
          c,k,j:Integer;

Begin
     writeln ('Enter your initial values');
         For c:= 1 to 3 do
             Readln(S[1,c]);

     Begin
         Writeln('Enter the coefficients your
equation line by line');
         For k:= 1 to 3 do
         Begin
             For J:= 1 to 4 do
                Readln(Initial[k,J]);
         end;
     end;
     for c:= 1 to 3 do
        write ('':2,S[1,c]:9);writeln;
     Begin
        For k:= 1 to 5 do
        Begin
              S[1,1]:=(Initial[1,4] -
(Initial[1,3]*S[1,3])-
(Initial[1,2]*S[1,2]))/Initial[1,1];
              S[2,1]:=(Initial[2,4] -
(Initial[2,3]*S[1,3])-
(Initial[2,1]*S[1,1]))/Initial[2,2];
```

```
                S[3,1]:=(Initial[3,4] -
(Initial[3,1]*S[1,1])-
(Initial[3,2]*S[2,1]))/Initial[3,3];

                write
(''':2,S[1,1]:9,''':5,S[2,1]:9,''':5,S[3,1]:9);writeln
;
                S[1,2]:=S[2,1];
                S[1,3]:=S[3,1];
          end
       end
end.
```

CHAPTER FIVE

5.0 DISCUSSION OF RESULTS

5.1 RESULTS

The exact solutions to problems in section 4.1 and 4.2 is (3, -2,5). Clearly Gauss-siedel performs better iterations with approximate result than Gauss-Jacobi. This is attributed to the successive displacement used by siedel. The Pascal program given in chapter four also illustrates this. It is also observed that the approximate solution becomes closer to the exact solution with increasing number of iterations.

5.2 CONCLUSION

The difference between the Jacobi and siedel methods is that in the latter, as each component of X_{r+1} is computed, we use it immediately in the iteration. For this reason, the Gauss-Siedel's method is sometimes called the method of successive displacement while Gauss-Jacobi's method is called simultaneous displacement.

The Gauss-Siedel and Jacobi methods do not always work. In some cases, one or both of these methods can fail to produce a good approximation to solution, regardless of the number of iterations performed. In such cases, the approximation are said to diverge. However, if by performing sufficiently many iterations, the solution can be obtained to any desired degree of accuracy, the approximations are said to converge.

The conditions for the convergence of method of simple iterations and Siedels method do not coincide but intersect. In some cases, Siedel's method yields more rapid convergence.

The following are the conditions for easy convergence.

(a) If the matrix is symmetric, then the Gauss-Siedel approximation to the solution convergence to the exact solution of the system for all choices of the initial approximation.

(b) Strictly diagonally dominant- if the absolute value of each diagonal entry is greater, then the sum of the absolute values of the remaining entries in the same row; that is

$|a_{11}| > |a_{12}| + |a_{13}| + \ldots + |a_{1n}|$

(c) When the coefficient matrix has a high proportion of zero (such matrices are called sparse), iterative methods can be used to advantages because there zeros simplify the iteration equations thereby reducing the amount of calculations.

(d) It may happen that a good estimate of the solution is known. If this used, as a starting value in an iterative method, then there is a good chance of obtaining a satisfactory approximate solution with fewer computations than a direct method would required.

(e) With clever programming, less computer memory is needed for iterative method than direct methods. Thus, if memory space is a problem, iterative may be essential.

(f) If the iterative methods diverge, as the rate of convergence is too slow, direct method may be essential.

UNPUBLISHED WRITE UP AND HANDOUT

DR. YOMI AIYESIMI 2000 "INTRODUCTION TO NUMERICAL ANALYSIS

DR. YOMI AIYESIMI 2000 " LECTURE NOTE ON GENERAL MATHEMATICS

ABDUL RAHEEM KOLA 2000" COMPUTER PROGRAMMING"

OMOYOSHO ROTIMI DAVID 2000 " THE USES OF COMPUTER IN PROPERTY VALUATION

REFERENCE

ANTHONY RALSTON & PHILIP RABINOURTZ: "FIRST COURSE IN numerical Analysis". MAC GRAW- HILL INTERNATIONAL EDITIONS.

E.A. VOLKOV: NUMERICAL METHODS" MIR PUBLISHERS MOSCOW PP 133

HOWARD ANTS AND CHRIS RONES: " ELEMENTARY LINEAR ALGEBRA WITH APPLICATION". PUBLISHED BY JOHN WILEY & SONS. PP 5,23,391

JIM WELSH & JOHN ELDER " INTRODUCTION TO PASCAL" 1ST EDITION, PUBLISHED BY ISO 1979.