

PROGRAMMING LOAD FLOW STUDIES USING SPARSITY TECHNIQUES

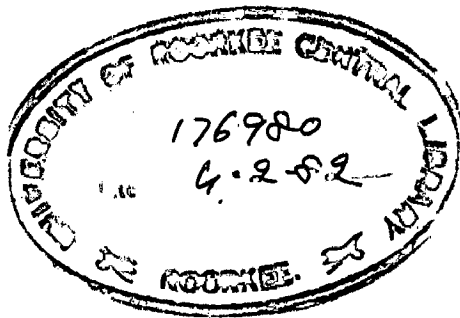
A DISSERTATION

Submitted in partial fulfilment of
the requirements for the award of the Degree
of
MASTER OF ENGINEERING
in
ELECTRICAL ENGINEERING
(Power System Engineering)

CHECKED
1995

by

D SURYANARAYANA



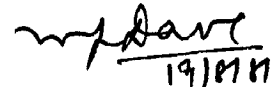
Ch. 82

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247672 (INDIA)
August, 1981

C E R T I F I C A T E

Certified that the dissertation entitled " PROGRAMMING
LOAD FLOW STUDIES USING SPARSITY TECHNIQUES " which is being
submitted by Shri D. SURYANARAYANA in partial fulfilment
for the award of the degree of Master of Engineering (Power
System Engineering) of University of Roorkee, is a record of
student's own work carried out by him under my guidance and
supervision. The matter embodied in this dissertation has not
been submitted for the award of any other degree or diploma.

This is further to certify that he has worked for a period
of seven months from 12th January to 14th August 1981 for preparing
this dissertation for Master of Engineering at this University.


19/8/81

(Dr. M.P. DAVE)
PROFESSOR IN ELECT. ENGG.
UNIVERSITY OF ROORKEE
ROORKEE, U.P.

Dated August, 1981.

ACKNOWLEDGEMENT

I wish to record my profound gratitude to Dr. M.P. Dave, Professor, Electrical Engineering Department, University of Roorkee for his able guidance, constant encouragement, un-failing patience, keen interest and valuable advice at every stage of preparation of this dissertation.

I am also thankful to Dr. D.R. Kohli, Professor & Head, Department of Electrical Engineering for providing the necessary facilities to carry out this work.

I am also thankful to the Ministry of Defence and Engineer-in-Chief's branch for having sponsored me for doing the Post Graduation at this University.

ROORKEE

Dated 18 August, 1981.


(D. SURYANARAYANA)

A B S T R A C T

The work presented in this work deals with application of sparsity techniques to load flow studies. As the size of the system increases, it becomes increasingly difficult to accommodate the problem within the core memory of the computer and sparse matrix techniques comes to the aid of the load flow analyst to overcome the problem of memory size and these techniques are absolutely essential for studying large modern interconnected systems. The technique also considerably speeds up the execution, as operations involving zero-elements are avoided and thereby finds application especially for on-line studies where execution time should be as small as possible.

Chapter II deals with the review of the load flow calculation methods and the equations on which the computations are based. In Chapter III the various sparsity techniques have been dealt with. Chapter IV describes the factorization methods which are generally applied and in detail deals with the bi-factorization method which has been used in this work. Detailed description of the computer algorithms for both Gauss-Seidel and Newton-Raphson method have been given in Chapter V.

The flow charts are given in appendix. Chapter VI gives the details and data of the problems studied and results obtained.

In this work sparsity techniques have been applied to load flow solution both by Gauss-Seidel and by Newton-Raphson method.

In the Newton-Raphson method the techniques of bi-factorization has been applied.

It is seen that the Gauss-Seidal method requires a large number of iterations for getting the solution. The Newton-Raphson method takes very few iterations to get the solution. However for the smaller systems the execution time is more.

The methods have been tested on a 5 bus, 8 bus and IEEE standard 57 bus and 118 bus systems. The computational work has been carried on DEC 2050 computer system of the Roorkee University and execution times referred in this work correspond to DEC 2050 system.

LIST OF PRINCIPAL SYMBOLS USED

Subscripts/Superscripts

P, q, I bus numbers

K, v iteration numbers

I_P	=	Vector of bus currents
E_{Bus}	=	Bus voltage vector
Y_{Bus}	=	admittance matrix
Z_{Bus}	=	Impedance matrix
E_P, V_P	=	Vector of Bus voltages
E_P^*	=	Conjugate of E_P
R	=	Resistance of transmission line
X	=	Reactance of transmission line
Y_{pp}	=	Driving point admittance
Y_{pq}	=	Transfer admittance
δ	=	Phase angle of the bus voltage
θ	=	angle of the elements of the admittance matrix expressed in polar coordinators
P_p	=	Real power input at Bus P
Q_p	=	Reactive power input at Bus P
S_p	=	Power at P bus $P_p - jQ_p$
Y_{pq}	=	admittance between buses P and Q
Δp	=	Difference between actual and computed real bus power
Δq	=	Difference between actual and computed reactive " "
$\Delta \delta$	=	Correction required in the bus voltage phase angle
$\Delta E $	=	Correction required in the bus voltage magnitude.

C O N T E N T S

CHAPTER	PARTICULARS	PAGE
I	INTRODUCTION	1
II	LOAD FLOW PROBLEM	5
III	SPARSITY TECHNIQUES	22
IV	MATRIX FACTORIZATION	34
V	ALGORITHMS AND FLOW CHARTS	50
VI	TEST PROBLEMS AND RESULTS	58
VII	CONCLUSION	97
	REFERENCES	
	APPENDIX	

CHAPTER - I

INTRODUCTION

1.1 Load Flow Studies

The principal objective of a utility is to deliver power to the consumer satisfactorily (i.e. with voltage and frequency within limits) and in a manner most economical to the company from the transmission and distribution points of view.

Load flow calculations provide power flows and voltages for a specified power system subject to the regulating capability of generators, condensers, and tap changing under load transformers as well as specified net interchange between individual operating systems. This information is essential for the continuous evaluation of the current performance of a power system and for analyzing the effectiveness of alternative plans for system expansion to meet increased load demand. These analyses require the calculation of numerous load flows for both normal and emergency operating condition. Thus load flow studies form the back bone for the design and operation of a power system.

The load flow problem consists of the calculation of power flows and voltages of a network for specified terminal or bus conditions. A single phase representation is adequate since power systems are usually balanced. Associated with each bus are four quantities: the real and reactive power, the voltage magnitude, and the phase angle. Three types of buses are represented in the load flow calculation and at a bus, two of the four quantities are specified. It is necessary to select one bus, called the slack bus, to provide the additional real and reactive power to supply the transmission losses, since these are unknown until the final solution is obtained.

At this bus the voltage magnitude and phase angle are specified. The remaining buses of the system are designated either as voltage controlled buses or load buses. The real power and voltage magnitude are specified at a voltage controlled bus. The real and reactive powers are specified at a load bus.

A given power system subject to a given set of power demands at its various buses can be operated in an infinite number of states and still satisfy the given demands. For the systems engineer the job becomes one of selecting the best possible state out of the myriad of possibilities. He selects this particular one after comparing a number of possible alternatives, obtained from a load flow study.

The state of the system could be represented by a vector, the state vector, made up of the bus voltage magnitudes and phase angles. With all the bus voltages known to both magnitude and phase, we in effect also know the line flows. In other words, we know the total structure of the power flow in the system.

Having obtained from a such study a selection of possible load flow configurations, the one particular configuration we should use is selected on the following basis.

1. The total amount of real power in the network emanates from the generator stations, the location and size of which are fixed. The generation must equal the demand at each moment, and since this power must be divided between the generators in a unique ratio in order to achieve optimum economic operation,

we conclude that the individual generator outputs must be closely maintained at predetermined set points. It is important to remember that the demand undergoes slow but wide changes throughout the 24 h of the day. We must therefore slowly, either continuously or in discrete steps, change these set points.

2. Certain transmission links can carry only certain amounts of power, and we must make sure that we do not operate these links too close to their thermal / stability limits.
3. It is necessary to keep the voltage levels of certain buses at rather close tolerances. This can be achieved by proper scheduling of reactive powers.
4. If the power system is part of a larger pool, it must fulfil certain contractual power-scheduling commitments via its tie lines to neighboring systems.
5. The disturbances following a massive network fault can cause system outages, the effects of which can be minimized by proper prefault load flow strategies.

1.1.1 Load Flow Studies of Large Systems.

As the size of the net work increases it becomes increasingly difficult to accommodate the problem in the available memory of the computer. In order to bring such problems within the memory size of the computer special techniques called sparsity techniques have to be used.

1.2 Sparsity Techniques.

A matrix having only a small percentage of non zero elements is said to be sparse. In a practical sense any $n \times n$ matrix is classified as sparse if it has order of n nonzero

elements, say two to ten non zero elements in each row, for large n . In other words if α is the total number of nonzero elements in the matrix and n is the order of matrix, then $\alpha \ll n^2$.

Sparse matrices occur in the solution of many important practical problems, e.g., in structural analyses, network theory and power distribution systems, numerical solution of differential equations, graph theory, as well as in genetic theory, behavioral and social sciences, and computer programming. As our technology increases in complexity, we can expect that large sparse matrices will continue to occur in many future applications involving large systems, e.g., scheduling and simulation of interconnected power system problems, scheduling of metropolitan fire engines of fire departments and ambulances, simulation of traffic lights, pattern recognition and urban planning.

The formulation and solution of problems in power systems, social, behavioral and environmental sciences in many cases lead to large sparse systems [29]. If such systems are non linear, then their linearization - often the first step towards the solution - will result in still larger sparse systems.

In the present work we will confine ourselves to the application of sparse matrix techniques to load flow problem.

CHAPTER - II
LOAD FLOW PROBLEM

The overall load flow problem can be sub divided into two sub problems namely:

1. The formulation of a suitable mathematical network model.

The model must describe adequately the relationships between voltages and powers in the interconnected system.

2. The application of numerical method for a solution. The solution must satisfy Kirchoff's laws, i.e., the algebraic sum of all flows at a bus must equal zero, and the algebraic sum of all voltages in a loop must equal zero. Other constraints placed on the solution are: the capability limits of reactive power sources, the tap setting range of transformers as well as specified net inter change between individual operating systems.

2.1 Network Model Formulation

The first step in any analysis of an electric energy system must be the formulation of a suitable network model. Such a model should relate a selected set of network voltages to another selected set of network currents or powers.

The networks that we shall be concerned with in our work are very large, containing often many hundreds, perhaps thousands, of individual network elements, and when we combine these individual elements to form the overall system model, we are faced with the need of performing tens of thousands of ele-

mentary algebraic operations.

As load flow studies are invariably carried out on a digital computer, so there is a need to develop network assembly methods that are systematic and amenable to computer use. The tabular nature of matrices makes them particularly well adapted to digital computer programming. Also, the methods should possess flexibility with regard to network changes. If we wish to perform investigations of the effects of certain localized network changes, we should be able to do so with a minimum of computational effort.

A network matrix equation provides a convenient mathematical model for a digital computer solution. The elements of a network matrix depend on the selection of the independent variables, which can be either currents or voltages. Correspondingly, the elements of the network matrix will be impedances or admittances.

The electrical characteristics of the individual network components can be presented conveniently in the form of a primitive network matrix. This matrix, while adequately describing the characteristics of each component, does not provide any information pertaining to the network connections. It is necessary, therefore, to transform the primitive network matrix into a network matrix that describes the performance of the interconnected network.

The form of the network matrix used in the performance equation depends on the frame of reference, namely, bus or loop.

In the bus frame of reference the variables are the nodal voltages and nodal currents. In the loop frame of reference the variables are loop voltages and loop currents.

Generally the bus admittance matrix is chosen to represent the network for carrying out load flow studies because of its easy formulation, and alteration and its sparse nature.(25) The bus admittance matrix is a very sparse matrix and the degree of sparsity for larger systems may be in excess of 95 percent (31). We have also chosen in our method the bus frame of reference in the admittance form.

2.2 Solution Techniques

The following table gives a brief summary of some of the main types of load flow solutions currently in application, and the requirements imposed on the numerical processes(1).

Load Flow Calculations - Types and - Requirements

<u>Types of Solution</u>	
Accurate	Approximate
Unadjusted	Adjusted
Off-line	On-line
Single case	Multiple cases

Properties required of Load-Flow Solution Method.

High speed	especially for:	Large systems, real time applications, multiple cases interactive applications
------------	-----------------	--

Low storage Especially for: Large systems Computer with
small core storage availability

Reliability Especially for: ill-conditioned problems outage
studies real-time applications

Versality Ability to handle conventional
and special features (adjust-
ments, representation of power
system apparatus); suitability
for incorporation into more
complicated processes.

Simplicity Ease (and cost) of coding,
maintaining, and enhancing
the algorithm and computer
program based on it.

Prior to and for some time after the advent of digital computers, load-flow solutions were obtained using network analyzers. The first really practical automatic digital solution methods appeared in literature in 1956 and subsequently (11-13).

These Y-matrix iterative methods were well suited to the early generations of computers, since they require minimal computer storage. Although they perform satisfactorily on many problems, they converge slowly, and too often not at all. The incentive to overcome this deficiency led

to the development of Z-matrix methods (3-5) which converge more reliably but sacrifice some of the advantages of Y-matrix iterative methods, notably storage and speed when applied to large systems. Around the same time, the Newton-Raphson method was shown to have very powerful convergence properties (6-7), but was computationally uncompetitive. Major break through in power-system network computation came in the mid - 1960's, with the development by Tinney and others of very efficient sparsity programmed ordered elimination (8). One of its earliest successes was in dramatically improving the computing speed and storage requirements of Newton's method, which has now come to be widely regarded as the preeminent general purpose load flow approach (8) and has been adopted by much of industry. (1)

2.3.1 Optimal Load Flow

An optimal load-flow calculation optimizes the active- and reactive-power dispatch of a system, including as control variables those single-criterion-control parameters that are adjusted during an ordinary load flow solution. All the relevant static limit constraints on the system operation are enforced.

Some practical approaches to this problem, including the now classical Dommel and Tinny method (9), adjust the control variables in some optimum-seeking manner in between conventional load flow solutions. Polar coordinate load-flow

formulations are the most natural choice for the optimization application, since voltage magnitudes in particular are then explicitly available as variables for control and limit enforcement purposes.

2.3.2 Diakoptic Techniques

Prior to the advent of sparsity techniques diakoptic approach was used to be applied for solving large problems.

The basic idea of diakoptics is to solve a large system by a breaking or tearing it apart into smaller subsystems; to first solve the individual parts, and then to combine and modify the solutions of the torn parts to yield the solutions of the original untorn problem. The result of the procedure is identical to one that would have been obtained if the system had been solved as one (10).

The uses of diakoptics are at least two fold: In the first application, larger systems can be solved efficiently by the use of diakoptics on a given computer than would otherwise be possible by processing the torn parts through the computer serially. The second application employs a multiplicity of computers which essentially operate in parallel, and thus provide more speed of execution than by the use of a single computer. The computer can be physically next to each other, thus forming a cluster of computers, or they can be miles apart. So we can expect larger problems to be solved with greater speeds by the use of diakptics than by solving the problem by conventional methods.

2.3 Network Performance Equations

The equation describing the performance of the network of a power system using the bus frame of reference in the admittance form is

$$\bar{I}_{BUS} = Y_{BUS} \bar{E}_{BUS} \quad (1)$$

The bus impedance and admittance matrices can be formed for the network including the ground bus. The elements of the matrices, then, will include the effects of shunt elements to ground such as static capacitors and reactors, line charging, and shunt elements of transformer equivalents. When the ground bus is included and selected as the reference node, the bus voltages in the above network performance equation are measured with respect to ground.

2.3.1 Bus Loading Equations

The real and reactive power at any bus p is

$$P_p - jQ_p = E_p^* I_p \quad (2)$$

and the current is

$$I_p = \frac{P_p - jQ_p}{E_p^*} \quad (3)$$

Where I_p is positive when flowing into the system.

In the formulation of the network equation, if the shunt elements to ground are included in the parameter matrix, then equation (3) is the total current at the bus.

2.3.2 Line Flow Equations

After the iterative solution of bus voltages is completed, line flows can be calculated. The current at bus p in the line connecting bus p to q is

$$i_{pq} = (E_p - E_q)Y_{pq} + E_p \frac{Y'_{pq}}{2} \quad (4)$$

where Y_{pq} = line admittance

Y'_{pq} = total line charging admittance

$E_p \frac{Y_{pq}}{2}$ = current contribution at bus p due to line

charging

The power flow, real and reactive, is

$$P_{pq} - jQ_{pq} = E_p^* i_{pq} \quad (5)$$

or

$$P_{pq} - jQ_{pq} = E_p^* (E_p - E_q)Y_{pq} + E_p^* E_p \frac{Y'_{pq}}{2} \quad (6)$$

Where at bus p the real power flow from bus p to q is P_{pq}

and the reactive is Q_{pq} . Similarly, at bus q the power

flow from q to p is

$$P_{qp} - jQ_{qp} = E_q^* (E_q - E_p)Y_{pq} + E_q^* E_q \frac{Y'_{pq}}{2} \quad (7)$$

The power loss in line p-q is the algebraic sum of the power flows determined from equations (6) and (7).

2.4 Solution Methods

2.4.1 Gauss Iterative Method Using Y_{BUS}

The solution of the load flow problem is initiated by assuming voltages for all buses except the slack bus, where the voltage is specified and remains fixed. Then, currents

are calculated for all buses except the slack bus s from the bus loading equation

$$I_p = \frac{P_p - jQ_p}{E_p^*} \quad \begin{matrix} p = 1, 2, \dots, n \\ p \neq s \end{matrix} \quad (8)$$

where n is the number of buses in the network. The performance of the network can be obtained from the equation

$$\bar{I}_{BUS} = Y_{BUS} \bar{E}_{BUS} \quad (9)$$

Selecting the ground as the reference bus, a set of $n - 1$ simultaneous equations can be written in the form

$$E_p = \frac{1}{Y_{pp}} \left\{ I_p - \sum_{\substack{q=1 \\ q \neq p}}^n Y_{pq} E_q \right\} \quad \begin{matrix} p = 1, 2, \dots, n \\ p \neq s \end{matrix} \quad (10)$$

The bus currents calculated from equation (8), the slack bus voltage, and the estimated bus voltage are substituted into equation (10) to obtain a new set of bus voltages. These new voltages are used in equation (8) to recalculate bus currents for a subsequent solution of equation (10). The process is continued until changes in all bus voltages are negligible. After the voltage solution has been obtained, the power at the slack bus and line flows can be calculated.

The network equation (10) and the bus loading equation (8) can be combined to obtain (11)

$$E_p = \frac{1}{Y_{pp}} \left(\begin{array}{c} P_p - jQ_p \\ E_p^* \end{array} - \sum_{\substack{q=1 \\ q \neq p}}^n Y_{pq} E_q \right) \quad \begin{array}{l} p = 1, 2, \dots, n \\ p \neq s \end{array} \quad (11)$$

which involves only bus voltages as variables. Formulating the load flow problem in this manner results in a set of nonlinear equations that can be solved by an iterative calculation.

The iterative process must continue until the magnitude of the change of the bus voltage between two consecutive iterations is less than a certain tolerance level for all bus voltages.

Computation of Slack Bus Power

This step is simple. After the iterations have converged, substitute our computed voltages (plus the assumed voltages of slack bus) in equation 2 and obtain directly the slack bus power.

Computation of Line Flows

The final step in the load flow analysis is the computation of the load flows on the various transmission lines of network, using the equations already described.

2.4.2 Gauss-Seidel Iterative Method Using Y_{BUS}

In this method the new calculated voltage immediately replaces the old value and is used in the solution of the subsequent equations. The flow diagram is shown in Appendix.

2.4.3 Newton-Raphson Method Using Y_{BUS}

The load flow problem can be solved by the Newton-Raphson method using a set of nonlinear equations to express the specified real and reactive powers in terms of bus voltages (7). What follows is the description of the Newton-Raphson method using polar coordinates.

The power at bus p is

$$P_p - jQ_p = E_p^* I_p \quad (14)$$

Substituting from the network performance equation (1) for I_p in (14)

$$P_p - jQ_p = E_p^* \sum_{q=1}^n Y_{pq} E_q \quad (15)$$

Since $E_p = |E_p| e^{j\delta_p}$ and $Y_{pq} = |Y_{pq}| e^{-j\theta_{pq}}$

Equation (15) becomes

$$P_p - jQ_p = \sum_{q=1}^n |E_p E_q Y_{pq}| e^{-j(\theta_{pq} + \delta_p - \delta_q)} \quad (16)$$

Since $e^{-j(\theta_{pq} + \delta_p - \delta_q)} = \cos(\theta_{pq} + \delta_p - \delta_q) - j \sin(\theta_{pq} + \delta_p - \delta_q)$, the

real and imaginary components of power are

$$P_p = \sum_{q=1}^n |E_p E_q Y_{pq}| \cos(\theta_{pq} + \delta_p - \delta_q) \quad (17)$$

$$Q_p = \sum_{q=1}^n |E_p E_q Y_{pq}| \sin(\theta_{pq} + \delta_p - \delta_q) \quad p = 1, 2, \dots, n-1$$

(assuming nth bus as the reference bus)

This formulation results in a set of nonlinear simultaneous equations, two for each bus of the system. The real and reactive powers P_p and Q_p are known and the real and imaginary components of voltage e_p and f_p are unknown for all buses except the slack bus, where the voltage is specified and remains fixed. Thus there are $2(n - 1)$ equations to be solved for a load flow solution.

The Newton-Raphson method requires that a set of linear equations be formed expressing the relationship between the changes in real and reactive powers and the components of bus voltages. Expressing in the matrix form

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} \Delta \delta \\ \Delta |E| \end{bmatrix} \quad (18)$$

The elements of the Jacobian are calculated from equations (17) and are

For J_1 :

$$\frac{\partial P_p}{\partial \delta_q} = |E_p E_q Y_{pq}| \sin(\theta_{pq} + \delta_p - \delta_q) \quad q \neq p \quad (19)$$

$$\frac{\partial P_p}{\partial \delta_p} = \sum_{\substack{q=1 \\ q \neq p}}^n |E_p E_q Y_{pq}| \sin(\theta_{pq} + \delta_p - \delta_q)$$

For J_2 :

$$\frac{\partial P_p}{\partial |E_q|} = - |E_p Y_{pq}| \cos (\theta_{pq} + \delta_p - \delta_q) \quad q \neq p \quad (20)$$

$$\frac{\partial P_p}{\partial |E_p|} = 2 |E_p Y_{pp}| \cos \theta_{pp} + \sum_{\substack{q=1 \\ q \neq p}}^n |E_q Y_{pq}| \cos (\theta_{pq} + \delta_p - \delta_q)$$

For J_3 :

$$\frac{\partial Q_p}{\partial \delta_q} = |E_p E_q Y_{pq}| \cos (\theta_{pq} + \delta_p - \delta_q) \quad q \neq p \quad (21)$$

$$\frac{\partial Q_p}{\partial \delta_p} = \sum_{\substack{q=1 \\ q \neq p}}^n |E_p E_q Y_{pq}| \cos (\theta_{pq} + \delta_p - \delta_q)$$

For J_4 :

$$\frac{\partial Q_p}{\partial |E_q|} = |E_p Y_{pq}| \sin (\theta_{pq} + \delta_p - \delta_q) \quad q \neq p \quad (22)$$

$$\frac{\partial Q_p}{\partial |E_p|} = 2 |E_p Y_{pp}| \sin \theta_{pp} + \sum_{\substack{q=1 \\ q \neq p}}^n |E_q Y_{pq}| \sin (\theta_{pq} + \delta_p - \delta_q)$$

2.4.4. Voltage Controlled Buses

A modification of, or deviation from, the normal computational procedures for the solution of the load flow problem is required to take into account voltage controlled buses. At these buses the voltage magnitude and the real power are specified.

In the Gauss and Gauss-Seidel methods using Y_{BUS} , the reactive power at a voltage controlled bus p must be calculated before proceeding with the calculation of voltage at that bus. Separating the real and imaginary parts of the bus power equation

$$P_p - jQ_p = E_p^* \sum_{q=1}^n Y_{pq} E_q$$

the reactive bus power is

$$Q_p = e_p^2 B_{pp} + f_p^2 B_{pp} + \sum_{\substack{q=1 \\ q \neq p}}^n f_p (e_p G_{pq} + f_q B_{pq}) - e_p (f_q G_{pq} - e_q B_{pq}) \quad (23)$$

where e_p and f_p are the components of voltage at bus p .

The values of e_p and f_p must satisfy the relation

$$e_p^2 + f_p^2 = (E_p \text{ scheduled})^2 \quad (24)$$

in order to calculate the reactive bus power required to provide the scheduled bus voltage. The present estimates of e_p^k and f_p^k must be adjusted, therefore, to satisfy equation (24)

The phase angle of the estimated bus voltage is

$$\delta_p^k = \arctan \frac{f_p^k}{e_p^k}$$

Assuming that the angles of the estimated and scheduled voltages are equal, then adjusted estimates for e_p^k are

$$e_p^k \text{ (new)} = |E_p| \text{ (Scheduled)} \cos \delta_p^k \quad (23)$$

$$f_p^k \text{ (new)} = |E_p| \text{ (scheduled)} \sin \delta_p^k$$

Substituting $e_p^k \text{ (new)}$ and $f_p^k \text{ (new)}$ in equation (23), the reactive power Q_p^k is obtained and is used with $E_p^k \text{ (new)}$ for calculating the new voltage estimate $E_p^{k+1} \text{ (new)}$

for calculating the new voltage estimate

In actual practice the limits of reactive power source at the voltage controlled bus must be taken into account. If the calculated Q_p^k exceeds the maximum capability $Q_p \text{ (max)}$ of the source the maximum value is taken as the reactive power at that bus. If the calculated value is less than minimum capability $Q_p \text{ (min)}$ the minimum value is used. In either case it is impossible to obtain a solution with the specified scheduled voltage and therefore $E_p^k \text{ (new)}$ cannot be used in the calculation of E_p^{k+1} .

In the Newton-Raphson method the equations for a voltage controlled bus p are (in polar coordinates)

$$P_p = \sum_{q=1}^n |E_p^E Y_{pq}| \cos(\theta_{pq} + \delta_p - \delta_q) \quad (25)$$

and

$$|E_p^2| = E_p^2 \quad (26)$$

where equation (26) replaces the equation for the reactive power. The matrix equation relating the changes in bus powers and the square of voltage magnitudes to changes in voltage magnitude and phase angle is

ΔP	=	J_1	J_2	$\Delta \delta$
ΔQ		J_3	J_4	$\Delta E $
$ \Delta E ^2$		J_5	J_6	

The elements of the submatrices J_1, J_2, J_3 and J_4 are calculated as explained in section (2.4.3). From equation (25) both off diagonal & diagonal elements of J_5 are zero. The off diagonal elements of J_6 are zero as

$$\frac{\partial |E_p|^2}{\partial |E_q|} = 0 \quad q \neq p \quad (27)$$

and diagonal elements of J_6 are

$$\frac{\partial |E_p|^2}{\partial |E_p|} = 2 E_p \quad (28)$$

The change in the square of the voltage magnitude at bus p is $\Delta E_p^k|^2 = (E_p^k | \text{scheduled})^2 - |E_p^k|^2$ (29)

If sufficient reactive capability is not available to hold the desired magnitude of bus voltage the reactive power must be fixed at a limit. In this case the voltage cannot be maintained and the solution is not the desired one.

CHAPTER - III
SPARSITY TECHNIQUES

3.1 Introduction

In sparse matrix techniques, only the nonzero elements of the matrix are stored and processed, which will not only reduce the memory requirement but also reduce substantial amount of time, as operations involving zeros are not performed. Nearly all the schemes make use of two storage components.

- a) A facility for storing either the non-zero elements or an array of the matrix which includes all of the non-zero elements. This usually takes the form of a one-dimensional array and will be called the primary array.
- b) A means of recognizing which elements of the matrix are stored in the primary array. This usually takes the form of one or more one-dimensional arrays of integer identifiers, which will be called the secondary store.

3.2 Binary Identification

A novel scheme which makes use of the binary nature of computer storage is to record the pattern of non-zero elements of the matrix as the binary digits of secondary array elements. The matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.67 & 0 & 3.12 \\ -1.25 & 0.29 & 0 & 0 & 2.31 \end{bmatrix} \quad 3.1$$

has a pattern of non-zero elements indicated by the binary sequence

row 1	row 2	row 3
0 0 0 0 0	0 0 1 0 1	1 1 0 0 1

Hence this matrix could be stored by means of a primary array containing the five non-zero elements (2.67 3.12 -1.25 0.29 2.31) and a secondary store containing the binary sequence could be held in a word with fifteen or more bits, however, for larger matrices a number of words would be required.

If an $x \ n$ matrix has r as the ratio of the number of non-zero elements to total elements and if two words each of γ bits are required to store each non-zero element, then the primary array will occupy $2mnr$ words and the secondary array will occupy approximately mn/γ words. Since $2mn$ words would be required to store the matrix in the conventional way, the storage compaction of the binary identification scheme may be expressed as the ratio c where

$$2mnc \approx 2mnr + \frac{mn}{\gamma} \quad 3.2$$

giving

$$c \approx r + \frac{1}{2\gamma} \quad 3.3$$

This storage scheme differs from other sparse schemes in that some storage space (a single bit in the secondary store) is allocated to every zero element. It is therefore less efficient for very sparse matrices than schemes which do not contain any storage allocation associated with zero elements. Moreover, the main drawback is the difficulty

of implementing matrix operations with matrices stored in this way. Normally such implementations would produce much less efficient programs than could be achieved by using other sparse storage schemes.

3.3 Random Packing

Every non-zero element entered into the primary array may be identified by specifying its row and column numbers in the corresponding locations of two secondary arrays. Since each element is individually identified it is possible to store them in a random order. Thus matrix (3.1) could be represented by

Real array A = (0.29 3.12 -1.25 2.67 2.31 0 - -);
 Integer array IA = (3 2 3 2 3 0 - -)(3.25)
 Integer array JA = (2 5 1 3 5 0 - -)

One advantage of random packing is that extra non-zero elements can be added to the matrix by inserting them at the end of the list without disturbing the other items. It is often convenient to have a null entry in a secondary array to indicate termination of the list.

3.4 Systematic Packing

If the elements of a sparse matrix have been read in or constructed in a systematic order or have been sorted into a systematic order there is no need to adopt both row and column indices for each element. For row-wise packing it is the row indices which may be dispensed with, except insofar as it is necessary to specify where each row begins.

3.4.1. The Use of Row Address

The address of the first non-zero element in each row may be specified in a separate integer array. For example, matrix 3.1 could be represented by

$$\begin{aligned} \text{Real array } A &= (2.67 \ 3.12 \ -1.25 \ 0.29 \ 2.31) \\ \text{Integer array } JA &= (3 \ 5 \ 1 \ 2 \ 5) \quad (3.5) \\ \text{Integer array } ISTART &= (1 \ 1 \ 3 \ 6) \end{aligned}$$

The array of row addresses ISTART has been constructed so that the number of non-zero elements in row i is $ISTART(i+1) - ISTART(i)$, hence for a matrix with m rows, ISTART will contain $m+1$ entries.

3.4.2 The Use of Dummy Elements

Either in place of row addresses, or as an adjunct to them dummy elements may be included to indicate the start of each row and the end of the matrix. Several formats are possible for the dummy element and the corresponding entry in the column index array. For instance, a zero entry in the array JA could mark the presence of a dummy element and the dummy element itself could specify the row number (or be zero to indicate the end of the matrix). Hence matrix (3.1) would appear as

$$\begin{aligned} \text{Real array } A &= \left\{ \begin{matrix} 2 \\ \end{matrix} \right\} 2.67 \ 3.12 \ \left\{ \begin{matrix} 3 \\ \end{matrix} \right\} -1.25 \ 0.29 \ 2.31 \ \left\{ \begin{matrix} 0 \\ \end{matrix} \right\} \quad (3.6) \\ \text{Integer array } JA &= (0) \ 3 \ 5 \ (0) \ 1 \ 2 \ 5 \ (0) \end{aligned}$$

Alternatively, the row number could be specified in the integer array and distinguished from column numbers by a change of sign. In this case the dummy element itself would not be used.

Matrix (3.1) would appear as

$$\begin{array}{l} \text{Real arrayA} = \left\{ \begin{array}{ccccccc} (x) & 2.67 & 3.12 & (x) & -1.25 & 0.29 & 2.31 \\ (-2) & 3 & 5 & (-3) & 1 & 2 & 5 \end{array} \right\} \quad (3.7) \\ \text{Integer arrayJA} = \left\{ \begin{array}{ccccccc} (x) \\ (0) \end{array} \right\} \end{array}$$

In some cases (e.g. for the sparse matrix multiplication of section 3.) it is easier to program matrix operations if the integer identifier for a dummy element is larger rather than smaller than the column indices. This may be achieved by making the identifier equal to the row number plus a constant, the constant being larger than the largest column number. In a similar way it may be convenient to use an even larger number to indicate the end of the matrix. Thus, matrix (3.1) would appear as

$$\begin{array}{l} \text{Real arrayA} = \left\{ \begin{array}{ccccccc} (x) & 2.67 & 3.12 & (x) & -1.25 & 0.29 & 2.31 \\ (10002) & 3 & 5 & (10003) & 1 & 2 & 5 \end{array} \right\} \\ \text{Integer arrayJA} = \left\{ \begin{array}{ccccccc} (x) \\ (99999) \end{array} \right\} \quad (3.8) \end{array}$$

A further alternative use for the dummy element is to specify in the column index position the number of elements in the next row. If a dummy element is included for each row even if it is null, then there is no need to record the row number.

Thus matrix (3.2) could be stored as

$$\begin{array}{l} \text{Real arrayA} = \left\{ \begin{array}{ccccccc} (x \ x) & 2.67 & 3.12 & (x) & -1.25 & 0.29 & 2.31 \\ (0 \ 2) & 3 & 5 & (3) & 1 & 2 & 5 \end{array} \right\} \quad (3.5) \\ \text{Integer arrayJA} = \left\{ \begin{array}{ccccccc} (x) \\ (3) \end{array} \right\} \end{array}$$

The number of rows in the matrix will need to be specified elsewhere.

In any of the dummy element schemes shown above except the

first, the dummy elements in the real array may be omitted to save storage space. However, if this is done the addresses of the elements and their column will not coincide.

3.5 Compound Identifiers

In the random packing scheme (3.5) it is possible to reduce the storage requirement by combining the two indices for each element so that they can be held in one integer store. A suitable compound identifier would be $\bar{m} + j$ where \bar{n} is an integer equal to or greater than the total number of columns in the matrix. In a similar way it is possible to avoid the use of dummy elements for systematic packing by using a compound identifier for the first element of each row. For example, matrix (3.1) could be represented by

$$\begin{aligned} \text{Real array } A &= (2.67 \ 3.12 \ -1.75 \ 0.29 \ 2.31 \ \quad x) \\ \text{Integer array } JA &= (2003 \ 5 \ 3001 \ 2 \ 5 \ 99999) \end{aligned} \quad (3.10)$$

However, unless compound identification yields necessary or highly desirable storage space savings, it should not be used because

- (a) extra program will nearly always be required to interpret the compound identifiers and
- (b) it must not be used for matrices whose orders are so large that overflow of the integer register would result.

3.6.1. Necessity of Linked Storage for Matrix Inversion

The schemes so far described cannot be used when matrix inversion is required as in the case of Newton-Raphson method for solving the simultaneous equations. These simultaneous

equations can be solved by direct matrix inversion. Although these methods are fairly easy to program, they cannot, however exploit, sparsity and unfortunately produce a full inverse matrix for large problems, for storage is therefore extremely large and the methods are very inefficient.

The alternative direct methods to matrix inversion are the factorization techniques based on Gauss elimination. These methods sparsity can be exploited and, with a suitable ordering technique, a direct solution can be obtained with a minimum amount of storage and computation time. Factorization methods are, in themselves, relatively easy to program, since the technique of each method is based solely on the Gauss elimination process. The simplicity of these methods, however, is upset when the coefficient matrix as a whole is not stored, but only the non zero elements in a compact form.

The previously described storage schemes can be adopted if the number of non zero elements in one column did not vary in the course of computation. However during the factorization process new non zero elements are continually big generated. Also, elements which were previously non-zero, may become zero. However by using a suitable ordering technique, the number of new non-zero elements produced during factorization can be minimized. It is evident that the compacting and indexing schemes must be capable of implementing efficiently these continuous changes during factorization by incorporating the new non-zero elements in store and recording the available storage space.

In order to deal with this problem of continually changing of non zero terms in each column and row of the reduced matrix, a linked list technique becomes essential which is described in the next section.

3.6.2 Linked List Technique

Suppose we have stored a list of numbers in an array defined as VALUE.

Location	1	2	3	4	5	6
VALUE	30.5	50.9	26.3	45.7	-	-

There are many sequences in which the same list of numbers can be stored. Suppose, for instance, that these numbers are to be stored in ascending numerical order. One way of achieving this requirement is to change the numerical value of the elements in each location. Doing this, the new numbers in the array VALUE would be:

Location	1	2	3	4	5	6
VALUE	26.3	30.5	45.7	50.9	-	-

So far this process is very straightforward, even if a re-ordering routine has to be included in the program to arrange the numbers in a certain preferred sequence. Suppose now that it is necessary to add a new number to the list. If the new number can be added to the end of the list then the process is very simple. The size of the list is increased by one and the new number is inserted in the extra available position. If, however, the number is to be inserted in the middle of the existing list, then the process is more complicated.

Let the new number to be added be 28.2 and the new list to be kept in an ascending numerical order. One way of achieving this is to move the last value one position down the list, move the last but one value to the position previously occupied by the last value and continue this process until the new value can be inserted into the appropriate position. Using this method, the new list with the value 28.2 added becomes:

Location	1	2	3	4	5	6
VALUE	26.3	28.2	30.5	45.7	50.9	-

For a long list, this process of inserting new in a particular position within an existing list can be computationally very time-consuming. A more efficient method is to use a technique known as a linked list. Linked lists enable the numerical values of the numbers to be stored in any order, the desired sequence of the numbers being determined by the linking techniques. This linking technique consists of allocating a storage location for the numerical value of the next item. This associated address is shown for one location below.

	Numerical value of an item	
	address of next item ≠ 0 if more items to follow = 0 if present item is last item	

This linking technique requires the introduction of a new array which may be called LNXT and in which the address of the next required number in the list is stored. Using such an array, the list of the numbers in the original sequence is:

	*					
Location	1	2	3	4	5	6
<hr/>						
VALUE	30.5	50.9	26.3	45.7	-	-
LNXT	2	3	4	0		

In addition to the above list, it is necessary to record the address of the first number in the list. This can be stored in practice as a single integer variable, but for simplicity in the above example this element is marked by an asterisk.

To change the order of the list into ascending numerical values, the sequence of numbers in the array VALUE can be left undisturbed and the order modified by changing the addresses in the array LNXT. This gives:

			*			
Location	1	2	3	4	5	6
<hr/>						
VALUE	30.5	50.9	26.3	45.7	-	-
LNXT	4	0	1	2		

If now the number 28.2 is to be added in the list and the sequence of numbers is to remain in ascending numerical order, it is sufficient to add this new number to the end of the existing list and to change the addresses in the array LNXT. This gives:

Location	1	2	* 3	4	5	6
VALUE	30.5	50.9	26.3	45.7	28.2	-
LNXT	4	0	5	2	1	

Using the array LNXT to re-arrange the sequence of numbers in order to accommodate a new number within the list is computationally considerably more efficient than the shuffling process that had to be performed with the numbers in the array VALUE in the previous technique. Previously all the numbers greater than the new value had to be shifted sequentially, which, with long lists, could involve a considerable number of operations. In the present technique only one value in the original LNXT has to be changed and one new value added. The merits of this scheme are therefore clearly evident. It does, however, require additional storage and this must be balanced against the reduced computation time.

3.6.3 Techniques Adopted in the Present Work

In our work we have adopted the systematic packing technique of storing the matrices. In the case of Gauss-Seidel method only the diagonal & upper triangular non zero elements are stored and a five digit index number has been given for each of the diagonal elements viz., 10001 for the first diagonal element, 10002 for the second diagonal element and so on. An index number of 99999 has been given to the last diagonal element which is also the last element of the matrix.

In the case of Newton-Raphson method the full admittance matrix is stored in row wise systematic manner and a dummy element is inserted at the end of each row and a five digit index number has been given i.e., an index number of 10001 to indicate the end of first row, 10002 to indicate the end of second row and so on end an index number of 99999 has been given to indicate the end of the last row and also the matrix.

In the case of Newton-Raphson method, the matrix has been packed in both row and column wise systematic manner and in addition the linked list technique has been adopted for inversion and post multiplication with the vector.

CHAPTER IV

MATRIX FACTORIZATION

4.1 Introduction

The analysis of a large system using the network approach frequently necessitates the solution of hundreds and may be thousands of simultaneous equations having the form $AX = b$. Furthermore, several solutions are often required with the same coefficient matrix A but with a series of different b vectors. Such equations can be solved using any of the conventional and elementary methods. The solution also can be obtained by direct matrix inversion, but this requires n^2 storage locations for the coefficients and about n^3 arithmetic operations for the solution of n simultaneous linear equations (28). Even if the matrix 'A' is very sparse, its inverse is completely full and therefore this method is normally a very inefficient technique for solving a large number of equations.

We have already seen in chapter 3.6.1 the alternative to direct matrix inversion are the factorization techniques based on Gauss elimination. These are many possible factorization methods and adaptations. We will describe some of the important methods which have been developed over the past years.

4.2 Product Form of Inverse

In the product form of inverse the matrix A^{-1} is not calculated explicitly but is obtained by multiplying n factor matrices, i.e.,

$$A^{-1} = T_n \dots T_3 T_2 T_1$$

Each transformation matrix T_i ($i = 1, 2, \dots, n$) is a unit matrix except for its i -th column. Therefore, in digital computer solutions, only this i -th column need be stored; all other elements of the matrix are known implicitly. In general sparse network problems, the i -th column of T_i will also contain a large proportion of zero elements.

4.3 Triangulation of Matrices

Another effective and most widely used method of manipulating coefficient matrices to solve simultaneous linear equations is that associated with triangulation of matrices or triangular decomposition. These methods factorise the coefficient matrix into their triangular form on which several important and efficient modern techniques are based. The two methods which are discussed in this work are generally known as LH (or sometimes LU) and LDH (or LDU) methods.

4.3.1 LH Factorization

The LH method of factorization consists of expressing the coefficient matrix A as the product of two factor matrices such that

$$A = LH$$

where L = a lower triangular matrix

H = a higher (or upper) triangular matrix

which has unity elements on its diagonal.

4.3.2 LDH Factorization

In the case of LH factorization, the elements in the i -th column of 'L' are different to those in the i -th row of H. This means that both L and H must be known explicitly and both triangular matrices must be stored. This problem can be alleviated in the case of symmetrical coefficient matrix A by decomposing further the lower triangular matrix L. This method generally known as LDH factorization, expresses the original coefficient matrix A as a product of three factor matrices such that

$$A = L' DH$$

Where L' = a lower triangular matrix which has unity elements on its diagonal

H = a higher (or upper) triangular matrix which has unity elements on its diagonal

D = a diagonal matrix which has zero off-diagonal elements

4.4 Bi-factorization

This method has been adopted in the present work for obtaining the inverse in the case of Newton-Raphson method. This method is described in detail in this section.

The bi-factorization method should be used for sparse coefficient matrices that have non zero diagonal terms and are either strictly symmetric or asymmetric in element value but with a symmetric sparsity structure. Furthermore, it is assumed for reasons of round-off error

that the matrix is either symmetric and positive definite or is diagonally dominant (we say that a matrix is diagonally dominant by rows if each diagonal element is not less than the sum of the moduli of the elements in its row; a similar definition holds for diagonal dominance by columns).

In order to reduce computing time and to save storage, an optimally ordered pivotal sequence as well as a packed storage scheme and special programming techniques are essential.

4.4.1 Basic Computational Algorithm

A set of n linear equations can be expressed in matrix notation as

$$Ax = b \quad (1)$$

where A is a non-singular $n \times n$ coefficient matrix

x is a column vector of the n unknowns

and b is a known vector with at least one non-zero element.

In many practical applications the set of equations is to be solved for a series of different right-hand sides whereas A remains unchanged. The solution vector may then be computed directly from

$$x = A^{-1}b. \quad (2)$$

From the point of view of storage requirements and computation time it is not efficient to compute the inverse of A explicitly. This is particularly true for sparse matrices since it is unusual for their inverses to be other

than full. The bi-factorization method is based on the equation

$$L^{(n)} L^{(n-1)} \dots L^{(2)} L^{(1)} A R^{(1)} R^{(2)} \dots R^{(n-1)} R^{(n)} = I \quad (3)$$

where L are left-hand factor matrices,

R are right-hand factor matrices

and I is the unity matrix.

Equation (3) can be modified by simple transformations to

$$A^{-1} = R^{(1)} R^{(2)} \dots R^{(n-1)} R^{(n)} L^{(n)} L^{(n-1)} \dots L^{(2)} L^{(1)}. \quad (4)$$

Equation (4) shows that the inverse of A , in contrast to the familiar product form of the inverse, can also be expressed by a multiple product of $2n$ factor matrices.

In order to determine the factor matrices L and R the following sequence of intermediate matrices is introduced in equation (3):

$$A^{(0)} = A$$

$$A^{(1)} = L^{(1)} A^{(0)} R^{(1)}$$

$$A^{(2)} = L^{(2)} A^{(1)} R^{(2)}$$

.....

$$A^{(j)} = L^{(j)} A^{(j-1)} R^{(j)}$$

.....

$$A^{(n)} = L^{(n)} A^{(n-1)} R^{(n)} = I$$

This representation aims at transforming the initial coefficient matrix $A = A^{(0)}$ step by step to the unity matrix by forming the successive inner triple products $L^{(j)} A^{(j-1)} R^{(j)}$ ($j = 1 \dots n$).

4.4.2 Symmetrical Matrix A

For a symmetrical matrix A we have

$$a_{ik}^{(j-1)} = a_{ka}^{(j-1)}$$

and thus $r_{jk}^{(j)} = l_{ih}^{(j)}$ for $i = k \neq j$.

This means that because of symmetry the j th row of $R^{(j)}$ is identical to the j th column of $L^{(j)}$, except for the diagonal term. Therefore it is unnecessary to perform any operations to the right of the diagonal, thus saving about half of the reduction operations.

4.4.3 Asymmetrical Matrix A

In the case of an asymmetrical matrix A it is more advantageous from the computational point of view to further decompose each left-hand factor matrix L into a modified matrix C and a diagonal matrix D:

$$L^{(j)} = C^{(j)} D^{(j)}$$

The diagonal matrix $D^{(j)}$ differs from the unity matrix in only the j th diagonal term:

$$d_{jj}^{(j)} = 1 \quad a_{jj}^{(j-1)} = l_{jj}^{(j)}$$

The modified matrix $C^{(j)}$ differs, like $L^{(j)}$, from the unity matrix in only column j . This is column

$$\left[\begin{array}{c} 0 \dots 0 \\ 1 \\ c_{j+1,j}^{(j)} \\ c_{j+2,j}^{(j)} \dots c_{n,j}^{(j)} \end{array} \right]^T$$

where $c_{ij}^{(j)} = -a_{ij}^{(j-1)} = l_{ij}^{(j)} l_{jj}^{(j)}$ $i = (j+1) \dots n$.

4.4.4 Sparsity and Optimal Ordering

In case of sparse coefficient matrices, i.e. matrices with a great number of zeros, significant savings in storage and computation time can be obtained if a programming scheme is used which stores and processes only non-zero terms. Moreover, sparsity must be maintained as far as possible. This can be realised by a sparsity-directed pivotal selection which is referred to as "optimal ordering."

The objective of optimal ordering is to minimise the total number of fill-in terms. An optimum ordering strategy was developed by Carpentire and Canal. This strategy, however, requires relatively high efforts in additional programming and computation time so that in general a great deal of the obtainable advantages in sparsity get lost again.

Thus it is more advantageous to apply the following strategy which is frequently used in practice. This strategy yields only a near-optimal ordering sequence, but requires comparatively little additional computation. The principle of the strategy is to select at each step of the reduction process that column as pivot which contains that fewest number of non-zero terms. If more than one column meets this criterion, any one is selected. This scheme requires a current book-keeping of the number of non-zero terms in each column or row.

4.4.5 Storage Scheme

In order to exploit the benefits of sparsity, a packed matrix storage scheme in which only the non-zero terms are retained is employed. This requires, in addition to the matrix elements themselves, tables of indexing information to identify the elements and to facilitate their addressing.

A suitable storage scheme would be comparatively simple if the number of non-zero terms in one column did not vary in the course of computation. A difficulty arises, however because the number of non-zero terms in each column and row of the reduced matrix continually changes. The number of non-zero terms, on the one hand, is increased by the fill-in terms and, on the other hand, is decreased by the reduction process. For this reason a flexible storage mode is essential.

One feasible scheme for describing the symmetrical structure of a sparse matrix and identifying and addressing its elements in a packed table is described below. This scheme is somewhat different for the symmetrical case (symmetry in element value) and the asymmetrical case (asymmetry in element value but with a symmetric sparsity structure).

Symmetrical matrix

The non-zero matrix elements are stored columnwise in array CE. The row indices of the elements in CE are stored

in a parallel table ITAG. The accompanying table LNXT contains the location of the next non-zero element in CE in ascending order. The entry 0 in LNXT indicates the last term of a column.

The starting positions of the individual columns in CE are stored in table LCOL. The table NOZE contains the number of non zero elements in each column.

As can be seen from the example, the unused storage positions of the reserved arrays CE and LNXT also must be occupied by initial values. The vacant positions of array CE and the last position of table LNXT must be set zero. The other vacant positions of LNXT must be numbered consecutively.

Apart from this, the order of the matrix (number of columns and rows) is stored in N and the first vacant location in tables CE, ITAG and LNXT must be stored in LF.

Asymmetrical matrix

The storage mode of an asymmetrical matrix with a symmetric pattern of non-zero elements differs from the symmetrical case in two points. First, the diagonal terms are stored in a separate table DE. Second, the off-diagonal terms are stored in both directions, i.e. they are stored column-wise in CE and, in addition, row-wise in the parallel table RE. Because symmetry in structure is assumed, the table ITAG contains the row indices of the elements stored in CE as well as the column indices of the elements stored in RE. In case that a row or column has no off-

diagonal terms, i.e. it consists only in its diagonal-term (decoupled system), the respective position in table LCOL is to be set to zero.

The dual storage of the off-diagonal terms, in the first instance, might seem to be a waste of memory space. After having processed the simulation and ordering subroutine to be described in the next section, however, each off-diagonal terms is stored only once. The storage positions that become vacant in the course of computation are later utilised to store the fill-in terms. The advantage of dual storage is that it avoids use of a search subroutine and thus accelerates the program.

4.4.6. Programming

Programming is as important as the method itself. Optimal ordering can be determined during the course of computation, but it is more efficient to determine it by simulating the reduction process beforehand. Hence, the program can be split up into three parts:

1. Simulation and ordering
2. Reduction
3. Direct solution

In accordance with the different storage schemes for the symmetrical and asymmetrical case the programming is also somewhat different. Detailed flow charts are given in Appendix (for asymmetrical case only).

4.4.7 Simulation and Ordering

The optimal ordering process requires an additional table NSEQ. This table initially must contain the integer variables 1 to n in ascending sequence. At the end of the simulation process table NSEQ contains the pivotal sequence as it results from the applied ordering strategy.

Pivotal search

At first, among all columns which have not been pivotal column before, the column with the fewest number of non-zero elements is selected as pivotal column. If more than one column meets this criterion, the column number in the first location of table NSEQ is selected.

After having determined the pivotal index, no actual interchange of columns is carried out.

Instead, only the two respective indices within table NSEQ are interchanged such that the near-optimal pivotal sequence is built up step by step.

Indexing and Addressing Modification

All columns the index of which is contained in the pivotal column, are compared term by term with the pivotal column, and their accompanying indexing and addressing information is altered as follows:

If the processed column contains the pivotal index, the related matrix term is cancelled.

If any row index of the pivotal column is not contained in the column under consideration, this index is added to

the row indices in table ITAG (fill-in terms). The fill-in terms are stored not only in the vacant locations at the end of tables, CE, ITAG and LNXT but also in other locations becoming vacant in the course of the simulation process. The next vacant location is always indicated by LF.

Whenever a term is cancelled or added, the respective addressing information in LNXT and LCOL respectively must be altered appropriately. Furthermore, the bookkeeping of non-zero terms must be updated.

After processing the simulation and ordering subprogram, the tables LCOL, NOZE, NSEQ, ITAG and LNXT no longer contain the information on the structure of the original coefficient matrix, but contain instead the structure of the factor matrices.

4.4.8. Reduction

The reduction subprogram operates upon the storage image resulting from the simulation and ordering subprogram. The actual reduction of the coefficient matrix is guided by the pivotal sequence contained in table NSEQ. At each stage of the reduction process only those terms of the reduced residual matrix with subscripts corresponding to the row indices of the pivotal column have to be recalculated. For that purpose the corresponding columns are compared term by term with the pivotal column in much the same way as in the simulation and ordering subprogram.

Every derived term of the factor matrices is left in

the position of the corresponding term of the coefficient matrix.

In the symmetrical case, at the beginning of each reduction step the terms of the pivotal column are temporarily stored in vacant positions of table CE. This permits normalisation of the pivotal column, which means multiplying the pivotal column by the reciprocal of its diagonal term in the course of the reduction process.

Intermediate storage of the pivotal column is not necessary in the asymmetrical case because the pivotal column as well as the pivotal row is stored in CE and RE respectively, and the pivotal column has not to be normalised.

4.4.9. Direct Solution

The given vector must initially be stored in V. Then it is stepwise transformed to the solution vector by successive factor-matrix by-vector multiplications.

After having processed the direct solution subprogram Table V contains the solution.

The total number of arithmetical operations (multiplications and additions) for computing the direct solution in the bi-factorization method is the same as in the triangular decomposition method. An important advantage of the bi-factorization method, however, is realized in programming, because the symmetric structure of the coefficient matrix can be completely exploited.

The bi-factorisation method requires only half as much indexing information as the triangular decomposition method unless a search subroutine is applied.

The main characteristics of the bi-factorisation method and the programming scheme can be summarised as follows:

- (a) The method allows repeated solutions for different right-hand sides without repeating the reduction process.
- (b) Small memory requirements and short computation time can be realised, because only non-zero matrix terms are stored and processed.
- (c) The pivotal selection procedure requires only little additional computation.
- (d) Symmetry can entirely be exploited in programming for matrices having a symmetric pattern of non-zero terms.
- (e) The applied storage and programming scheme does not require any index renumbering nor rearrangement of matrix terms according to the ascertained pivotal sequence.

In this work we have used the bi-factorisation method for obtaining the solution by Newton-Raphson method. The Jacobian is symmetric in structure but asymmetric in element value. Hence the storage scheme as described for asymmetric case has been adopted.

CHAPTER - VALGORITHMS AND FLOW CHARTSGauss-Seidel Method

As already explained in the chapter III the Y-Bus has been packed in the row-wise form and since the Y-Bus is symmetric, only the upper triangular portion is stored.

6.1 Read the System Data

The data is read in the following manner:

- (a) Number of buses, number of lines, number of voltage controlled buses and number of tap changing transformers.
- (b) Acceleration factor.
- (c) Line number, starting bus of the line, ending bus of the line, line impedance in p.u. and half line charging susceptance in p.u.
- (d) Starting bus number, ending bus number, transformer reactance, transformer tapsetting and transformer number.
- (e) Bus number, initial bus voltage (in case of voltage controlled buses the voltage magnitude to be maintained is assumed as the starting voltage and the phase angle zero), real and reactive power generation at the bus, real and reactive power demand at the bus.
- (f) Voltage controlled bus number, and magnitude of the voltage to be maintained at the bus for all the voltage controlled buses.
- (g) Generator bus number, minimum reactive power limit and maximum reactive power limit.

176980

6.2 Assembly of Data

- (a) From the tap changing transformer data calculate the equivalent series impedance.
- (b) Arrange the initial bus number and ending bus number of line data such that the smaller bus number is the starting bus and the larger bus number is the ending bus.
- (c) Now arrange the data in the ascending order of initial bus numbers.
- (d) In case where there are more than one line with the same initial bus number arrange the data in the ascending order of ending bus numbers.

6.3 Form Y-Bus

- (a) Calculate the diagonal elements which are the algebraic sum of all admittances incident to a node.
- (b) The off diagonal elements are obtained as the negative of the admittance connected between the nodes.
- (c) Since the data is arranged in the required manner the elements of the Y-Bus are calculated row wise and also the location of the column is given by the ending bus number.
- (d) Whenever the initial bus number changes the diagonal element is inserted and appropriate column number given as explained previously, i.e. the first diagonal element is given the number 10001, the second diagonal element, is given the number 10002 and so on.

- (e) If there are no off diagonal elements in the row the next diagonal element is inserted and appropriate column number given.
- (f) The last diagonal element which is also the last element of the matrix is a given an index number of 99999.

6.4 Iterative Computation of Voltages

The iterative algorithm used is as per equation 11 of chapter II. It is necessary to begin the iterations with an initial guess, and since we know that in a real system the voltage spread will not be too great, it is customary to use a "flat voltage start", meaning that we set initially all voltages except at the voltage controlled buses equal to the specified slack bus voltage V_1 for example, $1 + j0$ p.u.

In the equation 11 of chapter II quantities Y_{pp} , P_p , Q_p and Y_{pq} do not change through the iteration process.

So the equation can be rewritten:

$$V_p^{v+1} = A_p / \left(\begin{matrix} V_p \\ V_p \end{matrix} \right)^* - \frac{1}{Y_{pp}} \left(\begin{matrix} p-1 \\ \sum_{\mu=1}^{p-1} Y_p V_{\mu}^{v+1} - \sum_{\mu=p+1}^n Y_p V_{\mu}^v \end{matrix} \right)$$

for $p = 2, 3, \dots, n$ (assuming first bus as reference)

Here to obtain the summation of the product of $Y_p V$

which essentially consists of multiplying the each row of the admittance matrix by the vector of bus voltages. Systematic row wise packing helps in this process. The lower triangular elements are built up by scanning the matrix.

6.5 Voltage Controlled Buses

It is convenient to group all the voltage controlled buses at one place as it will simplify the programming. The following conditions must be satisfied for the voltage controlled buses.

Condition 1 The voltages V must satisfy, the specified requirements

$$V_p = V_{p \text{ spec}}$$

Condition 2 We must under no circumstances violate the requirement

$$Q_{p, \min} < Q_p < Q_{p, \max}$$

The second requirement may be violated if the specified voltage magnitude $V_{p \text{ spec}}$ is either too low or or too high. We remember that the only means of controlling V_p at our disposal is the reactive power Q_p and since we a priori do not know exactly how much reactice power is needed to reach the specified voltage, it may conceivably happen that we have specified a V_p value beyond the capability of the Q_p source.

The following are steps used for the voltage controlled buses:

- (a) Having identified the bus as a voltage controlled bus, we immediately make the temporary voltage magnitude substitution by the specified voltage but the phase angle is kept as is

- (b) We now compute the reactive bus power needed to maintain the voltage magnitude specified. This computation is based upon voltage magnitude specified.
- (c) Now we compare the magnitude of the Q computed with the allowable limits. If the computed ' Q ' is within the limits, it means the voltage can be kept at the specified value. If the computed ' Q ' value is beyond the ' Q ' limits, we can check the magnitude of ' Q ' for the the next few iterations. If the ' Q ' limits are not satisfied in the next few iteration also it means the voltage cannot be maintained at the specified voltage. In this case the solution is not the desired solution and we have to raise the ' Q ' limits suitably in order to maintain the voltage specified.

6.6 Test for Convergence

The iterative process must continue until the magnitude of the change of the bus voltage, $|\Delta V_p^{v+1}|$ between two consecutive iterations is less than a certain tolerance level for all bus voltages. We express this in mathematical form as follows:

$$|\Delta V_p^{v+1}| = |V_p^{v+1} - V_p^v| < \epsilon$$

In order to do that a dummy variable ΔV_{\max} , is introduced. Whenever we start the bus count, this variable is reset to zero. Upon completion of the bus count, this variable tells us the largest $|\Delta V_p^{v+1}|$ value that has been recorded for any bus.

At the end of the bus count, should V_{\max} not fall below the tolerance value , then a new iteration cycle is initiated.

6.7 Computation of Slack Bus Power

After the iterations have converged, we substitute our computed voltages (plus V_1) into equation (15) of chapter II to obtain slack bus power.

6.8 Computation of Line Flows

The line flows are calculated using equations (6 & 7) of chapter II. The flow chart is attached in Appendix A.2.1.

7.1 Newton-Raphson Method

These steps of reading and assembly of data are exactly similar to Gauss-Seidel method (6.1 & 6.2).

7.2 Formation of Y-Bus

In this method the full Y-Bus is formed and stored. The Y-Bus is stored in the systematic row-wise manner. To indicate the end of the row a dummy element of value 0.0 is introduced and a five digit index number given. The end of the matrix is indicated by an index number of 99999 .

Y-Bus is formed in exactly the same way as explained in the previous method. The only change being:

- (a) Whenever the initial bus number changes a dummy element of 0.0 is inserted and an appropriate index number is given.
- (b) After the last element of the matrix a dummy element with

an index number 99999 is inserted.

7.3 Calculation of Jacobian Matrix

The jacobian matrix is again a sparse matrix which is symmetric in structure but the values are asymmetric. The jacobian is stored in the manner described in chapter IV.

The elements are calculated as per the formulas (equation numbers, 19-22 of chapter II). The elements are also calculated row wise. The necessary information like the, number of non-zero elements in each row, starting position of columns, location of next term, sequence of pivotal indices, and row index of the elements stored are also calculated simultaneously. The bus powers are also computed based on the assumed bus voltages. The elements are calculated depending whether the bus is a voltage controlled or load bus.

7.4 Calculation of Voltage Corrections

The difference in computed bus powers and actual bus powers is calculated. If the difference is less than a specified tolerance the iterative process is stopped and the line flows calculated.

If the difference is more than the specified tolerance, the changes required in the bus voltage magnitude and phase angle are calculated by inversion of the jacobian and multiplication with the vector of differences of computed and actual bus powers.

The sub routine INV1 once for all determines the optimum

pivotal ordering strategy as the structure of the jacobian remains unchanged with iterations.

If the convergence is not obtained only the new elements of the jacobian are calculated based on the corrected voltage magnitude and phase angle in subsequent iterations.

The flow chart for Load flow solution by Newton-Raphson method is enclosed in Appendix (A.2.2).

CHAPTER - VITEST PROBLEMS AND RESULTS

The problems studied along with the data and results are appended below:

6.1 Problem No. 1:

A single line diagram of a 5 bus power system is shown in Appendix A.2.9. With bus one as the slack, and the remaining buses as load buses find the load flow solution. The line data is given in the Table 6.1 and Table 6.2 gives the data of scheduled generation and loads at the buses.

Results of the load flow study are given in TableNos. 6.3, 6.4 and 6.5. Per unit values are on 100 MVA base.

Table 6.1

Line No.	Between Buses	Line Impedance		Half of Line charging susceptance (p.u)
		R per unit	X per unit	
1	1-2	0.02	0.06	0.030
2	1-2	0.08	0.24	0.025
3	2-3	0.06	0.18	0.020
4	2-4	0.06	0.18	0.020
5	2-5	0.04	0.12	0.015
6	3-4	0.01	0.03	0.010
7	4-5	0.08	0.24	0.025

Table 6.2

Bus No.	Assumed Bus Voltage	Generation		Load	
		Megawatts	Megavars	Megawatts	Megavars
1	1.06 + j0.0	0	0	0	0
2	1.0 + j0.0	40	30	20	10
3	1.0 + j0.0	0	0	45	15
4	1.0 + j0.0	0	0	40	5
5	1.0 + j0.0	0	0	60	10

Table 6.3

Bus	Voltage Magnitude	Phaseangle (Deg)	Real Power	Reactive Power
1	1.06000	0.00000	1.2930	-0.0751
2	1.04751	-2.80065	0.2000	0.2000
3	1.02479	-4.98709	-0.4500	-0.1500
4	1.02367	-5.32044	-0.4000	-0.0500
5	1.01802	-6.14372	-0.6000	-0.1000

Table 6.4Line Flow

Line	SB	EB	Real Power	Reactive Power
1	2	1	-0.8726	0.0619
1	1	2	0.8866	-0.0864
2	3	1	-0.3945	-0.0300
2	1	3	0.4064	0.0113
3	3	2	-0.2430	-0.0678
3	2	3	0.2465	0.03540
4	4	2	-0.2747	-0.0592
4	2	4	0.2791	0.0295
5	5	2	-0.5369	-0.0716
5	2	5	0.5482	0.0734
6	4	3	-0.1892	0.0321
6	3	4	0.1895	-0.0519
7	5	4	-0.0632	-0.0284
7	4	5	0.0635	-0.0228

Table 6.5

Sl No	Method	Acceleration factor	No. of C.P.U. iterations	Time (in Sec.)
1	Gauss-Seidel	1.0	23	0.31
	"	1.2	16	0.27
2	Newton-Raphson	1.0	2	0.57

6.2 Problem No. 2:

A single line diagram of a 8 bus power system is shown in figure A.2.8. With bus 1 as reference, buses 2,3,4 as voltage controlled buses and the balance as load buses, find out the load flow solution. Table 6.6 gives the line data and Table 6.7 gives the data of scheduled generation and loads at the buses.

Also study the effect of varying the acceleration factor on the number of iterations and c.p.u. time, in the case of Gauss-Seidel method.

Results of load flow study are given in Table Nos. 6.8, 6.9 and 6.10.

Table 6.6

Line No.	Between Buses	Line Impedance		Line charging susceptance (p.u.)
		R per unit	X per unit	
1	1-2	0.010	0.070	0.05
2	1-6	0.002	0.010	0.0
3	1-5	0.003	0.300	0.0
4	1-4	0.008	0.065	0.03
5	4-5	0.0035	0.020	0.0
6	3-4	0.0075	0.063	0.06
7	8-3	0.001	0.015	0.0
8	7-3	0.0025	0.023	0.0
9	2-3	0.001	0.081	0.08
10	2-7	0.0032	0.030	0.0

Cont..... Table 6.6

11	6-7	0.0021	0.01	0.0
12	6-5	0.002	0.013	0.0
13	5-8	0.0016	0.021	0.0
14	7-8	0.0021	0.0311	0.0

Table 6.7

BUS	Bus power (in m. w.)	Voltage	Q_{\min}	Q_{\max}
			(in m.v.e.r.)	
1	Unspec.	1.0+j0.0		
2	-23.30 + j(unspec)	$V_1 = 1.0$	-10.00	10.00
3	15.00 + j(unspec)	$V_2 = 1.0$	-10.00	10.00
4	15.00 + j(unspec)	$V_3 = 1.0$	-10.00	10.00
5	25.00 + j20.00	unspec.		
6	-22.00 - j13.00	unspec.		
7	25.00 + j00.00	unspec.		
8	00.00 + j10.00	unspec.		

Table 6.8

BUS	Voltage Magnitude	Phase Angle (DEG)	Real Power	Reactive Power
1	1.00000	0.00000	3.3998	5.6903
2	1.00001	-17.90487	-23.3000	9.7992
3	1.00001	10.70021	15.0000	6.9128
4	1.00001	-8.10451	-20.0000	2.8639
5	1.05722	6.27096	25.0000	20.0000
6	.92760	0.36610	-22.0000	-13.0000
7	.95400	8.54623	25.0000	0.0000
8	.93501	9.04251	0.0000	-10.0000

Table 6.9

LINE FLOW

Line	SB	EB	Real Power	Reactive Power
1	2	1	-4.2073	1.2680
1	1	2	4.4010	0.0381
2	4	1	-2.1179	0.3995
2	1	4	2.1552	-0.1267
3	5	1	4.0318	1.8242
3	1	5	-3.9792	-1.2985
4	6	1	-0.7213	-6.5698
4	1	6	0.8228	7.0775
5	3	2	6.0047	0.6515
5	2	3	-5.6029	2.2278
6	7	2	14.2024	0.3511
6	2	7	-13.4927	6.3018
7	4	3	-4.9457	1.4061
7	3	4	5.1446	0.2048

Contd...

1	2	3	4	5
8	7	3	-1.7426	-1.6895
8	3	7	1.7588	1.8384
9	8	3	-2.0624	-3.8882
9	3	8	2.0846	4.2206
10	5	4	13.5288	2.3118
10	4	5	-12.9389	1.0589
11	6	5	-8.9111	-7.4778
11	5	6	9.2257	9.5223
12	8	5	1.8549	-5.5277
12	5	8	-1.7927	6.3444
13	7	6	12.7472	0.7423
13	6	7	-12.3710	1.0491
14	8	7	0.2090	-0.5841
14	7	8	-0.2081	0.5978

Table 6.10

Sl No.	Method	Acceleration factor	No. of iterations	CPU Time (in Secs)
1	Gauss-Seidel	1.0	26	0.67
		1.1	22	0.59
		1.2	18	0.53
		1.25	17	0.53
		1.3	15	0.52
		1.4	13	0.47
		1.7	29	0.69
2	Newton-Raphson	1.0	3	0.83

6.3 Problem No. 3

IEEE standard 57 bus problem has been taken. The diagram in Appendix A. 2.6. The data has been given in table numbers A.4.1 to A.4.5. The last bus (57th bus) has been taken as the reference bus. Buses from 51 to 56 are the voltage controlled bus and the rest are load buses. The results of load flow study are given in Table 6.11 and 6.12.

IEEE 57-BUS TEST SYSTEM

TABLE A.4.1 Impedance and Line-charging Data

Line Designation	Resistance p.u.*	Reactance p.u.*	Line charging** p.u.*
1	2	3	4
57-56	0.0083	0.028	0.0645
56-55	0.0298	0.085	0.0409
55-4	0.0112	0.0366	0.0190
4-5	0.0625	0.1320	0.0129
4-54	0.0430	0.1480	0.0174
54-7	0.0200	0.1020	0.0138
54-53	0.0339	0.1730	0.0235
53-52	0.0099	0.0505	0.0274
52-10	0.0369	0.1679	0.0220
52-11	0.0258	0.0848	0.0109
52-51	0.0648	0.2950	0.0386
52-13	0.0481	0.1580	0.0203
13-14	0.0132	0.0434	0.0055
13-15	0.0269	0.0869	0.0115
57-15	0.0178	0.0910	0.0494
57-16	0.0454	0.2060	0.0273
57-17	0.0238	0.1080	0.0143
55-15	0.0162	0.0530	0.0272
4-18	0.0000	0.5550	0.0000
4-18	0.0000	0.4300	0.0000
5-54	0.0302	0.0641	0.0062
7-53	0.0139	0.0712	0.0097

contd.

TABLE A.4.1 contd.

1	2	3	4
10-51	0.0277	0.1262	0.0164
11-13	0.0223	0.0732	0.0094
51-13	0.0178	0.0580	0.0302
51-16	0.0180	0.0813	0.0108
51-17	0.0397	0.1790	0.0238
14-15	0.0171	0.0547	0.0074
18-19	0.4610	0.6850	0.0000
19-20	0.2830	0.4340	0.0000
20-21	0.0000	0.7767	0.0000
21-22	0.0736	0.1170	0.0000
22-23	0.0099	0.0152	0.0000
23-24	0.1660	0.2560	0.0042
24-25	0.0000	1.1820	0.0000
24-25	0.0000	1.2300	0.0000
24-26	0.0000	0.0473	0.0000
26-27	0.1650	0.2540	0.0000
27-28	0.0618	0.0954	0.0000
28-29	0.0418	0.0587	0.0000
7-29	0.0000	0.0648	0.0000
25-30	0.1350	0.2020	0.0000
30-31	0.3260	0.4970	0.0000
31-32	0.5070	0.7550	0.0000
32-33	0.0392	0.0360	0.0000
32-34	0.0000	0.9530	0.0000

contd.

TABLE A.4.1 contd.

1	2	3	4
34-35	0.0520	0.0780	0.0016
35-36	0.0430	0.0537	0.0008
36-37	0.0290	0.0366	0.0000
37-38	0.0651	0.1009	0.0010
37-39	0.0239	0.0379	0.0000
36-40	0.0300	0.0466	0.0000
22-38	0.0192	0.0295	0.0000
11-41	0.0000	0.7490	0.0000
41-42	0.2070	0.3520	0.0000
41-43	0.0000	0.4120	0.0000
38-44	0.0289	0.0585	0.0010
15-45	0.0000	0.1042	0.0000
14-46	0.0000	0.0735	0.0000
46-47	0.0230	0.0680	0.0016
47-48	0.0182	0.0233	0.0000
48-49	0.0834	0.1290	0.0024
49-50	0.0801	0.1280	0.0000
50-12	0.1386	0.2200	0.0000
10-12	0.0000	0.0712	0.0000
13-49	0.0000	0.1910	0.0000
29- 9	0.1442	0.1870	0.0000
9- 8	0.0762	0.0984	0.0000
8- 6	0.1878	0.2320	0.0000
6- 3	0.1732	0.2265	0.0000
11-43	0.0000	0.1530	0.0000

contd.

TABLE A.4.1 contd.

1	2	3	4
44-45	0.0624	0.1242	0.0020
40- 2	0.0000	1.1950	0.0000
2-41	0.5530	0.5490	0.0000
2-42	0.2125	0.3540	0.0000
39- 1	0.0000	1.3550	0.0000
1- 2	0.1740	0.2600	0.0000
38-49	0.1150	0.1770	0.0030
38-48	0.0312	0.0482	0.0000
52- 3	0.0000	0.1205	0.0000

* Base MVA = 100

** Line charging : One-half of total charging of line

TABLE A.4.2 Regulated Bus Data

Bus Number	Voltage Magnitude p.u.	Min.MVAR Capability	Max MVAR Capability
51	1.015	-50	155
52	0.980	- 3	9
53	1.005	-140	200
54	0.980	- 8	25
55	0.985	-10	60
56	1.010	-17	50

TABLE A.4.3 Transformer Data

Transformer Designation	Tap Setting
4-18	0.970
4-18	0.978
7-29	0.967
52- 3	0.940
10-12	0.930
11-41	0.955
11-43	0.958
13-49	0.895
14-46	0.900
15-45	0.955
21-20	1.043
24-25	1.000
24-25	1.000
24-26	1.043
34-32	0.975
39- 1	0.980
40- 2	0.958

TABLE A.4.4 Static Capacitor Data

Bus Number	Susceptance p.u.*
18	0.100
25	0.059
8	0.063

TABLE A.4.5 Generation and Load Data

Bus Number	Real power generation p.u. *	Load	
		Real p.u. *	Reactive p.u. *
1	2	3	4
1.	0.0	0.067	0.020
2	0.0	0.076	0.022
3	0.0	0.068	0.034
4	0.0	0.000	0.000
5	0.0	0.130	0.040
6	0.0	0.041	0.014
7	0.0	0.000	0.000
8	0.0	0.200	0.100
9	0.0	0.049	0.022
10	0.0	0.050	0.020
11	0.0	0.000	0.000
12	0.0	0.180	0.053
13	0.0	0.180	0.023
14	0.0	0.105	0.053
15	0.0	0.220	0.050
16	0.0	0.430	0.030
17	0.0	0.420	0.080
18	0.0	0.272	0.098
19	0.0	0.033	0.006
20	0.0	0.023	0.010
21	0.0	0.000	0.000
22	0.0	0.000	0.000
23	0.0	0.063	0.021
24	0.0	0.000	0.000
25	0.0	0.063	0.022
26	0.0	0.000	0.000
27	0.0	0.093	0.005
28	0.0	0.046	0.023
29	0.0	0.170	0.026
30	0.0	0.036	0.018

contd.

TABLE A.4.5 contd.

1	2	3	4
31	0.0	0.058	0.029
32	0.0	0.016	0.008
33	0.0	0.380	0.019
34	0.0	0.000	0.000
35	0.0	0.060	0.030
36	0.0	0.000	0.000
37	0.0	0.000	0.000
38	0.0	0.140	0.070
39	0.0	0.000	0.000
40	0.0	0.000	0.000
41	0.0	0.063	0.030
42	0.0	0.071	0.044
43	0.0	0.020	0.010
44	0.0	0.120	0.018
45	0.0	0.000	0.000
46	0.0	0.000	0.000
47	0.0	0.297	0.116
48	0.0	0.000	0.000
49	0.0	0.180	0.085
50	0.0	0.210	0.105
51	3.1	3.770	0.240
52	0.0	1.210	0.260
53	4.5	1.500	0.220
54	0.0	0.750	0.020
55	0.4	0.410	0.210
56	0.0	0.030	0.880
57	slack bus	0.550	0.770

*Base MVA = 100

Table 6.11

Voltage Magnitude	Phase Angle (DEG)	Real Power	Reac Power
.87539	-17.42966	-0.0670	-0.0200
.88380	-16.81561	-0.0760	-0.0220
.97047	-10.88694	-0.0680	-0.0340
.98051	- 7.37273	0.0000	0.0000
.97640	- 8.60529	-0.1300	-0.0400
.94157	-12.00492	-0.0410	-0.0140
.98248	- 7.69161	0.0000	0.0000
.92267	-12.73050	-0.2000	-0.0370
.93549	-11.92716	-0.0490	-0.0220
.98348	-11.49845	-0.0500	-0.0200
.97310	-10.19551	0.0000	0.0000
.97233	-12.84201	-0.1800	-0.0530
.98170	- 9.79313	-0.1800	-0.0230
.97407	- 9.31332	-0.1050	-0.0530
.98749	- 7.17752	-0.2200	-0.0500
1.01337	- 8.85126	-0.4300	-0.0300
1.01746	- 5.39189	-0.4200	-0.0800
.97265	-12.07298	-0.2720	0.0020
.93479	-13.66450	-0.0330	-0.0060
.92430	-13.90093	-0.0230	-0.0100
.92271	-13.53585	0.0000	0.0000
.92305	-13.47369	0.0000	0.0000
.92151	-13.55081	-0.0630	-0.0210
.91188	-13.94757	0.0000	0.0000

contd...

2	3	4	5
.89018	-19.93265	-0.0640	0.0270
.91313	-13.58858	9.0000	0.0000
.94029	-11.98298	-0.0930	-0.0050
.95741	-10.86365	-0.0460	-0.0230
.97238	-10.10033	-0.1700	-0.0260
.86635	-20.60745	-0.0360	-0.0180
.83242	-21.45155	-0.0580	-0.0180
.84223	-20.36584	-0.0160	-0.0290
.83966	-20.41810	0.0380	-0.0190
.87659	-15.08632	0.0000	0.0000
.88384	-14.80767	-0.0600	-0.0300
.89408	-14.49347	0.0000	0.0000
.90216	-14.23138	0.0000	0.0000
.92614	-13.31517	-0.1400	-0.0700
.90058	-14.28833	0.0000	0.0000
.89284	-14.56610	0.0000	0.0000
.93345	-14.89041	-0.0630	-0.0300
.89160	-16.38939	-0.0710	-0.0440
.96057	-11.56728	0.0200	-0.0100
.93835	-12.39412	-0.1200	-0.0180
.97591	-9.66806	0.0000	0.0000
.96055	-11.27404	0.0000	0.0000
.93842	-12.90833	-0.2970	-0.1160
.93487	-13.06525	0.0000	0.0000
.94055	-13.29973	-0.1800	-0.0850
.93164	-13.88064	-0.2100	-0.1050

contd...

1	2	3	4	5	6
51	1.01499	-10.46=38	-0.6700	1.0203	
52	.97999	-9.59779	-1.2100	0.2558	
53	1.0000	-4.53207	3.0000		
	1.00499	-4.53207	3.0000	0.4248	
54	.97999	-8.74506	-0.7500	0.0094	
55	.98499	-6.00693	-0.0100	-0.2015	
56	1.01000	-1.19273	-0.0300	-0.8874	
57	1.04000	0.00000	4.2361	1.1241	

Table 6.12

Sl No	Method	Acc factor	No. of iterations	CPU time (in secs.)
1.	GAUSS-SEIDAL	1.7	46	4.42
		1.5	88	6.89
2.	NEWTON-RAPHSON	1.0	3	2.73

6.4 Problem No. 4

IEEE Standard¹¹⁸ bus problem was taken. The diagram is in Appendix A.2.7. The data has been given in table numbers A.5.1 to A.5.5. Bus No. 1 is taken as reference bus and Buses from 2 to 54 are voltage controlled buses. Buses from 56 to 118 are load buses. The results of the load flow study are given in Table number 6.13 and 6.14.

IEEE 118-BUS TEST SYSTEM

TABLE A.5.1 Impedance and Line-charging Data

Line Designation	Resistance p.u. *	Reactance p.u. *	Line Charging p.u. *	**
1	2	3	4	
30-55	0.0303	0.0999	0.0063	
30-56	0.0129	0.0424	0.0027	
55- 6	0.0187	0.0616	0.0039	
56-57	0.0241	0.1080	0.0071	
56- 6	0.0484	0.1600	0.0101	
2-60	0.0209	0.0688	0.0043	
2-57	0.0018	0.0080	0.0005	
57-60	0.0203	0.0682	0.0043	
57- 3	0.0119	0.0540	0.0035	
3-58	0.0045	0.0208	0.0013	
58- 6	0.0086	0.0340	0.0021	
4-71	0.0043	0.0504	0.1285	
4-59	0.0024	0.0305	0.2905	
59- 5	0.0026	0.0322	0.3075	
60- 6	0.0059	0.0196	0.0012	
60-61	0.0222	0.0731	0.0047	
6-63	0.0212	0.0834	0.0053	
6-117	0.0329	0.1400	0.0089	
6-62	0.0215	0.0707	0.0045	
61- 7	0.0744	0.2444	0.0156	
62- 7	0.0595	0.1950	0.0125	
7-64	0.0132	0.0437	0.0111	contd.

TABLE 5.1 contd.

1	2	3	4
7- 9	0.0120	0.0394	0.0025
7-72	0.0380	0.1244	0.0080
63-64	0.0454	0.1801	0.0116
64-53	0.0091	0.0301	0.0019
64- 8	0.0123	0.0505	0.0032
64-14	0.0474	0.1563	0.0100
8- 9	0.0111	0.0493	0.0028
9-65	0.0252	0.1170	0.0074
9-16	0.0752	0.2470	0.0158
65-66	0.0183	0.0849	0.0054
66-67	0.0209	0.0970	0.0061
67-68	0.0342	0.1590	0.0101
68-15	0.0317	0.1153	0.0293
68-10	0.0135	0.0492	0.0124
68-11	0.0156	0.0800	0.0216
10-31	0.1022	0.4115	0.0255
10-32	0.0488	0.1960	0.0122
11-13	0.0318	0.1630	0.0441
12-71	0.0079	0.0860	0.2270
13-15	0.0229	0.0755	0.0048
13-116	0.0164	0.0741	0.0049
13-69	0.0191	0.0855	0.0054
69-70	0.0237	0.0943	0.0059
70-14	0.0108	0.0331	0.0020
71-75	0.0046	0.0540	0.1055
14-15	0.0298	0.0985	0.0062

contd.

TABLE 5.1 contd.

1	2	3	4	
15-53	0.0615	0.2030	0.0129	
15-115	0.0135	0.0612	0.0040	
72-74	0.0415	0.1420	0.0091	
16-17	0.0087	0.0268	0.0014	
16-74	0.0026	0.0094	0.0024	
16-78	0.0413	0.1681	0.0105	
73-17	0.0022	0.0102	0.0006	
73-74	0.0110	0.0497	0.0033	
74-76	0.0321	0.1060	0.0067	
74-18	0.0593	0.1680	0.0105	
75-28	0.0090	0.0986	0.2615	
76-18	0.0184	0.0605	0.0038	
18-77	0.0145	0.0487	0.0030	
18-19	0.0555	0.1830	0.0116	
77-19	0.0410	0.1350	0.0086	
19-21	0.0358	0.1610	0.0430	
78-79	0.0608	0.2454	0.0151	
79-80	0.0224	0.0901	0.0056	
80-20	0.0400	0.1356	0.0083	
80-21	0.0684	0.1860	0.0111	
20-81	0.0380	0.1270	0.0079	
20-82	0.0601	0.1890	0.0118	
81-21	0.0191	0.0625	0.0040	
81- 1	0.0844	0.2778	0.0177	
82-21	0.0179	0.0105	0.0031	
21-83	0.0267	0.0752	0.0046	contd.

TABLE 5.1 contd.

1	2	3	4
21-84	0.0486	0.1370	0.0085
21-29	0.0090	0.0459	0.0124
21- 1	0.0985	0.3240	0.0207
21-22	0.0398	0.1450	0.0367
83-87	0.0474	0.1340	0.0083
84-85	0.0203	0.0588	0.0035
84-88	0.0255	0.0719	0.0044
85-86	0.0405	0.1635	0.0101
86-22	0.0263	0.1220	0.0077
22-23	0.0169	0.0707	0.0050
22-24	0.0027	0.0095	0.0018
22-25	0.0503	0.2293	0.0149
23-24	0.0048	0.0151	0.0009
23-25	0.0473	0.2158	0.0141
24-87	0.0343	0.0966	0.0060
24-88	0.0343	0.0966	0.0060
24-25	0.0407	0.1200	0.0276
25-89	0.0317	0.1450	0.0094
25-26	0.0328	0.1500	0.0097
89-26	0.0026	0.0135	0.0036
89-27	0.0123	0.0561	0.0036
26-27	0.0082	0.0376	0.0024
27-29	0.0482	0.2180	0.0144
27-92	0.0258	0.1170	0.0077
90-91	0.0017	0.0200	0.0540
91-28	0.0027	0.0302	0.0950

contd.

TABLE 5.1 contd.

1	2	3	4
28-93	0.0014	0.0160	0.1595
29-92	0.0224	0.1015	0.0067
93-54	0.0003	0.0040	0.0410
93-98	0.0017	0.0202	0.2020
1-95	0.0405	0.1220	0.0310
1-36	0.0309	0.1010	0.0259
1-31	0.0300	0.1270	0.0305
31-94	0.0088	0.0355	0.0021
31-34	0.0401	0.1323	0.0084
31-95	0.0428	0.1410	0.0090
94-32	0.0446	0.1800	0.0111
94-33	0.0087	0.0454	0.0029
34-95	0.0123	0.0406	0.0025
95-118	0.0145	0.0481	0.0029
95-36	0.0601	0.1999	0.0124
35-118	0.0164	0.0544	0.0034
35-36	0.0444	0.1480	0.0092
36-96	0.0037	0.0124	0.0031
36-37	0.0108	0.0331	0.0175
36-99	0.0298	0.0853	0.0204
96-97	0.0054	0.0244	0.0016
97-37	0.0156	0.0704	0.0046
37-107	0.0356	0.1820	0.0123
37-108	0.0183	0.0934	0.0063
37-109	0.0238	0.1080	0.0071

contd.

TABLE 5.1 contd.

1	2	3	4	
37-44	0.0454	0.2060	0.0136	
99-107	0.0162	0.0530	0.0136	
99-100	0.0112	0.0366	0.0095	
100-101	0.0625	0.1320	0.0064	
100-38	0.0430	0.1480	0.0087	
101-38	0.0302	0.0641	0.0030	
38-102	0.0350	0.1230	0.0069	
38-103	0.0200	0.1020	0.0069	
38-40	0.0239	0.1730	0.0117	
102-39	0.0282	0.2074	0.0111	
103-40	0.0139	0.0712	0.0048	
40-41	0.0158	0.0653	0.0397	
40-43	0.0079	0.0380	0.0240	
41-42	0.0254	0.0836	0.0053	
42-43	0.0387	0.1272	0.0081	
43-104	0.0258	0.0848	0.0054	
43-105	0.0481	0.1580	0.0101	
43-45	0.0648	0.2950	0.0193	
43-111	0.0123	0.0559	0.0036	
104-105	0.0223	0.0732	0.0047	
105-106	0.0132	0.0434	0.0027	
105-107	0.0269	0.0869	0.0057	
105-45	0.0178	0.0580	0.0151	
106-107	0.0171	0.0547	0.0037	
107-108	0.0173	0.0885	0.0060	contd.

TABLE 5.1 contd.

1	2	3	4
109-45	0.0397	0.1790	0.0119
44-45	0.0180	0.0813	0.0054
45-110	0.0277	0.1262	0.0082
45-46	0.0160	0.0525	0.0134
45-47	0.0451	0.2040	0.0135
45-112	0.0605	0.2290	0.0155
110-111	0.0246	0.1120	0.0073
46-50	0.0391	0.1813	0.0115
46-47	0.0466	0.1584	0.0101
46-48	0.0535	0.1625	0.0102
47-48	0.0099	0.0378	0.0024
48-112	0.0140	0.0547	0.0036
48-49	0.0530	0.1830	0.0118
48-113	0.0261	0.0703	0.0046
112-49	0.0530	0.1830	0.0118
113-114	0.0105	0.0288	0.0019
114-50	0.0278	0.0762	0.0050
50-51	0.0220	0.0755	0.0050
50-52	0.0247	0.0640	0.0155
115-116	0.0023	0.0104	0.0007
4-57	0.0000	0.0267	0.0000
12-11	0.0000	0.0382	0.0000
71-64	0.0000	0.0388	0.0000
75-74	0.0000	0.0375	0.0000
90-25	0.0000	0.0386	0.0000

contd.

TABLE 5.1 contd.

1	2	3	4
91-26	0.0000	0.0268	0.0000
28-29	0.0000	0.0370	0.0000
93- 1	0.0000	0.0370	0.0000
98-37	0.0000	0.0370	0.0000

* Base MVA = 100

** Line charging : one-half of total charging
of line

TABLE A.5.2 Voltages at Generator Buses

Bus Number	Voltage Magnitude p.u.	Bus Number	Voltage Magnitude p.u.
1	1.035	28	1.005
2	0.998	29	1.050
3	0.990	30	0.955
4	1.015	31	0.984
5	1.050	32	0.980
6	0.990	33	0.991
7	0.970	34	0.958
8	0.973	35	0.943
9	0.962	36	1.006
10	0.992	37	1.040
11	1.050	38	0.985
12	1.015	39	1.015
13	0.968	40	1.005
14	0.967	41	0.985
15	0.963	42	0.980
16	0.984	43	0.990
17	0.980	44	1.010
18	0.970	45	1.017
19	0.985	46	1.010
20	1.005	47	0.971
21	1.025	48	0.965
22	0.955	49	0.952
23	0.952	50	0.973
24	0.954	51	0.980
25	0.985	52	0.975
26	0.995	53	0.993
27	0.998	54	1.005

TABLE A.5.3 Transformer Data

Transformer Designation	Tap Setting
4-57	0.985
12-11	0.960
71-64	0.960
75-74	0.938
90-25	0.960
91-26	0.985
28-29	0.935
93- 1	0.935
98-37	0.935

TABLE A.5.4 Static Capacitor Data

Bus Number	Susceptance *** p.u.*
16	0.14
20	0.10
34	0.12
48	0.20
49	0.06
50	0.06
57	-0.401
74	-0.25
79	0.10
80	0.10
82	0.15
97	0.20
99	0.20
100	0.10

*** The negative values represent reactors whereas the positive values are for capacitors.

TABLE A.5.5 Generation and Load Data

Bus Number	Real power generation p.u.*	Load	
		Real p.u.*	Reactive p.u.*
1	2	3	4
1	Slack bus	0.00	0.00
2	-0.09	0.30	0.12
3	0.00	0.52	0.22
4	-0.28	0.00	0.00
5	4.50	0.00	0.00
6	0.85	0.47	0.10
7	0.00	0.90	0.30
8	0.00	0.60	0.34
9	0.00	0.45	0.25
10	-0.13	0.00	0.00
11	2.20	0.00	0.00
12	3.14	0.00	0.00
13	-0.09	0.62	0.13
14	0.07	0.43	0.27
15	0.00	0.59	0.23
16	0.00	0.59	0.26
17	0.00	0.31	0.17
18	-0.46	0.20	0.23
19	-0.59	0.37	0.23
20	0.19	0.28	0.10
21	2.04	0.87	0.30
22	0.48	1.13	0.32
23	0.00	0.63	0.22
24	0.00	0.84	0.18
25	1.55	2.77	1.13
26	1.60	0.00	0.00
27	0.00	0.77	0.14
28	3.91	0.00	0.00

contd..

TABLE A.5.5 contd.

1	2	3	4
29	3.92	0.39	0.18
30	0.00	0.51	0.27
31	0.00	0.66	0.20
32	-0.12	0.00	0.00
33	-0.06	0.00	0.00
34	0.00	0.68	0.27
35	0.00	0.68	0.36
36	0.00	0.61	0.28
37	4.77	1.30	0.26
38	0.00	0.24	0.15
39	0.00	0.00	0.00
40	6.07	0.00	0.00
41	-0.85	0.78	0.42
42	-0.10	0.00	0.00
43	0.00	0.65	0.10
44	-0.42	0.00	0.00
45	2.52	0.37	0.18
46	0.40	0.23	0.16
47	0.00	0.38	0.25
48	0.00	0.31	0.26
49	-0.22	0.28	0.12
50	0.00	0.39	0.30
51	0.36	0.00	0.00
52	-0.43	0.25	0.13
53	-0.06	0.00	0.00
54	-1.84	0.00	0.00
55	0.00	0.20	0.09
56	0.00	0.39	0.10
57	0.00	0.00	0.00
58	0.00	0.19	0.02
59	0.00	0.00	0.00
60	0.00	0.70	0.23
61	0.00	0.34	0.16

contd.

TABLE A.5.5 contd.

1	2	3	4
62	0.00	0.14	0.01
63	0.00	0.25	0.10
64	0.00	0.11	0.03
65	0.00	0.18	0.03
66	0.00	0.14	0.08
67	0.00	0.10	0.05
68	0.00	0.07	0.03
69	0.00	0.17	0.07
70	0.00	0.24	0.04
71	0.00	0.00	0.00
72	0.00	0.23	0.09
73	0.00	0.33	0.09
74	0.00	0.00	0.00
75	0.00	0.00	0.00
76	0.00	0.27	0.11
77	0.00	0.37	0.10
78	0.00	0.18	0.07
79	0.00	0.16	0.08
80	0.00	0.53	0.22
81	0.00	0.34	0.00
82	0.00	0.20	0.11
83	0.00	0.17	0.04
84	0.00	0.17	0.08
85	0.00	0.18	0.05
86	0.00	0.23	0.11
87	0.00	0.12	0.03
88	0.00	0.12	0.03
89	0.00	0.78	0.03
90	0.00	0.00	0.00
91	0.00	0.00	0.00
92	0.00	0.28	0.07
93	0.00	0.00	0.00

contd.

TABLE A.5.5 contd.

1	2	3	4
94	0.00	0.00	0.00
95	0.00	0.47	0.11
96	0.00	0.71	0.26
97	0.00	0.39	0.32
98	0.00	0.00	0.00
99	0.00	0.54	0.27
100	0.00	0.20	0.10
101	0.00	0.11	0.07
102	0.00	0.21	0.10
103	0.00	0.48	0.10
104	0.00	0.12	0.07
105	0.00	0.30	0.16
106	0.00	0.42	0.31
107	0.00	0.38	0.15
108	0.00	0.15	0.09
109	0.00	0.34	0.08
110	0.00	0.22	0.15
111	0.00	0.05	0.03
112	0.00	0.43	0.16
113	0.00	0.02	0.01
114	0.00	0.08	0.03
115	0.00	0.08	0.03
116	0.00	0.22	0.07
117	0.00	0.20	0.08
118	0.00	0.33	0.15

*Base MVA = 100

TABLE NO.6.1.3

1 BUS	2 Voltage Magnitude	3 Phase Angle (DEG)	4 Real Power	5 Reactive power
1	1.03500	0.00000	5.1604	-0.6341
2	.99798	-14.40884	-0.3900	-0.2414
3	.98998	-16.69348	-0.5200	-0.0504
4	1.01500	- 8.93624	-0.2800	1.5818
5	1.04999	6.04734	4.5000	0.1570
6	.98997	-17.50190	0.3800	0.9212
7	.96998	-18.56041	-0.9000	-0.0976
8	.97298	-18.25133	-0.6000	-0.0003
9	.96198	-18.72452	-0.4500	-0.3404
10	.99198	- 8.90146	-0.1300	-0.0412
11	1.04999	- 1.77088	2.2000	0.6008
12	1.01500	0.02298	3.1400	0.4402
13	.96798	-14.38410	-0.7100	-0.0220
14	.96697	-17.00560	-0.3600	0.0996
15	.96298	-14.94035	-0.5900	-0.3154
16	.98398	-18.55778	-0.5900	-0.0367
17	.97998	-19.01238	-0.3100	-0.0498
18	.96998	-22.56691	-0.6600	0.1234
19	.98499	-21.37960	-0.9600	0.2458
20	1.00499	-11.44341	-0.0900	0.0151
21	1.02499	- 8.99635	1.1700	1.0959
22	.95499	-14.65920	-0.6500	-0.1999
23	.95199	-14.94896	-0.6300	-0.1560
24	.95399	-14.76095	-0.8400	-0.1340
25	.98500	-10.58796	-1.2200	-0.2344

Cont....

26	.99500	- 5.90271	-1.2200	-0.2978
27	.99799	- 6.51950	-0.7700	-0.0899
28	1.00500	- 2.30082	3.9100	1.5788
29	1.05000	- 2.46411	3.5300	-0.1542
30	.95497	-19.01499	-0.5100	-0.2770
31	.98400	- 7.37443	-0.6600	-0.0296
32	.97999	- 8. 89667	-0.1200	-0.0873
33	.99100	- 7.99805	-0.0600	0.1059
34	.95800	- 8.34431	-0.6800	-0.1798
35	.94300	- 8.23884	-0.6800	-0.2834
36	1.00600	- 3.31204	-0.6100	-0.0245
37	1.04000	- 1.07710	3.4700	1.0235
38	.98500	2.21841	-0.2400	-0.1138
39	1.01500	0.32236	0.0000	0.1367
40	1.00500	9.49917	6.0700	0.0493
41	.98500	3.11223	-1.6300	0.2014
42	.98000	3.14376	-0.1000	-0.1177
43	.99000	3.70329	-0.6500	-0.1461
44	1.01000	- 3.05001	-0.4200	-0.1558
45	1.01700	- 2.07822	2.1500	0.9262
46	1.01000	- 5.82019	0.1700	0.6405
47	.97100	- 8.39083	-0.3800	-0.2007
48	.96500	- 9.49555	-0.3100	-0.2018
49	.95200	-12.55991	-0.5000	0.0270
50.	.97299	-11.99528	-0.3900	-0.1988

Cont....

51	.98000	-10.35059	0.3600	-0.0136
52	.97499	-15.09490	-0.6800	0.2999
53	.99297	-16.03479	-0.0600	0.1652
54	1.00500	- 2.85847	-1.8400	0.9181
55	.97098	-18.47270	-0.2000	-0.0900
56	.96715	-18.12381	-0.3900	-0.1000
57	1.00188	-13.96364	0.0000	-0.4000
58	.98298	-17.14156	-0.1900	-0.0200
59	1.03275	-1 .56600	0.0000	0.0000
60	.98467	-16.98045	-0.7000	-0.2300
61	.96686	-18.35205,	-0.3400	-0.1600
62	.98269	-18.21781	-0.1400	-0.0100
63	.98210	-17.79244	-0.2500	-0.1000
64	.99256	-15.99445	-0.1100	-0.0300
65	.95381	-17.81185	-0.1800	-0.0300
66	.95344	-16.20290	-0.1400	-0.0800
67	.96457	-13.62939	0.1000	-0.0500
68	.99732	- 8.71453	-0.0700	-0.0300
69	.96086	-16.10805	-0.1700	-0.0700
70	.96282	-17.11500	-0.2400	-0.0400
71	.97662	-10.91894	0.0000	0.0000
72	.96874	-19.16520	-0.2300	-0.0900
73	.98002	-19.00848	-0.3300	-0.0900
74	.98840	-18.07024	0.0000	-0.2500
75	.95305	-12.88521	0.0000	0.0000

Cont.....

76	.96871	-21.47420	-0.2700	-0.1100
77	.96641	-22.99732	-0.3700	-0.1000
78	.97263	-18.54239	-0.1800	-0.0700
79	.97980	-16.02773	-0.1600	0.0200
80	.98328	-14.20458	-0.5300	-0.0400
81	1.01594	- 9.19936	-0.3400	0.0000
82	1.01977	- 9.98413	-0.2000	0.0400
83	1.00001	-11.01087	-0.1700	-0.0400
84	.96443	-13.60405	-0.1700	-0.0800
85	.95377	-14.55172	-0.1800	-0.0500
86	.94345	-15.54150,	-0.2300	-0.1100
87	.96933	-13.53727	-0.1200	-0.0300
88	.95720	-14.38508	-0.1200	-0.0300
89	.99302	- 6.79354	-0.7800	-0.0300
90	.96677	- 7.20151	0.0000	0.0000
91	.98170	- 5.42737	0.0000	0.0000
92	1.01887	- 5.09583	-0.2800	-0.0700
93	1.00176	- 2.42376	0.0000	0.0000
94	.98655	- 7.78369	0.0000	0.0000
95	.96623	- 7.06798	-0.4700	-0.1100
96	1.00331	- 3.61496	-0.7100	-0.2600
97	1.00894	- 3.31522	-0.3900	-0.1200
98	.99335	- 1.88862	0.0000	-0.0000
99	.98574	- 2.83483	-0.5400	-0.0700

Cont....

100	.98190	-1.69731	-0.2000	0.0000
101	.97852	0.73312	-0.1100	-0.0700
102	.98474	0.56132	-0.2100	-0.1000
103	.98691	5.40991	-0.4800	-0.1000
104	.98391	0.71666	-0.1200	-0.0700
105	.98769	-1.42843	-0.3000	-0.1600
106	.97766	-2.39092	-0.4200	-0.3100
107	.98937	-2.54612	-0.3800	-0.1500
108	1.00908	-2.15625	-0.1500	-0.0900
109	1.02217	-2.64544	-0.3400	-0.0800
110	.99003	-0.49192	-0.2200	-0.1500
111	.98827	2.20642	-0.0500	-0.0300
112	.96008	-9.74087	-0.4300	-0.1600
113	.96573	-10.68539	-0.0200	-0.0100
114	.96654	-11.13837	-0.0800	-0.0300
115	.96974	-15.26356	-0.0800	-0.0300
116	.95966	-15.27303	-0.2200	-0.0700
117	.97255	-19.03045	-0.2000	-0.0800
118	.94869	- 8.07298	-0.3300	-0.1500

Table 6.14

S.No.	Method	Acceleration factor	No. of 8 iterations	CPU time (in secs)
1	Gauss Seidal Method	1.7	116	37.58
		1.6	>150	...

CHAPTER VIIC O N C L U S I O N :

Sparsity techniques have been applied in programming load flow studies, both by Gauss-Seidal method and by Newton-Raphson method. The row wise systematic packing with necessary indexing information has been used in storing the admittance matrix. The Jacobian in the case of Newton-Raphson method has been stored in both row wise and column wise systematic packed form. In addition the linked list technique has been used as it is adept to deal with the continually changing number of non-zero elements in the rows and columns during the reduction process.

In the case of Newton-Raphson technique, bi-factorization method has been used which is especially suitable for load flow studies as the jacobian matrix is diagonally dominant and having a symmetric sparsity structure but asymmetric in element value. As the structure of the jacobian remains same in each iteration, the simulation and ordering subroutine will once for all determine the pivotal sequence which can be applied in subsequent iterations.

The results of the problems studied have been given in Chapter VI. It is seen that in the case of Gauss-Seidal method the number of iterations required to reach the solution is more compared to the Newton-Raphson method. The effect of acceleration

factor on the number of iterations in the case of the Gauss-Seidal method has also been studied. For the smaller systems the optimum acceleration factor is in the range of 1.2 to 1.4 (For example for the 5 bus system it is approximately 1.2 and for the 8 bus system it is approximately 1.4). For the larger systems it is approximately 1.7 and if it is reduced the number of iterations increases.

The time taken to obtain a solution in the case of Newton-Raphson method for the smaller systems is more (i.e. for a 5 bus system it is 0.57 secs as against 0.27 secs. for Gauss-Seidal method with an acceleration factor of 1.2). This is because of the degree of sparsity in the smaller system is small and also due to time taken to calculate the jacobian elements. However the solution was obtained in two iterations.

In the case of larger systems the time for solution by Newton Raphson method (time taken for 57 bus system is 0.73s) is less than by the Gauss-Seidal method (time taken for 57 bus system is 4.5 secs). This is due to fast convergence of Newton Raphson method (No. of iterations 3 in case of 57 bus) Also for the 57 bus system studied the percentage of sparcity is 93.5 % (only 213 non zero elements in the admittance matrix)

Suggestions for variations and improvement:

In the case of Gauss-Seidal method only the diagonal elements and the non-zero elements of the upper triangle of Y-Bus are stored. In this case the matrix has to be searched everytime to build up the remaining non-zero elements of the row and so the computation time is more. This can be reduced by storing all the non zero elements of the Y-Bus. However ~~the~~ memory requirement will go up.

In the case of Newton Raphson method the subroutine JACOB calculates the elements of the jacobians for each iteration and also calculates the real and reactive bus powers. The program can be made faster by having a separate subroutine^{to} calculate the bus powere and if the convergence criterion is not met with, then only to calculate the elements of the jacobian.

In the present work, non linear loads and on load tap changing, could not be considered. These can be incorporated in the program.

REFERENCESPAPERS

1. Stott, B., "Review of Load Flow Calculation Methods" special issue on Computers in Power Industry, Proc., IEEE, Vol. 62 No. 7, July 1974.
2. Churchill, M.E., "A sparse matrix procedure for power system analysis programs " in " Large Sparse Sets of Linear Equations" (J.K. Reid, ed.) pp 127-138, Academic Press, New York, 1978.
3. Gupta, P.P., and Humphrey Davier, M.W., "Digital Computers in Power System Analysis", Proc, Inst. Elec. Eng., Vol. 108 A, PP. 383-404, January 1961.
4. Brheaur, A. and Denmed, J.K. "Some improved methods of digital network analysis", Proc. Inst., Elec. Eng. Vol 109A, pp 109-116, February 1962.
5. Brown, H.E., Carter, G.K., Happ, H.H. & Person, C.E. "Power flow solution by impedance matrix iterative method" IEE Trans. Power App. Syst. Vol. PAS-82, pp. 1-10, Setp. 1963.
6. Vanbess, J.E. "Iteration methods for digital load flow studies" AIEE Trans (Power APP. Syst.) Vol. 78, PP. 583-588 August 1959.
7. Vanness, J.E. & Griffen, J.H. "Elimination methods for load flow studies" AIEE Trans. (Power APP.Syst.) Vol-80, pp 229-304, June, 1961.
8. Tiney, W.F. & Walker, J.W. "Direct Solution of Sparse Network Equations by Systematically Ordered triangular factorization", Proc. IEEE, Vol. 55, PP. 1801-1809, Nov. 1967.

9. Dommel, H.W. & Tinney, W.F. "Optimal Power Flow Solution" IEEE Trans. Power App. Syst., Vol. PAS-871, PP. 1866-1874, October 1968.
10. Happ, H.H. "Diakoptics - The solution of system problems by Tearing", Special issue on Computers in Power Industry, Proc. IEEE, Vol. 62, No. 7, July 1974, PP. 930-940.
11. Ward, J.E. & Hale, H.W., "Digital Computer Solution of Load Power Flow Problems", AIEE Trans (Power App. Syst.) Vol. 75 PP. 398-404, June, 1956
12. Glimn, A.F. & Stagg, G.W. "Automatic Calculation of Load Flows", AIEE Trans. (Power App. Syst.), Vol 76, PP 817-828, Oct. 1957.
13. Brown, R.J. & Tinney, W.F., "Digital solutions for Large Power Networks", AIEE Trans (Power App. Syst.), Vol 76, PP. 347-355, June, 1957.
14. Nisato and Tinney, W.F., "Techniques for exploiting the Sparsity of the network admittances matrix", IEEE Transactions on Power Apparatus and Systems 82 (1963), 944-950.
15. Tinney, W.F. "Some examples of sparse matrix methods for power network problems". In Proceedings of the 3rd Power Systems Computations Conference" Rome, 1969.

16. Zollenkopf, K. , " Bi-Factorization: Basic Computational Algorithm and Programming Techniques " In Large Sparse Sets of Linear Equations (J.K. Reid, ed.) PP. 75-96, Academic Press, New York.
17. Brown, Roney J., and Tinney, William F., " Digital Solutions for Large Power Networks, Trans. AIEE, Vol. 76, Pt.II, PP. 347-355, 1957.
18. Gronin, J.H., and M.B. Newman: "Digital Load Flow Program for 1,000 Bus Systems ", Trans. IEEE on Power Apparatus and Systems, Vol. 83, pp. 718-720, 1964.
19. Ogbuobiri, E.C. " Dynamic Storage and Retrieval in Sparsity Programming " IEEE Trans. Power Apparatus System PAS 89, 150-55. 1970.
20. Ogbuobiri, E.C. " Sparsity Techniques in Power-Systems Grid-Expansion Planning ". In Large Sparse Sets of Linear Equations (J.K. Reid, ed) PP 219-230, Academic Press, New York, 1971.
21. Ogbuobiri, E.C., Tinney, W.F. & Walker, J.W. "Sparsity directed decomposition for Gaussian elimination on matrices " IEEE Trans. Power Apparatus Systems, PAS 89, 141-155, 1970.
22. Berry, R.D. " An optimal ordering of electronic circuit equations for a sparse matrix solution ". IEEE, Trans. (CT-18), 40-50, 1971.

23. Hacttel, G., Brayton, R. & Gustavson, F. " The Sparse Tableau Approach to Network Analysis and Design " IEEE Trans. (CT-18), 101-13, 1971.

B O O K S.

24. Stevenson, W.D. " Elements of Power Systems Analysis ", 2nd ed, Mc.Graw Hill Book Company, New York, 1962.
25. Stagg, G.W. and Abiad, A.H.E " Computer Methods in Power Systems Analysis," Mc. Graw Hill Book Company, New York, 1968.
26. Reid, J.K. (ed), "Large Sparse Sets of Linear Equation ", Academic Press, New York, 1971.
27. Jennings, A. " Matrix Computation for Engineers and Scientists ", John Wiley & Sons, New York, 1977.
28. Brameller, A., Allan, R.N., & Hamam, Y.M. " Sparsity, its practical application to system analysis " Pitman Ltd., London, 1976.
29. Tewarson, R.P. " Sparse Matrices ", Academic Press, New York, 1973.
30. Pai, M.^A. " Computer Techniques in Power System Analysis ", Tata Mc. Graw Hill Publishing Company Ltd., New Delhi, 1979.
31. Elgerd, O.I. " Electrical Energy Systems Theory ", Mc. Graw Hill Book Company, New York, 1971.

APPENDIX

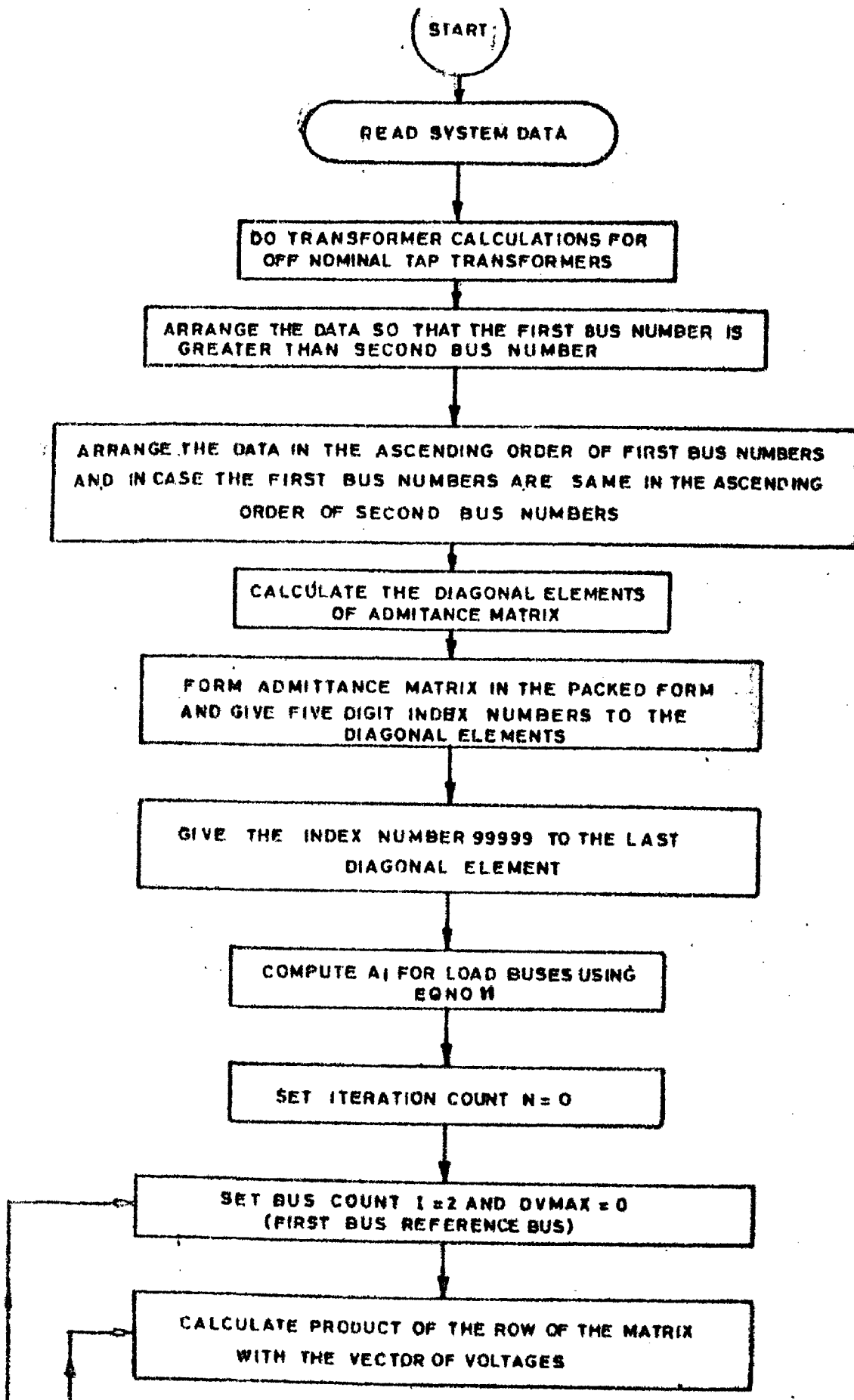


FIG. A.2.1 FLOW DIAGRAM FOR GAUSS-SEIDEL METHOD

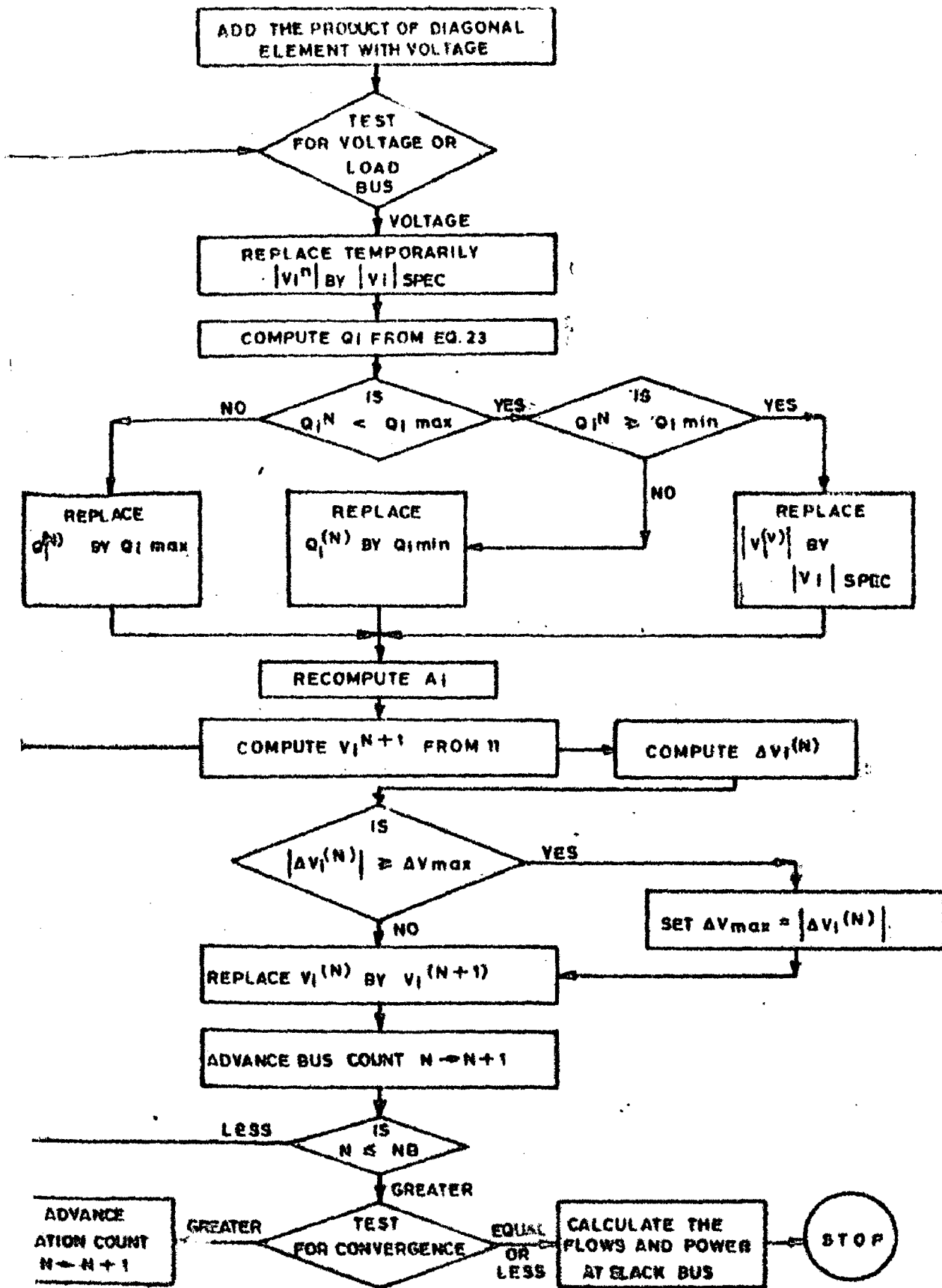


FIG. A.2.1 CONTINUED

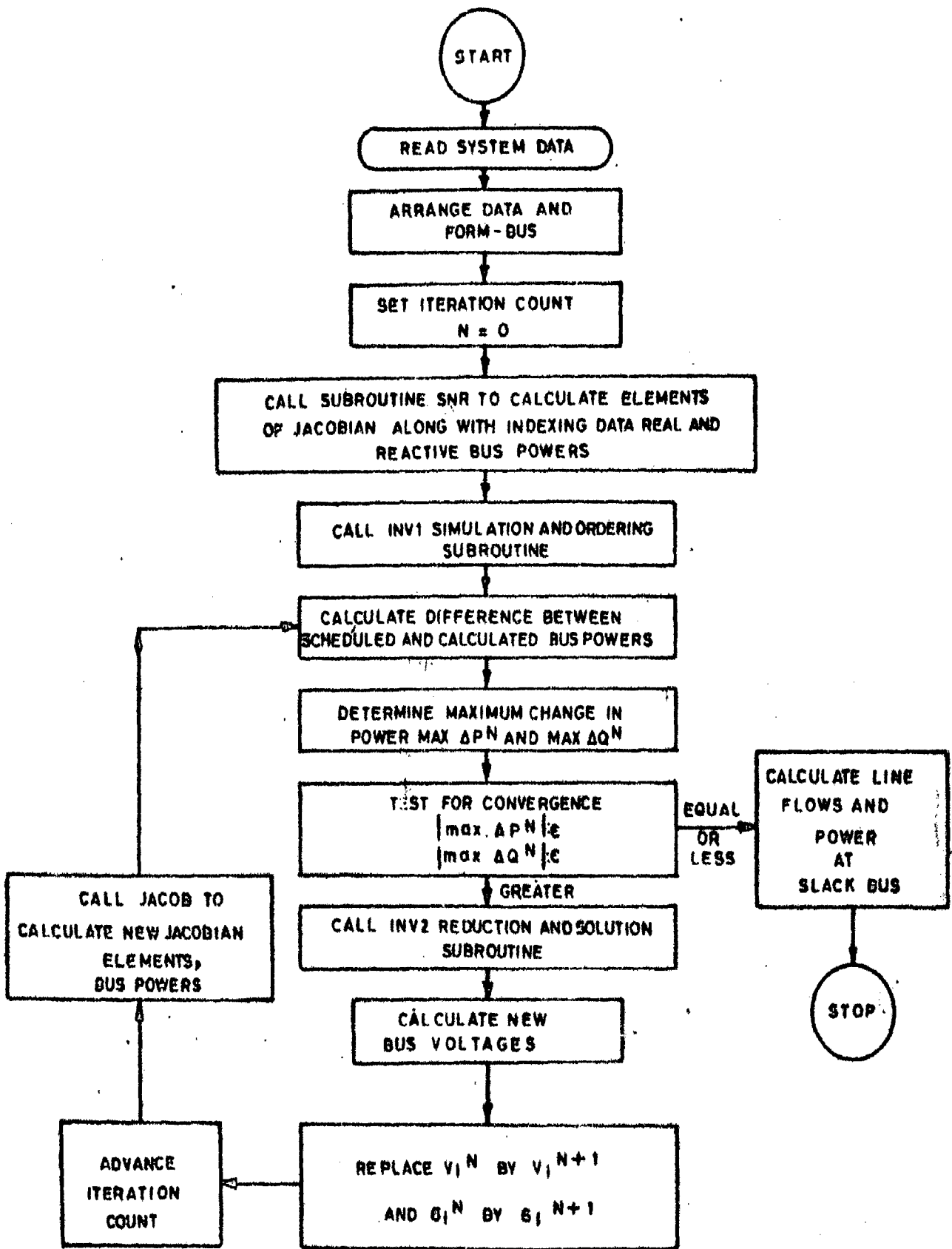
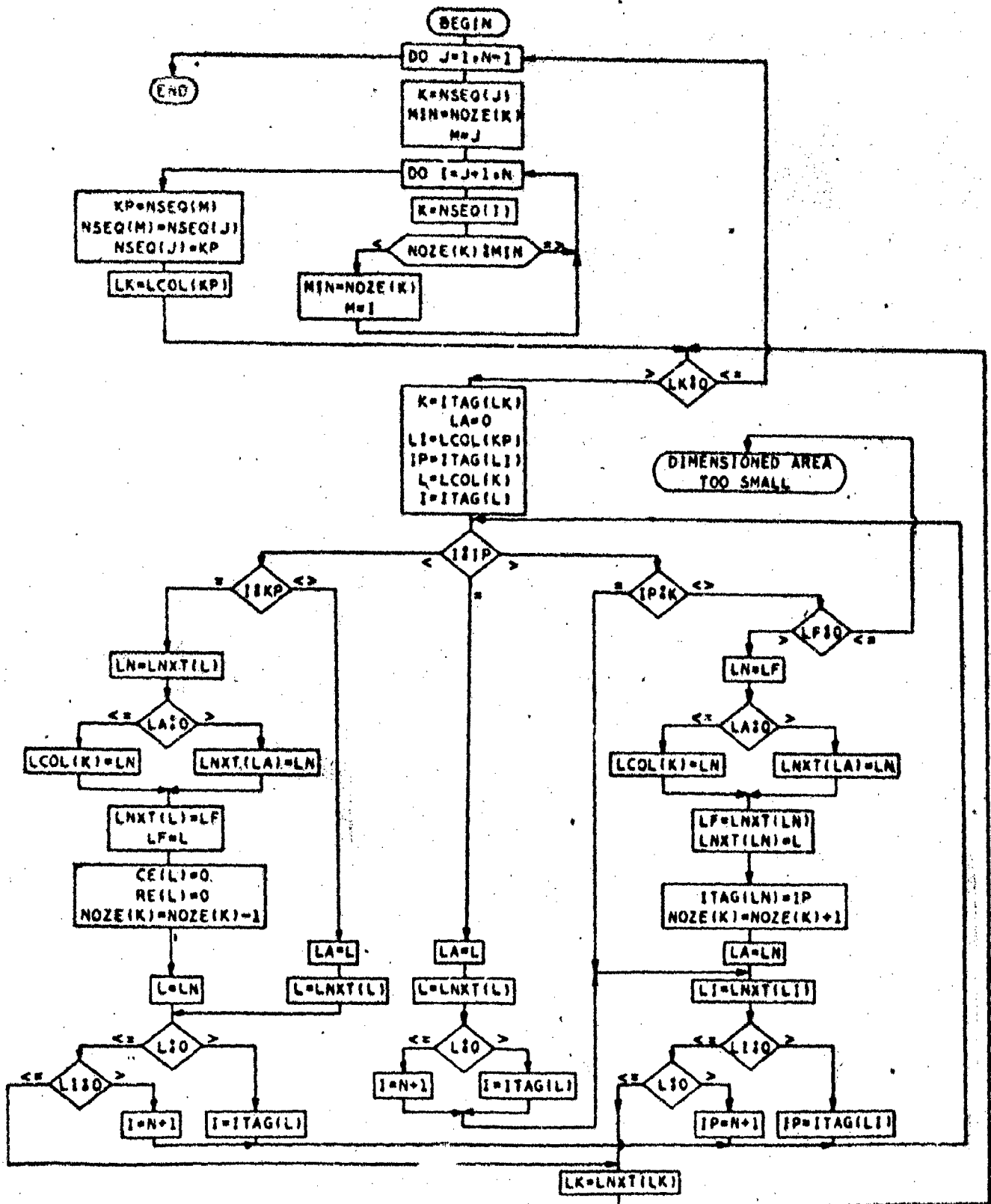
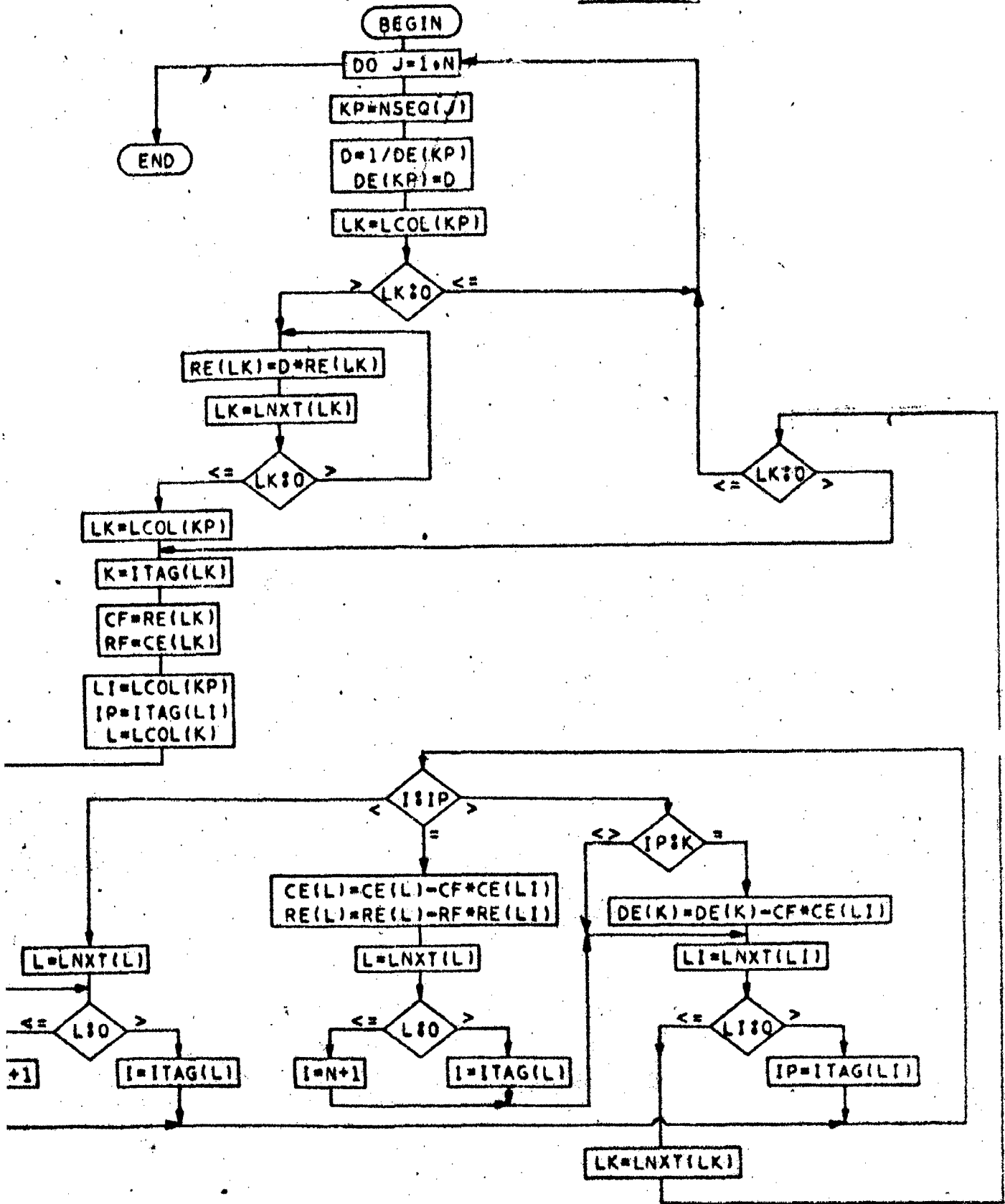


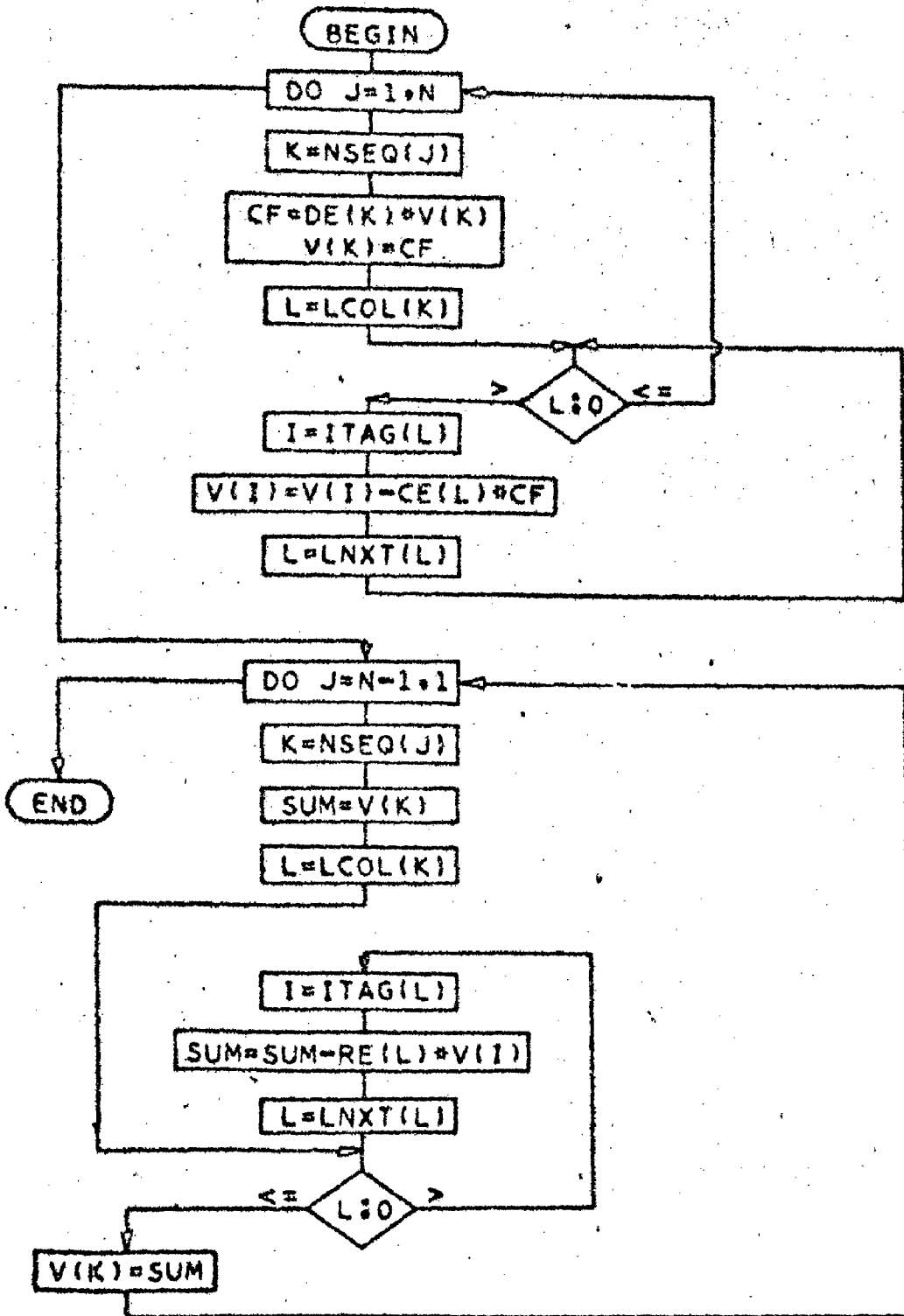
FIG. A.2.2 LOAD FLOW SOLUTION BY NEWTON-RAPHSON METHOD



A.2.3. Flow chart of sir station and ordering subroutine (asymmetrical matrix)



A.2.4. Flow chart of reduction subroutine (asymmetrical matrix)



Flow chart of direct solution subroutine (asymmetrical matrix)

Description of parameter

Integer variables

I row index of terms in processed column K or running index
IP row index of terms in pivotal column
J number of reduction step
K index of column under consideration or running index
KP pivotal index related to reduction step J
L location of terms in processed column K
LA location of preceding term in processed column K
LF indicator for next vacant location
LI location of terms in pivotal column (inner loop)
LK location of terms in pivotal column (outer loop)
LN location of new added fill-in term
LP location of intermediately stored terms of pivotal column
M intermediate integer variable
MIN minimum number of non-zero terms
N number of unknowns, order of the matrix

Real variables

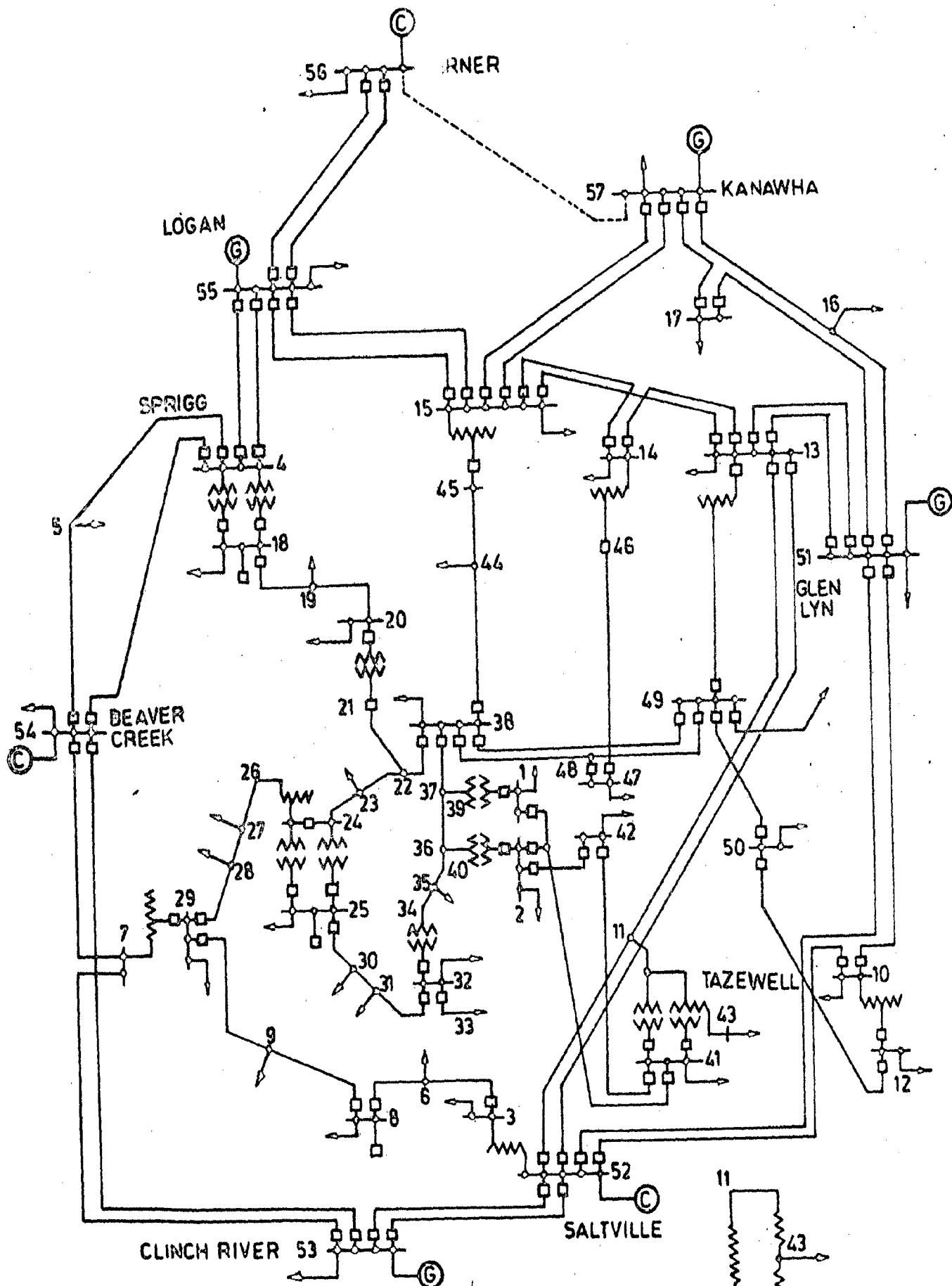
CF multiplier for columns
D diagonal (pivotal) term
RF multiplier for rows
SUM sum of products

Integer arrays

ITAG row index of elements stored in CE
LCOL starting position of columns
LNXT location of next term
NOZE number of non-zero terms
NSEQ sequence of pivotal indices

Real arrays

CF columnwise stored matrix terms
DE diagonal terms
RE rowwise stored matrix terms
V vector of unknowns, solution vector



(C) BUS-CODE DIAGRAM
 (C) SYNCHRONOUS COMPENSATORS
 (G) GENERATORS

TAZEWELL TRANSFORMERS

FIG.A.2.6 IEEE 57-BUS TEST SYSTEM

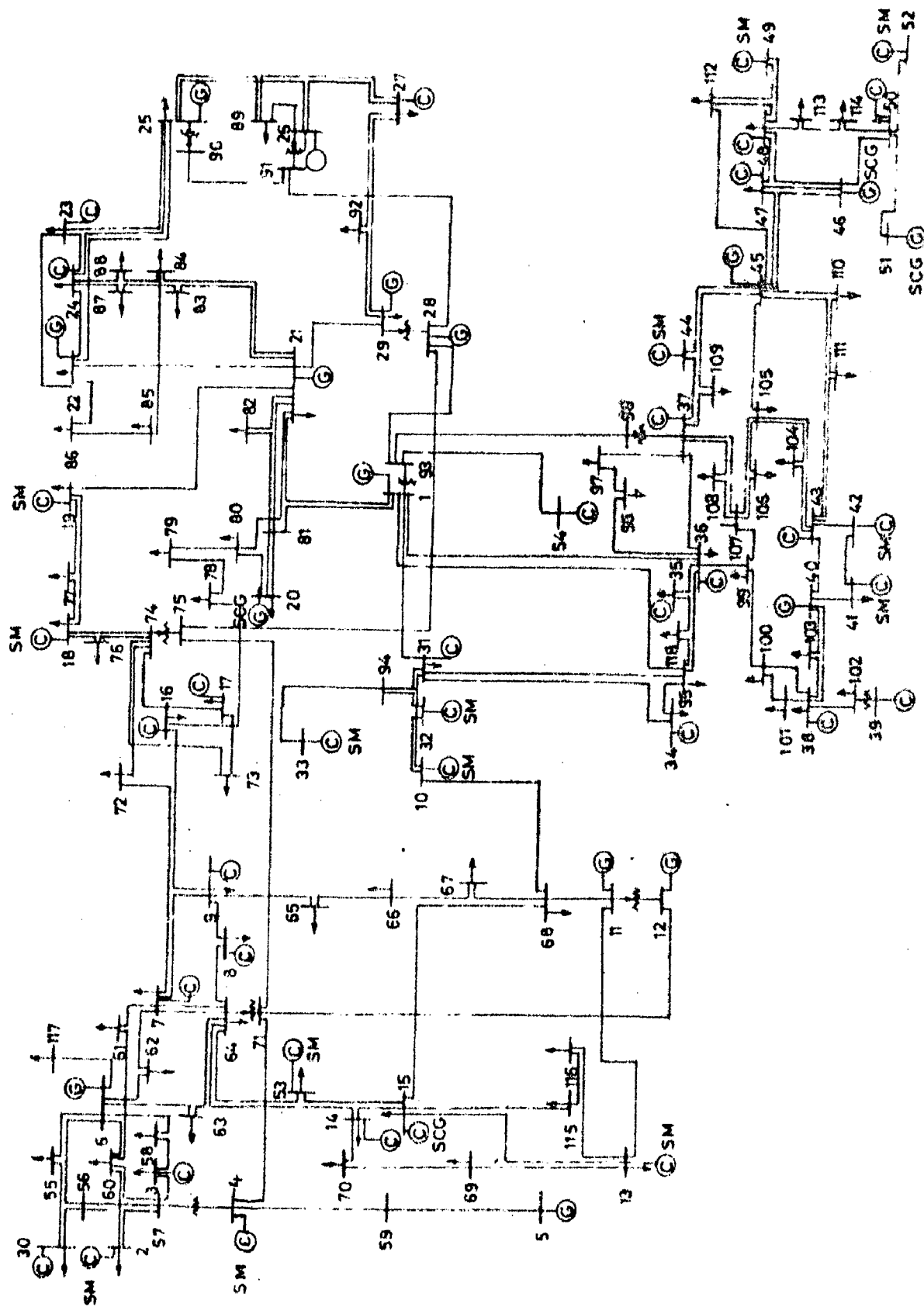


FIG A.2.7 I.E.E.E. 118 BUS TEST SYSTEM

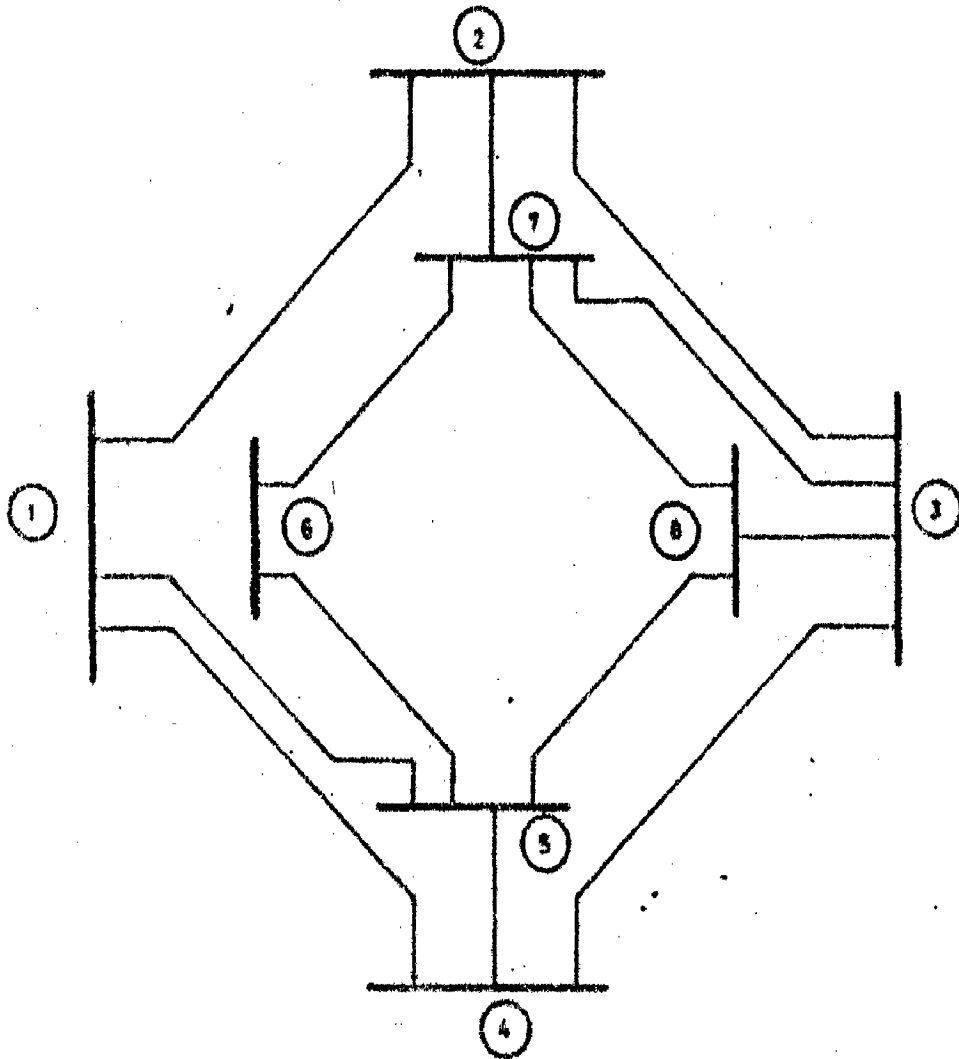


FIG A.2 8 8 - BUS SYSTEM

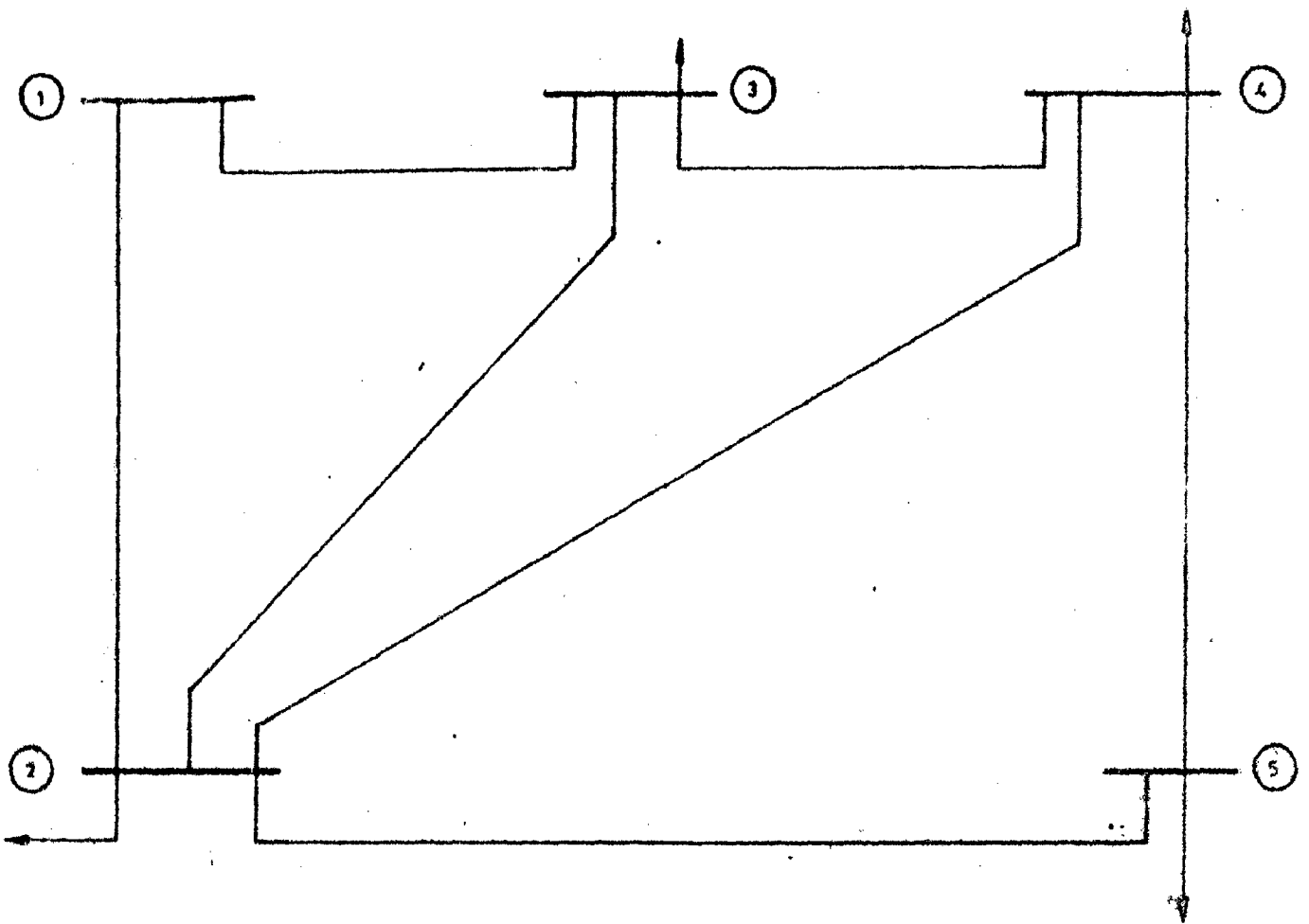


FIG. A.2.9 5 - BUS SYSTEM