THE LOAD FLOW PROBLEM

WITHOUT SLACK BUS

by

Low Sek Luen
B. Eng. (Hons.) (University of Malaya)

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

Department of Electrical Engineering,
McGill University,
Montreal, CANADA.
August, 1979.

ABSTRACT

In conventional load flow studies, for mathematical reasons, a generation bus must be selected as the slack bus which cannot have its real generation specified independently. This is at variance with the physical power system in which no such bus is so designated.

This thesis develops two physically meaningful load flow models which do not require a slack bus. In the Floating System Voltage Load Flow, all the real power injections are independently specified at the expense of freeing the voltage level of the system. With the Participation Factors Load Flow, the total power generated is not specified a priori, nor are the real power generated at each PV bus; instead at each PV bus is specified its fractional contribution to the total generation required to satisfy the total demand plus network loss.

The applications of these no-slack bus load flow methods are also investigated. Employing the provision that all the real generations can be independently specified in the Floating System Voltage Load Flow, economic dispatch calculations can be simplified. Given the proper physical interpretations, the Participation Factors Load Flow enables modelling of a power system with consideration for the actions of the turbine speed control governors.

RÉSUMÉ

Dans les études d'écoulement de puissance, pour des raisons mathématiques, une barre de génération doit être choisie comme la barre d'oscillation qui ne peut pas avoir sa production spécifiée indépendamment. Une telle barre n'existe pas dans les réseaux électriques de puissance causant une variance entre la pratique et le modèle.

Cette thèse développe deux modèles qui se rapprochent plus du système physique sans la nécessité de définir une barre d'oscillation. Dans l'Ecoulement de Puissance avec Voltage de Réseau Libre, toutes les injections de puissance réelles sont indépendamment spécifiées en laissant libre le niveau de voltage du réseau. Dans l'Ecoulement de Puissance aux Facteurs de Participation, la puissance totale active de production ainsi que la production à chaque barre PV ne sont pas spécifiées. A chaque barre PV on spécifie la contribution fractionnelle de la génération totale requise pour satisfaire la demande totale et les pertes de transmission.

Les applications de ces méthodes d'écoulement de puissance sans barre d'oscillation sont aussi étudiées. Il est possible, par exemple, de simplifier largement le calcul du dispatching économique en employant la proprieté de l'Ecoulement de Puissance avec Voltage de Réseau Libre où les productions réelles peuvent être specifiées indépendamment. L'Ecoulement de Puissance aux Facteurs de Participation nous permet en plus, de modeler le système en tenant compte des actions des gouvernails de vitesse des turbines.

ACKNOWLEDGEMENTS

The author expresses his sincere gratitude to his supervisor, Professor F. D. Galiana for his guidance, kindness and encouragement. The author also gratefully acknowledges the financial support provided in part by the Division of Electric Energy Systems, U. S. Department of Energy and the National Science and Engineering Research Council of Canada.

On the non-academic side, the author offers his deep appreciation to his wife, Diana for her constant encouragement and untiring efforts as homemaker. Last but not least, the author is grateful to his parents for their love and sacrifices.

TABLE OF CONTENTS

1. INTRODUCTION

In almost any kind of power system analysis the load flow program has an important role to play. Essentially the load flow program establishes the steady state operating point of the system in terms of the voltages at all the nodes of the system. In another era, before the digital computer appeared on the scene, load flow studies were carried out as analog simulations of the system on AC boards which were scaled-down electrical models of the power system under study. However with the advent of the digital computer, digital simulations have displaced the AC boards. This changeover is due to a number of significant advantages the digital computer has over the AC boards [1], the principal ones being cost, accuracy, flexibility and convenience.

Ever since the first load flow programs in the mid-1950's, a proliferation of papers on the subject had appeared in the literature. A survey of the papers presented is given in [2]. The extensive past efforts notwithstanding, current problems of increasing system sizes and on-line applications still demand innovations. As systems grow larger, memory space in the computer is severely taxed, while for on-line applications, execution times need to be reduced. These difficulties are ameliorated by the recognition and exploitation of the fact that most power system matrices are sparse.

With sparse matrices much storage space can be saved by not storing the zero elements of the matrix. When it is estimated that more than ninety-nine percent of the elements of many large power system matrices are zeroes, one can appreciate the savings realized. At the same time, matrix operations involving the zero elements are trivial, so without performing operations on those zero elements which are not stored, execution time is reduced too. Hence sparse matrix handling is a valuable technique to incorporate into load flow programs.

In the currently accepted methods of load flow calculations, there is entrenched the concept of a slack bus which is the generator to which is apportioned all the transmission loss of the network. Although there exists no slack bus as such in the physical system, the slack bus is nevertheless a necessary consequence of the load flow formulations in current use. This discrepancy between the real world and the model of the system can, as it will be shown in this thesis, be resolved.

Whereas there has been research on the subject of no-slack bus load flow methods [3,4], to the best of the author's knowledge, the work carried out in this thesis is the first attempt to systematically formulate the load flow problem as they appear in the following pages under the names the Floating System Voltage Load Flow and the Participation Factors Load Flow. The author believes the ability to incorporate the droop characteristics of generation units to the load flow solution method is also without precedent.

This thesis examines the justification for the slack bus and develops two models which realistically describe the physical system and at the same time do not require the specification of one of the buses as the slack bus. Many other such no-slack bus formulations are possible when it is recognized that in essence, in a load flow the unknowns are calculated from a set of simultaneous algebraic equations which relates the given operating conditions to the unknowns. Provided that the set of equations has been formulated as a consistent set with as many independent equations as unknowns, it constitutes a valid load flow model of the system.

In the conventional load flow methods the given operating conditions are the real and reactive powers at the load buses and real powers and voltage magnitudes at the generator buses except the slack bus which only has its voltage magnitude specified. Had the slack bus real generation been specified as well, it would overspecify the system, which would almost certainly result in an inconsistent system of equations.

However the above need not be and in the last analysis are not the exact operating conditions of a power system, governors may operate to alter the generation levels when there is a deviation from standard frequency resulting from an unbalance between generation and demand. Similarly, generator exciters and tap changing transformers may fine tune the system voltages. Also a certain line, for example, a tie-line, may be under contract to transmit a given power and this becomes a specified operating condition. The equation relating this tie-line flow to the unknown variables could equally well be one of the equations that constitute a valid load flow model of the system. Thus this thesis is addressed to the alternative formulations of the load flow problem and their relevance to the physical power system to be modelled.

Chapter TWO provides the background material and describes the conventional load flow formulation. It also presents in some detail the analytical as well as numerical considerations of the Newton-Raphson load flow method.

Chapter THREE develops two no-slack bus load flow formulations and their implementations. In the Floating System Voltage Load Flow, all the generation buses are dispatched but their voltage magnitudes are variable whereas in the Participation Factors Load Flow, the generation buses' voltages are fixed while the total generation is variable through the participation of all the generation units.

Chapter FOUR describes applications of the new load flow formulations. Through the Floating System Voltage Load Flow, economic dispatch problems can be simplified. The chapter also offers a possible physical interpretation of the Participation Factors. Using this physical interpretation, the Participation Factors Load Flow can be applied to model the allocation of new generations to rectify a generation-demand imbalance situation.

Chapter FIVE presents the detailed results of the study of a 5 bus system. Examples of the applications of the new load flow formulations are also described.

2. THE CONVENTIONAL LOAD FLOW

2.1 Formulation of the Conventional Load Flow Problem

In a load flow problem it is desired to find the voltages at all the nodes of a power network given the operating conditions of the system. Then with the values of the voltages so obtained, any other dependent variables like the power injections at the buses or the power flows between the buses could be calculated. In the conventional load flow formulation, the operating conditions are specified by the real and reactive injections at the load buses and the voltage magnitudes and real injections at the generator and regulated buses. In the load flow problem then, the objective is to find the voltages at all the nodes such that these specified injections are satisfied. Each specified injection being expressible as a quadratic function of the nodal voltages in the network, the load flow problem can thus be formulated and solved as a set of nonlinear algebraic equations.

It was mentioned above that in the conventional formulation, the operating conditions of the network are specified by the real and reactive injections at the load buses and the voltage magnitudes and real injections at the generation buses. However in a network with yet undetermined transmission loss it is not possible and mathematically untenable to specify all the real injections a priori, with the reactive injections at the load buses and the voltage magnitudes at the generation buses specified as well. Hence appears the necessity of the artifice of a slack bus (one of the generation buses) which has only its voltage magnitude specified but not its real injection. In the system there is also an arbitrarily chosen reference bus from which all the voltage angles of other buses are measured. The slack bus is often (but need not necessarily be) chosen to be the reference bus. Having considered these details, we shall proceed with the mathematical formulation of the load flow problem.

The real and reactive power injections into a node i can be expressed as

$$P_i + jQ_i = V_i I_i^* \tag{1}$$

where V_i is the voltage at node i and I_i^* the conjugate of the current entering node i.

The current entering a node i may be expressed in terms of the nodal voltages as follows

$$I_i = \sum_k Y_{ik} V_k \tag{2}$$

where the summation is over all the nodes in the system and Y_{ik} is the $(i,k)^{th}$ element of the bus admittance matrix of the network.

Working with rectangular co-ordinates, the nodal voltages are resolved into the real and imaginary components

$$V_i = e_i + jf_i$$

From (1) and (2) we have,

$$\begin{aligned} P_i &= \text{Real } \{ V_i I_i^* \} \\ &= \text{Real } \{(e_i + jf_i) \sum_k Y_{ik}^* (e_k - jf_k)\} \\ &= \text{Real } \{(e_i + jf_i) \sum_k (g_{ik} - jb_{ik})(e_k - jf_k)\} \\ &= e_i \sum_k (e_k g_{ik} - f_k b_{ik}) + f_i \sum_k (f_k g_{ik} + e_k b_{ik}) \qquad (3) \end{aligned}$$

and $$Q_i = f_i \sum_k (e_k g_{ik} - f_k b_{ik}) - e_i \sum_k (f_k g_{ik} + e_k b_{ik}) \qquad (4)$$

Further, the voltage magnitude squared at a node i is

$$|V|_i^2 = e_i^2 + f_i^2 \qquad (5)$$

Equations (3), (4) and (5) constitute the types of equations to be solved in the load flow problem. The set of load flow equations are:

For a load bus i

$$P_{i\,spec} = e_i \sum_k (e_k g_{ik} - f_k b_{ik}) + f_i \sum_k (f_k g_{ik} + e_k b_{ik})$$

and $$Q_{i\,spec} = f_i \sum_k (e_k g_{ik} - f_k b_{ik}) - e_i \sum_k (f_k g_{ik} + e_k b_{ik})$$

For a generator bus i

$$P_{i\,spec} = e_i \sum_k (e_k g_{ik} - f_k b_{ik}) + f_i \sum_k (f_k g_{ik} + e_k b_{ik})$$

and $$|V|^2_{i\,spec} = e_i^2 + f_i^2$$

For the slack bus

$$|V|^2_{s\,spec} = e_s^2 + f_s^2$$

Hence there are in all (2n - 1) simultaneous algebraic equations to solve, 2 equations each from the load and generation buses and one from the slack bus, n being the number of buses in the system. However for the reference bus with its zero voltage angle implying $f_{ref} = 0$, equation (5) is trivial to solve and the system can be reduced to (2n - 2) equations with (2n - 2) unknowns.

2.2 The Newton Raphson Algorithm

With the problem formulated, next we proceed with the solution. The equations (3), (4) and (5) can be written concisely in the form

$$P_{i\,spec} = P_i(\underline{e},\underline{f})$$
$$Q_{i\,spec} = Q_i(\underline{e},\underline{f}) \tag{6}$$
$$|V|^2_{i\,spec} = V_i(\underline{e},\underline{f})$$

where P_i, Q_i and V_i are functions of $\underline{e}$ and $\underline{f}$, the vectors of unknown real and imaginary components of the nodal voltages.

By the Taylor series expansion,

$$P_{i\,spec} = P_i(\underline{e}_o + \Delta\underline{e},\ \underline{f}_o + \Delta\underline{f}) \tag{7}$$

$$= P_i(\underline{e}_o,\underline{f}_o) + \left.\frac{\partial P_i}{\partial \underline{e}}\right|_{\underline{e}_o}\Delta\underline{e} + \left.\frac{\partial P_i}{\partial \underline{f}}\right|_{\underline{f}_o}\Delta\underline{f} + \ldots$$

where ... denotes higher order terms in $\Delta\underline{e}$ and $\Delta\underline{f}$.
Grouping together the known quantities onto the L.H.S.

$$P_{i\,spec} - P_i(\underline{e}_o,\underline{f}_o) = \left.\frac{\partial P_i}{\partial \underline{e}}\right|_{\underline{e}_o}\Delta\underline{e} + \left.\frac{\partial P_i}{\partial \underline{f}}\right|_{\underline{f}_o}\Delta\underline{f} + \ldots$$

Similarly,

$$Q_{i\,spec} - Q_i(\underline{e}_o,\underline{f}_o) = \left.\frac{\partial Q_i}{\partial \underline{e}}\right|_{\underline{e}_o}\Delta\underline{e} + \left.\frac{\partial Q_i}{\partial \underline{f}}\right|_{\underline{f}_o}\Delta\underline{f} + \ldots \tag{8}$$

$$|V|^2_{i\,spec} - V_i(\underline{e}_o,\underline{f}_o) = \left.\frac{\partial V_i}{\partial \underline{e}}\right|_{\underline{e}_o}\Delta\underline{e} + \left.\frac{\partial V_i}{\partial \underline{f}}\right|_{\underline{f}_o}\Delta\underline{f} + \ldots$$

Equation (7) can be viewed as, given an estimate of the solution $(\underline{e}_0, \underline{f}_0)$ the equation is satisfied if this estimated value of the solution is corrected by an amount $(\Delta\underline{e}, \Delta\underline{f})$. Hence finding $(\Delta\underline{e}, \Delta\underline{f})$ is equivalent to solving the original set of equations. Provided the corrections are small (i.e. the estimated solution is close to the actual solution) the higher order terms in (8) could be neglected and the load flow problem may be reduced to a set of linear equations in $\Delta\underline{e}$ and $\Delta\underline{f}$. This set of equations is:-

$$
\begin{aligned}
P_{i\,spec} - P_i(\underline{e}_0, \underline{f}_0) &= \left.\frac{\partial P_i}{\partial \underline{e}}\right|_{\underline{e}_0} \Delta\underline{e} + \left.\frac{\partial P_i}{\partial \underline{f}}\right|_{\underline{f}_0} \Delta\underline{f} \\
Q_{j\,spec} - Q_j(\underline{e}_0, \underline{f}_0) &= \left.\frac{\partial Q_j}{\partial \underline{e}}\right|_{\underline{e}_0} \Delta\underline{e} + \left.\frac{\partial Q_j}{\partial \underline{f}}\right|_{\underline{f}_0} \Delta\underline{f} \qquad (9) \\
|V|^2_{k\,spec} - V_k(\underline{e}_0, \underline{f}_0) &= \left.\frac{\partial V_k}{\partial \underline{e}}\right|_{\underline{e}_0} \Delta\underline{e} + \left.\frac{\partial V_k}{\partial \underline{f}}\right|_{\underline{f}_0} \Delta\underline{f}
\end{aligned}
$$

where i = all the buses except the slack, j = all the load buses and k = all the generator buses except the slack

This unweildly set of equations (9) can be more elegantly expressed in linear algebraic notation

$$\underline{b} = [J]\underline{x} \qquad (10)$$

where $\underline{b}$ is the vector of mismatches consisting of $P_{i\,spec} - P_i(\underline{e}_o,\underline{f}_o)$, $Q_{j\,spec} - Q_j(\underline{e}_o,\underline{f}_o)$, $|V|^2_{k\,spec} - V_k(\underline{e}_o,\underline{f}_o)$; $\underline{x}$ is the vector $(\Delta\underline{e},\Delta\underline{f})$ of unknown corrections to the estimated solution and $[J]$ is the $(2n-2)\times(2n-2)$ matrix of differential terms commonly known as the Jacobian matrix.

The solution to (10) is

$$\underline{x} = [J]^{-1}\underline{b} \tag{11}$$

provided the Jacobian is non-singular.

Had the equations of the load flow problem been linear equations, solution (11) would be exact. However the functions in (6) are quadratic and the second order differential terms in equations (8) exist. The Newton Raphson algorithm ignores these terms under the assumption that given an initial estimate close to the actual solution, the numerical magnitudes of these terms are insignificant in comparison with the first order terms. Hence the solution given by equation (11) is inexact and more iterations are required to converge to the solution of satisfactory accuracy.

With iterative numerical methods, there is always the question of when satisfactory accuracy has been attained. In the N - R algorithm however, unlike other iterative algorithms e.g. the Gauss Seidel, without further effort it is possible to judge if the solution is of sufficient accuracy. The accuracy of the solution is reflected in the mismatch vector $\underline{b}$ in equation (10). It is to be noted that the vector consists of the differences of the specified injections and the actual injections as obtained from the solution. If these differences are within the practical or permissible variations at the corresponding buses in the network, then sufficient accuracy in the solution has been achieved.

2.3 The Constant Jacobian Approach

The algorithm developed in the previous section as it stands, is not well suited to the load flow calculations of large systems. For one, it requires the inverse of the Jacobian matrix and inverses are prohibitively laborious to calculate. It is also to be noted that the Jacobian matrix is dependent on the values of the voltages, hence the Jacobian matrix changes with every iteration and so the inverse has to be evaluated anew for each iteration. To save efforts it would be highly desirable to use the same inverse throughout the iterations if it can be proved to be mathematically sound.

Due to the quadratic nature of the algebraic relationship between the dependent and independent variables in the load flow equations it is mathematically tenable to use the same inverse Jacobian matrix throughout the iterations. This approach was expounded in [5] as an entirely new method in load flow calculations but was pointed out in [6] as but the conventional N - R load flow using a constant Jacobian matrix. The following demonstrates the validity of the constant Jacobian load flow.

The specified injections in the load flow equations can be expressed exactly in terms of the nodal voltage components to no higher than quadratic terms. Hence in each load flow equation the independent variable z_i can be expressed as a quadratic form as formulated in [7]:-

$$z_i = X^t[N]_i X$$

where $[N]_i$ is a sparse, symmetric matrix dependent only on the network parameters.

Since $[N]_i$ is symmetric,

$$\frac{\partial z_i}{\partial \underline{X}} = 2\underline{X}^t[N]_i$$

and

$$\frac{\partial^2 z_i}{\partial \underline{X}^2} = \frac{\partial}{\partial \underline{X}}\, 2\underline{X}^t[N]_i = 2[N]_i$$

Because z_i is a quadratic form, its expansion as a Taylor series will not have terms higher than the third, all higher terms being zero. So the Taylor expansion can be expressed exactly with the first three terms,

$$\begin{aligned} z_i(\underline{X}_o + \Delta\underline{X}) &= z_i(\underline{X}_o) + \left.\frac{\partial z_i}{\partial \underline{X}}\right|_{\underline{X}_o} \Delta\underline{X} + \tfrac{1}{2}\Delta\underline{X}^t \left[\frac{\partial^2 z_i}{\partial \underline{X}^2}\right]_{\underline{X}_o} \Delta\underline{X} \\ &= z_i(\underline{X}_o) + 2\underline{X}_o^t[N]_i\Delta\underline{X} + \Delta\underline{X}^t[N]_i\Delta\underline{X} \end{aligned}$$

Generalizing to include all the z_i's, the load flow equations can be compactly expressed as

$$\begin{aligned} \underline{Z}_{specified} &= \underline{Z}(\underline{X}_o + \Delta\underline{X}) \\ &= \underline{Z}(\underline{X}_o) + 2[\underline{J}(\underline{X}_o)]\Delta\underline{X} + [\underline{J}(\Delta\underline{X})]\Delta\underline{X} \end{aligned} \qquad (12)$$

where $\underline{Z} = (z_1, z_2, \ldots, z_{2n-2})^t$

$$\text{and} \quad [J(X_o)] = \begin{bmatrix} X_o^t[N]_1 \\ \vdots \\ X_o^t[N]_{2n-2} \end{bmatrix} = \frac{1}{2} \text{ Jacobian Matrix}$$

To solve this set of load flow equations one could use an iterative scheme

$$\Delta X = \frac{1}{2}[J(X_o)]^{-1}\{Z_{specified} - Z(X_o) - [J(\Delta X)]\Delta X\}$$

and iterate until ΔX remains constant.

The above approach may be considered as another load flow method, however it is equivalent to the Newton Raphson algorithm using a constant Jacobian matrix. This equivalence is shown as follows.

From (12),

$$2[J(X_o)]\Delta X = Z_{spec} - Z(X_o) - [J(\Delta X)]\Delta X$$

Expanding [J],

$$2\begin{bmatrix} X_o^t[N]_1 \\ \vdots \\ X_o^t[N]_{2n-2} \end{bmatrix}(X_{k+1} - X_o) = Z_{spec} - Z(X_o) - \begin{bmatrix} (X_k - X_o)^t[N]_1 \\ \vdots \\ (X_k - X_o)^t[N]_{2n-2} \end{bmatrix}(X_k - X_o)$$

$$2\begin{bmatrix} \underline{x}_0^t[N]_1\underline{x}_{k+1} \\ \vdots \\ \underline{x}_0^t[N]_{2n-2}\underline{x}_{k+1} \end{bmatrix} - 2\begin{bmatrix} \underline{x}_0^t[N]_1\underline{x}_0 \\ \vdots \\ \underline{x}_0^t[N]_{2n-2}\underline{x}_0 \end{bmatrix} = \underline{Z}_{spec} - \begin{bmatrix} \underline{x}_0^t[N]_1\underline{x}_0 \\ \vdots \\ \underline{x}_0^t[N]_{2n-2}\underline{x}_0 \end{bmatrix} -$$

$$\begin{bmatrix} \underline{x}_k^t[N]_1\underline{x}_k \\ \vdots \\ \underline{x}_k^t[N]_{2n-2}\underline{x}_k \end{bmatrix} + \begin{bmatrix} \underline{x}_0^t[N]_1\underline{x}_k \\ \vdots \\ \underline{x}_0^t[N]_{2n-2}\underline{x}_k \end{bmatrix} + \begin{bmatrix} \underline{x}_k^t[N]_1\underline{x}_0 \\ \vdots \\ \underline{x}_k^t[N]_{2n-2}\underline{x}_0 \end{bmatrix} - \begin{bmatrix} \underline{x}_0^t[N]_1\underline{x}_0 \\ \vdots \\ \underline{x}_0^t[N]_{2n-2}\underline{x}_0 \end{bmatrix}$$

$$2\begin{bmatrix} \underline{x}_0^t[N]_1\underline{x}_{k+1} \\ \vdots \\ \underline{x}_0^t[N]_{2n-2}\underline{x}_{k+1} \end{bmatrix} = \underline{Z}_{spec} - \begin{bmatrix} \underline{x}_k^t[N]_1\underline{x}_k \\ \vdots \\ \underline{x}_k^t[N]_{2n-2}\underline{x}_k \end{bmatrix} + 2\begin{bmatrix} \underline{x}_0^t[N]_1\underline{x}_k \\ \vdots \\ \underline{x}_0^t[N]_{2n-2}\underline{x}_k \end{bmatrix}$$

$$2\begin{bmatrix} \underline{x}_0^t[N]_1 \\ \vdots \\ \underline{x}_0^t[N]_{2n-2} \end{bmatrix}(\underline{x}_{k+1} - \underline{x}_k) = \underline{Z}_{spec} - \underline{Z}(\underline{x}_k)$$

$$\underline{x}_{k+1} - \underline{x}_k = [2[J(\underline{x}_0)]]^{-1}\{\underline{Z}_{spec} - \underline{Z}(\underline{x}_k)\}$$

The last equation above is the solution of the N - R algorithm using a constant Jacobian matrix .

Thus the above derivation establishes the validity of using the N - R algorithm without updating the Jacobian matrix during each iteration. The convergence rate may be slightly impaired and a few more iterations may be required but the overall effort is reduced.

2.4 Some Numerical and Computational Considerations

The load flow equations $\underline{b} = [J]\,\underline{x}$ when solved in the straightforward manner follows the procedure of finding the inverse of [J] and obtains the solution $\underline{x} = [J]^{-1}\,\underline{b}$. However this procedure is almost invariably never done in practice for large systems. In addition to evaluating the inverse matrix $[J]^{-1}$ which requires at least n^3 operations one has to further expend another n^2 operations for multiplying the inverse with $\underline{b}$. To solve a set of simultaneous equations, which actually is the case in the N - R solution of the load flow problem, the preferred method is the Gaussian elimination which, together with back-substitution, requires $\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$ operations [8]. A closer examination of the Gaussian elimination method will show that it is equivalent to factoring the coefficient matrix [J] into a product of a lower triangular matrix [L] and an upper triangular matrix [U] [9]. Though equivalent, explicitly factoring the coefficient matrix into lower and upper triangular matrices offers some advantages over direct Gaussian elimination, for example, with the factor matrices it requires just an additional back-substitution to solve the related set of simultaneous equation $[J]\underline{x} = \underline{b}_2$ whereas with direct Gaussian elimination every step has to be repeated anew.

There are different methods for factoring a matrix into its lower and upper triangular factor matrices [14]. The most suitable method for the work in this thesis is the Doolittle method (the Doolittle method is described in [14]) in which at the k^{th} step, the k^{th} rows of [L] and [U] are generated using the previous rows of [L] and [U] and the k^{th} row of the coefficient matrix. It is to be noted that the values of the rows previous to the k^{th} row of the coefficient matrix are not required in generating the k^{th} rows of [L] and [U]. The coefficient matrix, which in this case is the Jacobian matrix, is itself generated one row at a time. Hence after generating the k^{th} rows of [L] and [U], the k^{th} row of the Jacobian matrix will not be further required for subsequent rows of [L] and [U] and thus does not need to be stored. In short, no storage need to be allotted for storing the Jacobian matrix. For solving large systems this is highly desirable as storage conservation is essential. This brings us to the subject of sparsity programming which is a programming technique useful for conserving storage space in programs involving sparse matrices.

A matrix is classified as sparse if it has very few non-zero elements compared to the number of zero elements. In a large power system the bus admittance matrix is very sparse. The Jacobian matrix is sparse too. Because of its sparsity it would be wasteful to store the sparse matrices in the regular 2-dimensional arrays. The only significant data needed to be stored are the values of the non-zero elements and their row and column positions in the matrix. One other property of the bus admittance matrix that can be exploited for further storage saving purposes is its symmetry. By its symmetry only one half of the total non-zero elements needs to be stored.

Various methods have been presented for sparse matrix storage and retrieval [10,11,12]. In essence all methods are conceived with the intention of storing only the non-zero elements and the positional data concerning their corresponding locations in the matrix. Appendix B describes in detail the sparse matrix storage techniques employed in this thesis.

With the knowledge that sparsity in a matrix is desirable, naturally it is advantageous to preserve sparsity in matrices, when possible. This is one additional reason for L-U decomposition instead of finding the inverse of a matrix. Whereas the L and U matrices of a sparse matrix are sparse, the inverse of a sparse matrix is usually quite filled. Decomposing a sparse non-singular matrix into its L and U factors invariably introduces new non-zero elements. However there are many ways to minimize this additional fill-in. Theorems appearing in [13] predict how many fill-ins will result from choosing a certain element in the matrix as pivot in the factoring process. So, by exchanging rows and columns to obtain the right pivots giving the minimum fill-in, the optimum number of fill-in can be achieved.

With the complexities of the operations involved, such methods are only worthwhile for repeated calculations on the same network configuration, in which case the matrix structure is unchanged and the same sequence of pivots are used repeatedly. More practical methods with easier implementations but which do not generate the optimal fill-ins are also available [13]. One simple scheme is to choose pivots only among the diagonal elements of the matrix, which when applied on the bus admittance matrix is equivalent to re-numbering the original bus numbers in ascending order of the number of lines connecting to the bus [9,13].

In the L - U decomposition for the Jacobian matrix, re-numbering the buses as the above paragraph suggests, also helps to keep down the number of additional fill-ins due to the close relationship of the Jacobian matrix and the Y matrix. For decomposition of the Jacobian matrix there is one further way to decrease the fill-in. It will be recalled that the k^{th} row of the L and U matrices depend on the k^{th} rows of the Jacobian and the previous rows of the L and U matrices. It is evident then, that if the top rows are sparse, less fill-ins will result than had the top rows been full. In the Jacobian, rows corresponding to the voltage magnitude specifications at the PV buses are very sparse, being at most filled with two non-zero elements, namely $\partial V_i/\partial e_i$ and $\partial V_i/\partial f_i$. Hence in the interest of achieving greater sparsity in the L and U triangular matrices, it is desirable to place the rows corresponding to the voltage magnitude specifications as the very top rows. The set of equations (9) is thus rearranged as:-

$$|V|^2_{k\,spec} - V_k(\underline{e}_0, \underline{f}_0) = \left.\frac{\partial V_k}{\partial \underline{e}}\right|_{\underline{e}_0} \Delta \underline{e} + \left.\frac{\partial V_k}{\partial \underline{f}}\right|_{\underline{f}_0} \Delta \underline{f}$$

$$Q_{j\,spec} - Q_j(\underline{e}_0, \underline{f}_0) = \left.\frac{\partial Q_j}{\partial \underline{e}}\right|_{\underline{e}_0} \Delta \underline{e} + \left.\frac{\partial Q_j}{\partial \underline{f}}\right|_{\underline{f}_0} \Delta \underline{f}$$

$$P_{i\,spec} - P_i(\underline{e}_0, \underline{f}_0) = \left.\frac{\partial P_i}{\partial \underline{e}}\right|_{\underline{e}_0} \Delta \underline{e} + \left.\frac{\partial P_i}{\partial \underline{f}}\right|_{\underline{f}_0} \Delta \underline{f}$$

3. LOAD FLOW METHODS WITH NO SLACK BUS

3.1 The General Load Flow Problem

In the previous chapter the conventional load flow problem was discussed. It was realized that the real injection at one generation bus (to be called the slack or swing bus) should not be specified, its value being left free to be determined by the solution of the load flow. Had the slack bus real injection been specified at the onset of the load flow problem, the system would be overspecified and an inconsistent or redundant set of equations resulted. Hence the existence of the slack bus is dictated by the mathematical constraint implicit in the formulation of the set of load flow equations. In the physical system there is no slack bus as such, no one special bus is designated as a slack bus. Rather, the set point of each generation bus is fixed by the dispatch or load frequency control centers.

The existence of the slack bus in the load flow problem is thus recognised as being imposed by the mathematical formulation of the problem rather than by the physical system. In this thesis load flow models minus the artifice of a slack bus are investigated by re-working the formulation of the load flow problem.

In the load flow problem, the state variables are the nodal voltages $(e_1,\ldots,e_n,f_1,\ldots,f_n)^t = \mathbf{x}$ where e_i and f_i are the real and imaginary components respectively of the nodal voltages at node i and n is the number of buses in the system. All other variables of interest y_i in the system (e.g. bus injections or line flows) can be expressed in terms of the nodal voltages, i.e.

$$y_i = g_i(\mathbf{x})$$

Now if we choose 2n of these y_i's, ensuring that they are all independent and assign them values, we will have a consistent set of 2n algebraic equations in the unknown $\mathbf{x}$. These 2n equations constitute a valid load flow formulation.

3.2 The Floating System Voltage load Flow

In the formulation with a slack bus the total real power generation is not constrained. If all real injections are specified then the system will have a constraint on the total real injection. This total sum of real injections into the network is equal to the transmission loss. Thus the specification of all the injections is equivalent to the specfication of the transmission loss. If this sum is negative, at once the problem defined as such has no solution because demand had exceeded generation. However, with the transmission loss in the network correctly defined, which constraint can be removed? Due to the close relationship between system voltage level and transmission loss, one logical choice is to allow the system voltage level to float until such a value of transmission loss is obtained.

With this floating system voltage formulation then, the constraints on the voltage magnitudes at the PV buses are relaxed. A nominal value of voltage magnitude is specified for each PV bus but the actual value is allowed to vary proportionately to its nominial voltage to satisfy the transmission loss specified. This nominial voltage is the same value given to the PV buses in the conventional formulation. If the transmission loss have been realistically estimated in specifying the real injections, then the voltage magnitudes should not deviate appreciably from their nominal values.

We shall discuss the floating system voltage load flow in rectangular coordinates (for the treatment in polar co-ordinates refer to Appendix A). With the system voltage level floating, the voltage magnitudes at the PV buses are no longer specified exactly. Instead, their nominal values are given and the actual voltage at the bus is allowed to vary by a factor ρ which is determined by how much transmission loss had been estimated, as implied by specifying all the real powers. Thus the load flow equations are:-

$$P_{i\,spec} = P_i(\underline{x})$$
$$\rho^2 |V|^2_{i\,spec} = V_i(\underline{x}) \qquad i=1,\ldots,m$$
$$P_{j\,spec} = P_j(\underline{x})$$
$$Q_{j\,spec} = Q_j(\underline{x}) \qquad j=m+1,\ldots,n$$

where for ease of notation, it is assumed that the first m buses are PV buses in the system of n buses. The unknowns are ρ, the system voltage factor and $\underline{x}$, the vector of nodal voltages.

To be able to apply the conventional load flow solution methods with as little modifications as possible to this new set of load flow equations (thus saving efforts of doing everything from scratch), a simple transformation of variables is necessary. If we define $\underline{x}' = \frac{1}{\rho}\underline{x}$ then since $P_i(\underline{x})$ and $Q_i(\underline{x})$ are quadratic functions of $\underline{x}$ they can be expressed in terms of $\underline{X}'$ as $\rho^2 P_i(\underline{X}')$ and $\rho^2 Q_i(\underline{X}')$ and the set of equations can be written as:-

$$
\begin{aligned}
P_{i\,spec} &= \rho^2 P_i(\underline{X}') & & \\
|V|^2_{i\,spec} &= V_i(\underline{X}') & & i=1,\dots,m \\
P_{j\,spec} &= \rho^2 P_j(\underline{X}') & & \\
Q_{j\,spec} &= \rho^2 Q_j(\underline{X}') & & j=m+1,\dots,n
\end{aligned}
$$

The above structure of the load flow equations is almost identical to the conventional formulation Eqs. (6); at the L.H.S. are the specified quantities while at the R.H.S., functions of the unknown nodal voltages. One additional unknown, the factor ρ is introduced and so is one more equation. Altogether there are (2n-1) unknowns with (2n-1) equations after solving independently the voltage equation of the reference bus. The system can be solved with the Newton Raphson algorithm in steps similar to the conventional load flow. The Jacobian matrix is now (2n-1)x(2n-1) and is of the form

$$
\begin{bmatrix}
\partial \rho^2 P_i/\partial \underline{X}' & \partial \rho^2 P_i/\partial \rho \\
\partial V_i/\partial \underline{X}' & \partial V_i/\partial \rho \\
\partial \rho^2 P_j/\partial \underline{X}' & \partial \rho^2 P_j/\partial \rho \\
\partial \rho^2 Q_j/\partial \underline{X}' & \partial \rho^2 Q_j/\partial \rho
\end{bmatrix}
$$

This will require minor modifications to the Jacobian matrix generation algorithm of the conventional load flow. In the conventional load flow the Jacobian is (2n-2)x(2n-2) whereas here, it is (2n-1)x(2n-1). The extra row is due to inclusion of the real power equation for the bus designated as slack bus in the conventional load flow. However its evaluation poses no additional problem, it requires the same algorithm as used for evaluating the rows corresponding to the real powers equations. The extra column introduced is due to the unknown ρ. The differential terms of this column are evaluated as:-

$$\frac{\partial}{\partial \rho} \rho^2 P_i = 2\rho P_i$$

$$\frac{\partial}{\partial \rho} V_i = 0$$

$$\frac{\partial}{\partial \rho} \rho^2 Q_j = 2\rho Q_j$$

Save for these modifications to the Jacobian matrix the conventional load flow program could be used to solve this new formulation of the load flow problem.

3.2.1 An Iterative Approach to the Solution

The floating system voltage load flow can also be implemented as an iterative procedure using the conventional load flow algorithm. In the floating system voltage load flow, all the real generations are specified whereas in the conventional case the real injection of the slack is left unspecified. If at the solution of a conventional load flow the real generation of the slack bus does not agree with the value specified, it implies the transmission loss as calculated does not match the specified transmission loss. The calculated transmission loss can be corrected by an adjustment of the nodal voltages in the system.

With the system nodal voltages expressed as a vector $\underline{x}$ of the real and imaginary components, the transmission loss can be written as a quadratic form $P_L = \underline{x}^t[N]\underline{x}$. Matching the real injection at the slack bus to the value specified is equivalent to have the transmission loss as implicitly specified, match the transmission loss as calculated by the conventional load flow, i.e.

$$P_{L\,spec} = P_{L\,cal} = \underline{x}^t[N]\underline{x}$$

If the two losses, calculated and specified do not match, for adjustment, the nodal voltages can be scaled by a factor ρ so that the loss will match, that is, to have

$$\begin{aligned} P_{L\,spec} &= (\rho X)^t [N] (\rho X) \\ &= \rho^2 X^t [N] X \\ &= \rho^2 P_{L\,cal} \end{aligned}$$

So the factor ρ can be calculated from

$$\rho^2 = \frac{P_{L\,spec}}{P_{L\,cal}}$$

However in practice, it is not possible to adjust all the nodal voltages, only the voltage magnitudes at the PV buses are controllable. So the floating system voltage load flow can be implemented by iteratively solving the conventional load flow, comparing the specified loss with the calculated loss and adjusting the voltage magnitudes at the PV buses, repeating the procedure until the calculated loss matches the specified loss, at which time the real injection at the bus designated as the slack bus would have attained the value as specified.

3.3 The Participation Factors Load Flow

In the conventional load flow formulation the slack bus is introduced to allow total real generation to remain unspecified. However a slack bus is not really needed to achieve this purpose. One can still have the total real power generation left unspecified and yet not require a generation bus to be treated differently from all the others. This could be achieved by allocating a fractional share of the total real generation rather than an absolute value of generation to each generation bus. By this formulation, the total generation is still left unspecified but all the generation buses are treated equally. The detailed formulation is as follows:

Let α_i where $i = 1,\ldots,m$ be the fractional share each generation bus contributes to the total generation. These α_i's are specified quantities. The other specified quantities are $|V|_i$, P_j and Q_j, where $i = 1,\ldots,m$ and $j = m+1,\ldots,n$; m is the number of generation buses and n the total number of buses in the system. Let $\underline{x}$ be defined as a vector of the nodal voltages $(e_1,\ldots,e_n,f_1,\ldots,f_n)^t$. It is assumed without loss of generality that bus 1 is the reference bus with $f_1 = 0$. The power injections, real and reactive, at any bus i and the total power generated in the system are dependent functions of the state variable $\underline{x}$, represented by $P_i(\underline{x})$, $Q_i(\underline{x})$ and $P_T(\underline{x})$ respectively. With these definitions then, the power generated at a generation bus i is $\alpha_i P_T(x)$ and it should equal the network injection plus the demand at that bus. The equations representing the system are:-

$$\alpha_i P_T(\underline{x}) = P_i(\underline{x}) + P_{i\,demand} \tag{13}$$

$$|V|^2_{i\,spec} = V_i^2(\underline{x}) = e_i^2 + f_i^2 \qquad i = 1,\ldots,m \tag{14}$$

$$P_{j\,specified} = P_j(\underline{x}) \tag{15}$$

$$Q_{j\,specified} = Q_j(\underline{x}) \qquad j = m+1,\ldots,n \tag{16}$$

In the set of equations (14), the one corresponding to the reference bus can be solved separately; with $f_{ref} = 0$ the equation $|V|^2_{ref} = e^2_{ref} + f^2_{ref}$ is trivial to solve. Although it may not be apparent, the set of equations (13) is redundant; of the m equations in (13) any one of them can be obtained from the other (m - 1) equations. This can be shown by the following argument.

Since the α_i's are the fractional shares, they sum up to unity. Also, the total generation is the sum of the generations at all the generation buses,

$$P_T(\underline{x}) = \sum_{k=1}^{m} \{P_k(\underline{x}) + P_{k\,demand}\} \tag{17}$$

Thus, subtracting any $(m - 1)$ equations in (13) from both sides of equation (17) we have,

$$P_T(\underline{X}) - \sum_{k \neq i} \alpha_k P_T(\underline{X}) = \sum_k [P_k(\underline{X}) + P_{k\,dem}] - \sum_{k \neq i} [P_k(\underline{X}) + P_{k\,dem}]$$

The above equation can be simplified into

$$P_T(\underline{X})\,[1 - \sum_{k \neq i} \alpha_k] = \sum_k [P_k(\underline{X}) + P_{k\,dem}] - \sum_{k \neq i} [P_k(\underline{X}) + P_{k\,dem}]$$

which transforms to the equation corresponding to bus i

$$\alpha_i P_T(\underline{X}) = P_i(\underline{X}) + P_{i\,dem}$$

With these considerations then, the system has $(2n - 2)$ equations and $(2n - 2)$ unknown voltage components (f_{ref} is zero and e_{ref} has been separately solved). These constitute a consistent set of equations and is the formulation of the new load flow algorithm to be called the Participation Factors Load Flow. The set of load flow equations defined are:

$$\alpha_{i\,spec} = \frac{P_i(\underline{X}) + P_{i\,dem}}{P_T(\underline{X})}$$

$$
\begin{aligned}
|V|^2_{j\,spec} &= V_j(\underline{X}) \\
P_{k\,spec} &= P_k(\underline{X}) \\
Q_{k\,spec} &= Q_k(\underline{X})
\end{aligned}
\qquad (18)
$$

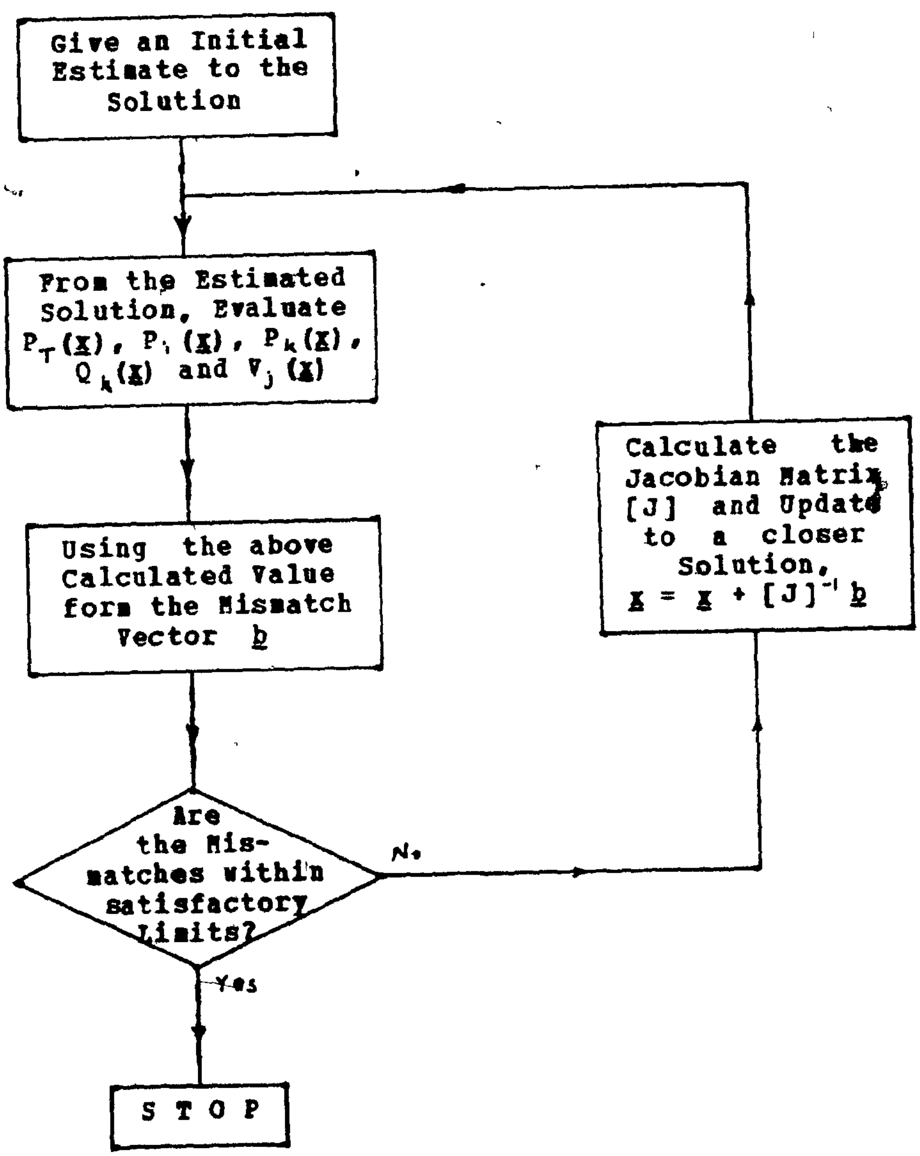
where $i = 1,\ldots,m$ $\quad i \neq g \quad$ g can be any value between 1 and m; $j = 1,\ldots,m$ $\quad j \neq$ ref; $k = m+1,\ldots,n$.

This new load flow algorithm can be implemented with only slight modifications to the conventional load flow algorithm. In the conventional load flow, the system of load flow equations, Eqs. (6), are:-

$$
\begin{aligned}
P_{i\,spec} &= P_i(\underline{X}) \\
|V|^2_{i\,spec} &= V_i(\underline{X}) \\
P_{j\,spec} &= P_j(\underline{X}) \\
Q_{j\,spec} &= Q_j(\underline{X})
\end{aligned}
\qquad (6)
$$

where $i = 2,\ldots,m$ and $j = m+1,\ldots,n$

It will be noticed that the 2 sets of equations Eqs. (6) and Eqs. (18) are identical in form, with the specified quantities on the L.H.S. and functions of the state vector $\underline{x}$ on the R.H.S. Hence a solution algorithm for the new load flow formulation can be chosen among those of the conventional load flow. The Newton Raphson algorithm for this no slack bus load flow formulation is outlined below.



The Jacobian matrix in this no slack bus formulation is:-

$$\frac{\partial}{\partial \underline{x}} \begin{bmatrix} \dfrac{P_i(\underline{X}) + P_{i\,demand}}{P_T(\underline{X})} \\ V_j(\underline{X}) \\ P_k(\underline{X}) \\ Q_k(\underline{X}) \end{bmatrix}$$

It is exactly the same Jacobian matrix of the conventional load flow except for the (m - 1) rows corresponding to the real injections of the generation buses. The differential terms in these rows are

$$\frac{1}{P_T(\underline{X})} \frac{\partial}{\partial \underline{X}} P_i(\underline{X}) \quad - \quad \frac{P_i(\underline{X})}{P_T^2(\underline{X})} \frac{\partial}{\partial \underline{X}} P_T(\underline{X})$$

The total power generation $P_T(\underline{X})$ can be expressed in terms of the total demand plus the transmission losses.

$$P_T(\underline{X}) = P_{Demand} + P_L(\underline{X})$$

It is a valid assumption that the losses are not greatly affected by small changes in the nodal voltages around the operating point. This can be justified by the following analytical considerations: The loss in the network can be expressed as

$$P_L(\underline{x}) = \underline{x}^t[N]\underline{x}$$

where $[N]$ is $\begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix}$, G being the real part of the bus admittance matrix. The differential term $\frac{\partial P_L}{\partial \underline{x}}$ then, is

$$\frac{\partial P_L}{\partial \underline{x}} = 2\underline{x}^t[N]$$

One property of the bus admittance matrix is that $g_{ii} = -\sum_j g_{ij}$. At the operating point the real part of the nodal voltages are of approximately equal order of magnitude and so are the imaginary parts. From this then, $\underline{x}^t\begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix}$ is of a small magnitude. Also $P_T^2 \gg P_i$. Thus the term $\frac{P_i(\underline{x})}{P_T^2(\underline{x})} \frac{\partial}{\partial \underline{x}} P_T(\underline{x}) = \frac{P_i(\underline{x})}{P_T^2(\underline{x})} \frac{\partial}{\partial \underline{x}} P_L(\underline{x})$ can be considered negligible.

Hence the differential terms in those rows in the Jacobian matrix corresponding to the real injections of the generation buses can be simplified to $\frac{1}{P_T(\underline{x})} \frac{\partial}{\partial \underline{x}} P_i(\underline{x})$ which is $\frac{1}{P_T(\underline{x})}$ times the corresponding terms in the Jacobian matrix of the conventional load flow. If we bring this factor to the other side of the corresponding equations (i.e. as in the original derivation $\alpha_i P_T = P_i(x) + P_{i\,dem}$) then the conventional Jacobian matrix can be used without modification.

4. POSSIBLE APPLICATIONS OF THE NEW LOAD FLOW FORMULATIONS

4.1 The Floating System Voltage Case

The floating system voltage load flow described in the previous chapter is a valid representation of a power system. The voltage magnitudes at the generation buses in an actual power system are indeed variable within permissible ranges by the generator exciters. Moreover all generation units are dispatched, there is no generation unit left undispatched to take up the slack. We shall examine an application of this load flow formulation in the area of economic dispatch. Being able to specify all the real generations independently as this load flow formulation prescribes, offers a simpler approach to economic dispatch. A brief review of the economic dispatch problem is in order before we proceed to elaborate on this application of the new load flow formulation.

In economic dispatch the goal is to obtain a generation plan such that the customers' demand is satisfied with the minimum generation cost. Hence the economic dispatch problem can be set up as an optimization problem:

$$\text{Min } C = \sum_{i=1}^{m} C_i(P_i)$$

$$\text{subject to } \sum_{i=1}^{m} P_i = P_D + P_L$$

where m is the total number of generation units

P_D is the total demand

and P_L the transmission loss.

The neccessary conditions for the optimum economic dispatch are given by the modified coordination equations

$$\frac{\dfrac{dC_i}{dP_i}}{1 - \dfrac{\partial P_L}{\partial P_i}} = \frac{dC_s}{dP_s} \qquad (19)$$

for $i = 1, \ldots m$ $i \neq s$ where s is the slack generation. Hence to solve the modified coordination equations it is necessary to evaluate $\frac{\partial P_L}{\partial P_i}$. In the conventional approach P_L is an unknown and has to be calculated from a load flow or approximated through a loss formula.

However in the floating system voltage load flow formulation enunciated, it is possible to specify the value of the transmission loss a priori at the expense of freeing the system voltage level. Thus with a specified value of transmission loss, say, a certain percentage λ of the total demand, i.e. $P_L = \frac{\lambda}{100} P_D$ the economic dispatch problem can be formulated as

$$\min \quad C = \sum C_i(P_i)$$

$$\text{subject to} \quad \sum P_i = P_D(1 + \lambda/100)$$

The necessary conditions for the optimum in this case are given by the condition of equal incremental cost of generation

$$\frac{dC_i}{dP_i} = \text{constant} \tag{20}$$

for $i = 1,\ldots,m$. It is obvious that these equations are simpler to solve than Eqs. (19).

In practice the cost of generation for each unit can be approximated by an empirical quadratic formula

$$C_i = a_i P_i + b_i P_i^2$$

which yields a linear incremental cost curve

$$\frac{dC_i}{dP_i} = a_i + 2b_i P_i \tag{21}$$

Under such assumption of quadratic generation cost the economic dispatch problem with transmission loss fixed a priori, say, at λ % of the total demand P_D, can be solved readily by analytical means.

From (20) and (21) it is required that

$$a_i + 2b_i P_i = \text{constant} = K \text{ (say)} \tag{22}$$

subject to the constraint

$$\sum P_i = P_D + P_L$$
$$= P_D (1 + \lambda/100)$$
$$= P_T$$

From (22)

$$P_i = \frac{K}{2b_i} - \frac{a_i}{2b_i}$$

To satisfy the constraint

$$\sum \left(\frac{K}{2b_i} - \frac{a_i}{2b_i} \right) = P_T$$

$$K \sum \frac{1}{2b_i} - \sum \frac{a_i}{2b_i} = P_T$$

$$K = \left\{ P_T + \sum \frac{a_i}{2b_i} \right\} \Big/ \sum \frac{1}{2b_i}$$

from which the optimum generations are

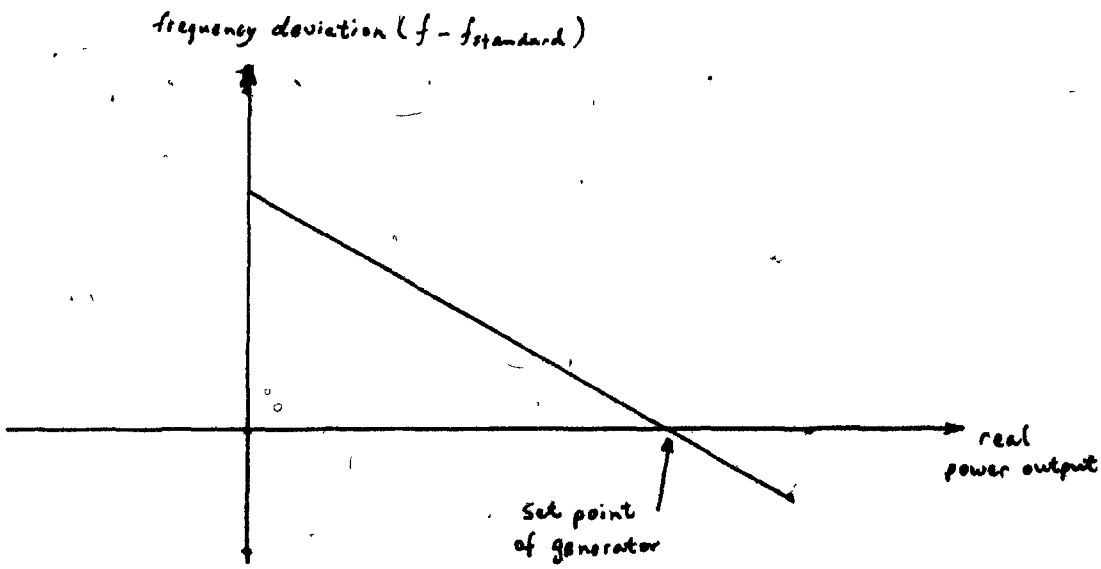
$$P_i = \frac{K}{2b_i} - \frac{a_i}{2b_i}$$

The above solution approach to the economic dispatch problem greatly simplifies the numerical work needed, compared to solving the coordination equations (19). With Eqs. (19) the differential term $\frac{\partial P_L}{\partial P_i}$ requires substantial computational effort to evaluate. An issue of contention however, is, to what degree of confidence can one assign the value of λ% of total demand as the transmission loss. With experience from operating the network, one usually acquires an idea of the magnitude of the transmission loss. Furthermore, it is recently reported in [15] that the optimum dispatch is not very sensitive to the transmission loss evaluated hence an experienced estimate of the loss is sufficient.

4.2 The Participation Factors Case

The manner in which the α_i's are defined may have appeared arbitrary but this formulation of the load flow problem is not without physical basis or applicability. Given the proper interpretations, this formulation can be adapted for modelling systems with consideration for the effects of turbine governors on the generation units. A detailed exposition of the performance of governor-controlled turbine generators is given in [16].

For each governor there is a characteristic called the droop. This quantity relates the change of real power output with the deviation from standard frequency. As any power mismatch between generation and demand is quickly manifested in a deviation from standard frequency, this droop characteristic is a primary mechanism to restore the balance between generation and demand.



A typical droop characteristic of a turbine governor.

We shall illustrate how this droop characteristic helps to restore the balance between generation and demand. For simplicity, we consider a two bus system with a generator delivering power to a load. The system is originally in steady state. We shall examine what happens when the load increases by a small amount ΔP. Since there is no change in the input mechanical power, the generator is now delivering more power than it is receiving from its prime mover. This, it can only achieve through tapping its stored rotational kinetic energy, resulting in a drop in its rotational speed and consequently its frequency. As the speed decreases, the speed control mechanism, its response characterised by the droop, will go into effect. We notice from the figure above that as the frequency drops, generation is increased. This increase in generation helps to offset the increase in the load and eventually the system will arrive at a new steady state with a lower frequency but an increased generation matching the increased load.

As the slope of the droop characteristic governs how much extra load each generator will take up following a deviation from standard frequency, the α_i's as defined for each generation bus in this Participation Factors Load Flow, can be interpreted as a representation of the slope of the droop characteristic. In this case then, the real power equation corresponding to the generation unit i with droop characteristic D_i (real power p.u. / frequency deviation) is

$$P_{i\,set\,point} + D_i \Delta f = P_i(\underline{x}) + P_{i\,demand}$$

or

$$D_i \Delta f = P_i(\underline{x}) + (P_{i\,demand} - P_{i\,set\,point})$$

which is intrinsically the real power equation

$$\alpha_i P_T = P_i(\underline{x}) + P_{i\,demand}$$

corresponding to a generation bus i in this load flow formulation.

A potential application of this load flow formulation, then, is in the following situation: The set points of all generation units have been fixed. However the total power demand including the network loss does not equal the sum of the generations established by the set points. Through the control actions of the droop characteristics, the generators will operate at generations levels different from those values indicated on the set points. This load flow formulation, with the α_i's properly assigned values to model the droop characteristics, will be able to predict the new operating points of the generation units.

5. RESULTS AND DISCUSSION

5.1 General Observation

In the course of this investigation, many sample power systems were studied, including the AEP 14 bus system, the 25 bus system in [17] and the AEP 118 bus system. The 118 bus system was an especially fruitful experience in computer programming. With large systems, programming techniques are very important. Poorly written programs may suffice for a small system study without any noticeable effect of unduly excessive computation time but large systems are quite another matter. Inefficient programs on large systems very quickly become evident as execution time soars. The execution times for the 118 bus system were an order of magnitude different between the author's first attempts and the final program. In its final form, the program takes less than 5 seconds on an Amdahl V7 machine to L - U decompose the 234 x 234 Jacobian matrix.

5.2 Detailed Study of a 5 Bus System

For the purpose of discussion we shall use a 5 bus system. The power system studied was taken from [17] which was a modification of the sample system used in a load flow example in a text-book [18]. The network configuration is shown in the following figure, the line data is as given in Table I and the base case generation and load data is in table II.

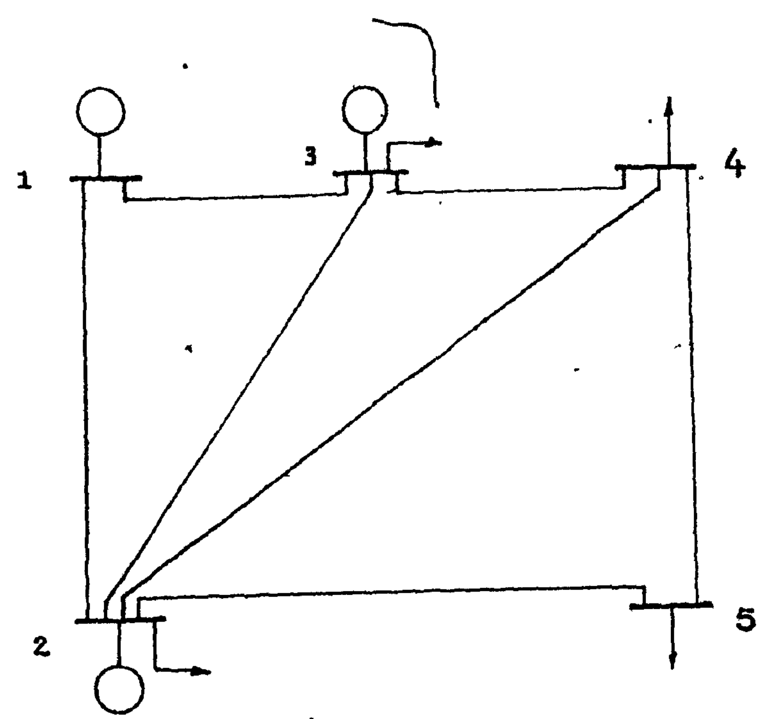


Table I Impedance and Line Charging Data

Line	Resistance	Reactance	Line Charging
1 - 2	0.02	0.06	0.030
1 - 3	0.08	0.24	0.025
2 - 3	0.06	0.18	0.020
2 - 4	0.06	0.18	0.020
2 - 5	0.04	0.12	0.015
3 - 4	0.01	0.03	0.010
4 - 5	0.08	0.24	0.025

Table II Base Case Generation & Loading Data

Bus NO.	Bus Type	Generation	Load	Voltage Magnitude
1	Generation	0.448+j0.058	0.00+j0.00	1.06
2	Generation	0.692+j0.043	0.20+j0.10	1.05
3	Generation	0.527+j0.033	0.45+j0.15	1.04
4	Load		0.40+j0.05	
5	Load		0.60+j0.10	

To establish the base case, the load flow problem with the above data was solved using the conventional formulation with the Newton Raphson algorithm. Bus 1 was designated the slack bus as well as the reference bus i.e. the voltage angle of bus 1 is 0 and its real generation is left unspecified. The results appear as follows.

CONVERGES TO WITHIN 0.00010 FOR THE MAXIMUM MISMATCH IN 4 ITERATIONS

		VOLTAGE		GENERATION		DEMAND		MISMATCH	
		MAGNITUDE	ANGLE	REAL	REACTIVE	REAL	REACTIVE	$Q/\|V\|^2$	POWER
BUS	1	1.060	0.0	0.448	0.058	0.0	0.0		
	TO BUS 3			0.158	0.010				
	TO BUS 2			0.290	0.040				
BUS	2	1.050	-0.81	0.692	0.043	0.200	0.100	-0.000	-0.000
	TO BUS 5			0.494	0.055				
	TO BUS 4			0.172	-0.001				
	TO BUS 3			0.114	-0.001				
	TO BUS 1			-0.286	-0.110				
BUS	3	1.040	-1.82	0.527	0.034	0.450	0.150	-0.000	0.000
	TO BUS 4			0.347	-0.017				
	TO BUS 2			-0.113	-0.041				
	TO BUS 1			-0.157	-0.050				
BUS	4	1.037	-2.38	0.0	0.0	0.400	0.050	-0.000	-0.000
	TO BUS 5			0.117	-0.010				
	TO BUS 3			-0.346	-0.002				
	TO BUS 2			-0.171	-0.033				
BUS	5	1.024	-3.81	0.0	0.0	0.600	0.100	-0.000	-0.000
	TO BUS 4			-0.115	-0.040				
	TO BUS 2			-0.485	-0.060				

TOTAL SYSTEM LOSS = 0.017

Poor Copy

5.2.1 The Participation Factors Load Flow

The system was next solved using the no slack bus formulation specifying the fractional share of the total generation for each generation bus. Though other values could be used for the α_i's, in the interest of easy verification with the conventional N - R base case, the generations for bus 1, 2 and 3 are specified in the ratio 448 : 692 : 527, the real generation values obtained from the converged conventional load flow. The results as expected, agree with those obtained from the conventional load flow.

CONVERGES TO WITHIN 0.00010 FOR THE MAXIMUM MISMATCH IN 5 ITERATIONS

		VOLTAGE		GENERATION		DEMAND		MISMATCH	
		MAGNITUDE	ANGLE	REAL	REACTIVE	REAL	REACTIVE	$Q/\|V\|^2$	POWER
BUS	1	1.060	0.00	0.448	0.058	0.000	0.000	0.000	0.000
TO BUS	3			0.159	0.010				
TO BUS	2			0.290	0.049				
BUS	2	1.050	-0.81	0.692	0.043	0.200	0.100	0.000	-0.000
TO BUS	5			0.494	0.055				
TO BUS	4			0.172	-0.031				
TO BUS	3			0.114	-0.001				
TO BUS	1			-0.283	-0.110				
BUS	3	1.040	-1.82	0.527	0.034	0.450	0.150	0.000	-0.000
TO BUS	4			0.347	-0.017				
TO BUS	2			-0.113	-0.041				
TO BUS	1			-0.157	-0.059				
BUS	4	1.027	-2.38	0.000	0.000	0.400	0.050	0.000	-0.000
TO BUS	5			0.117	-0.010				
TO BUS	3			-0.346	-0.002				
TO BUS	2			-0.171	-0.039				
BUS	5	1.024	-3.81	0.000	0.000	0.600	0.100	0.000	-0.000
TO BUS	4			-0.115	-0.040				
TO BUS	2			-0.495	-0.060				

TOTAL SYSTEM LOSS = 0.017

Poor Copy

This load flow formulation is also suited to model how the governors will react to distribute the additional generation required if total demand plus transmission loss does not equal total generation. In the following case presented, the generations were set to satisfy the total demand without any allocation for the transmission loss. The set points of the generators were inputted as negative values of real demand. The generations are to bear the transmission loss in the ratio 1 : 1 : 1, i.e. equally. From the results shown it is seen that the total transmission loss of 0.018 is distributed equally among the three generation buses each contributing 0.006.

CONVERGES TO WITHIN 0.00010 FOR THE MAXIMUM MISMATCH IN 5 ITERATIONS

		VOLTAGE.		GENERATION		DEMAND		MISMATCH	
		MAGNITUDE	ANGLE	REAL	REACTIVE	REAL	REACTIVE	$Q/\|V\|^2$	POWER
BUS	1	1.060	0.00	0.006	0.009	-0.600	0.000	0.000	0.000
TO BUS	3			0.155	0.011				
TO BUS	2			0.451	-0.002				
BUS	2	1.050	-1.36	0.006	0.139	-0.200	0.100	0.800	-0.000
TO BUS	5			0.471	0.061				
TO BUS	4			0.126	0.013				
TO BUS	3			0.056	0.018				
TO BUS	1			-0.448	-0.054				
BUS	3	1.040	-1.77	0.006	-0.009	-0.200	0.150	0.000	-0.000
TO BUS	4			0.415	-0.078				
TO BUS	2			-0.056	-0.061				
TO BUS	1			-0.153	-0.061				
BUS	4	1.037	-2.45	0.000	0.000	0.400	0.050	0.000	-0.000
TO BUS	5			0.139	-0.017				
TO BUS	3			-0.414	0.022				
TO BUS	2			-0.125	-0.054				
BUS	5	1.024	-4.21	0.000	0.000	0.600	0.100	0.000	-0.000
TO BUS	4			-0.138	-0.031				
TO BUS	2			-0.462	-0.062				

TOTAL SYSTEM LOSS = 0.018

Poor Copy

5.2.2 The Floating System Voltage Load Flow

The floating system voltage load flow formulation was applied to the system in which the voltage in the system was allowed to vary to satisfy the transmission loss specified. For illustration, the base case data was used except for a change being that the total generation was reduced by 0.1% of the original values. The results were as presented below.

CONVERGES TO WITHIN 0.00010 FOR THE MAXIMUM MISMATCH IN 6 ITERATIONS

		VOLTAGE		GENERATION		DEMAND		MISMATCH	
		MAGNITUDE	ANGLE	REAL	REACTIVE	REAL	REACTIVE	$Q/\|V\|^2$	POWER
BUS	1	1.120	0.00	0.448	0.081	0.000	0.000	0.000	0.000
	TO BUS 3			0.158	0.016				
	TO BUS 2			0.289	0.065				
BUS	2	1.110	-0.71	0.691	0.018	0.200	0.100	0.000	-0.000
	TO BUS 5			0.493	0.049				
	TO BUS 4			0.172	0.001				
	TO BUS 3			0.114	0.003				
	TO BUS 1			-0.288	-0.135				
BUS	3	1.099	-1.59	0.526	-0.007	0.450	0.150	0.000	-0.000
	TO BUS 4			0.346	-0.034				
	TO BUS 2			-0.113	-0.050				
	TO BUS 1			-0.157	-0.073				
BUS	4	1.097	-2.09	0.000	0.000	0.400	0.050	-0.000	0.000
	TO BUS 5			0.116	-0.018				
	TO BUS 3			-0.345	0.013				
	TO BUS 2			-0.171	-0.045				
BUS	5	1.086	-3.39	0.000	0.000	0.600	0.100	-0.000	0.000
	TO BUS 4			-0.115	-0.039				
	TO BUS 2			-0.485	-0.061				

TOTAL SYSTEM LOSS = 0.015

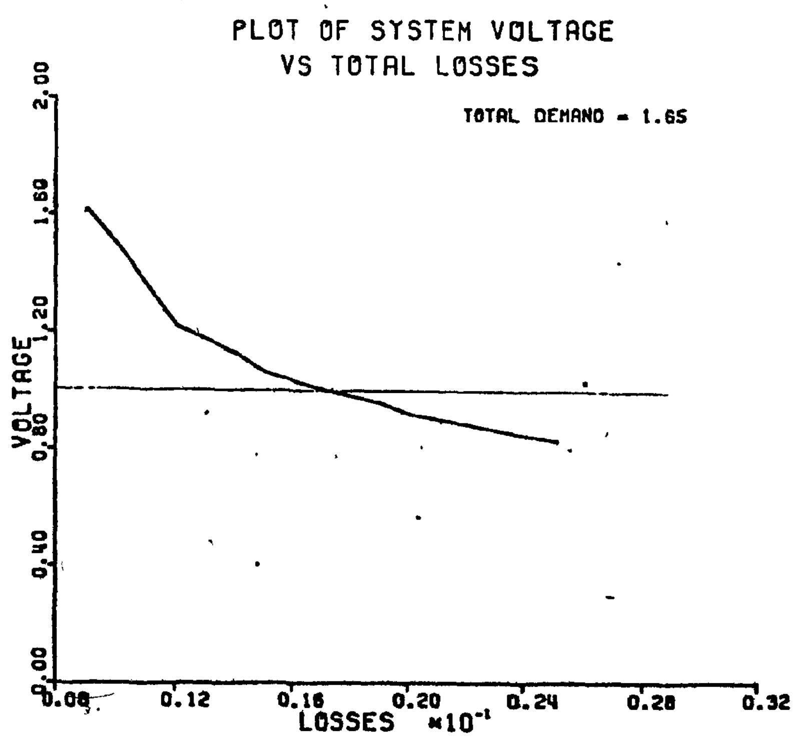
SYSTEM VOLTAGE FLOATING FACTOR = 1.0569

Poor Copy

Comparing the results with the base case, it is seen that all the voltage magnitudes are higher than before. This is necessary because by specifying 0.1% less generation but keeping the demand as before, the reduction in generation must be made up for by a similar reduction in transmission loss. To achieve this reduced loss, the system voltage must rise. In this case it rose by 5.69%, i.e. all the generation bus voltages rose by 5.69% of their original or nominal values.

Convergence difficulties arose when the Jacobian matrix was not updated with every iteration. The proof validating the constant Jacobian approach no longer applies in this case as the power equations $z_i = \rho^2 \underline{X}^t [N]_i \underline{X}$ cannot be expressed as a quadratic form of the unknowns $(\rho, \underline{X})^t$. In view of this the iterative solution described in Section 3.2.1 is more attractive since the initial Jacobian matrix could be applied without change throughout the following iterations.

With the floating system voltage load flow formulation, it is possible to investigate how the system voltage level will vary with transmission loss. In the investigation carried out, different values of transmission loss are specified and it is noted that low loss requires high system voltages while high loss has low system voltages. The results are summarized in the plot below.



Since this load flow formulation allows all real generations to be specified, once the transmission loss are fixed, say at a certain percentage of the total demand, economic dispatch can be performed relatively easily as described in Section 4.1.

Obviously the total cost of meeting the demand will depend on the transmission loss specified which in turn is dependent on the system voltage level. However with experience acquired through simulation of the network or actual operation of the existing power system, the transmission loss can be specified at a value which will not involve excessively high or abnormal voltage levels. An example of the economic dispatch followed by the load flow solution to the minimum cost generation plan is given below. The system is the same 5 bus system and the cost data obtained from the same source [17], is as follows:-

$$C_1 = 200P_1 + 60P_1^2$$
$$C_2 = 150P_2 + 75P_2^2$$
$$C_3 = 180P_3 + 70P_3^2$$

COST OF GENERATION = 355.53 AT A TRANSMISSION LOSS OF 0.010 OF TOTAL DEMAND

CONVERGES TO WITHIN 0.00010 FOR THE MAXIMUM MISMATCH IN 6 ITERATIONS

		VOLTAGE		GENERATION		DEMAND		MISMATCH	
		MAGNITUDE	ANGLE	REAL	REACTIVE	REAL	REACTIVE	$Q/\|V\|^2$	POWER
BUS	1	1.078	0.00	0.448	0.065	0.000	0.000	0.000	0.000
	TO BUS 3			0.158	0.012				
	TO BUS 2			0.290	0.053				
BUS	2	1.068	-0.78	0.692	0.036	0.200	0.100	0.000	-0.000
	TO BUS 5			0.493	0.053				
	TO BUS 4			0.172	-0.000				
	TO BUS 3			0.114	0.000				
	TO BUS 1			-0.288	-0.117				
BUS	3	1.058	-1.75	0.527	0.022	0.450	0.150	0.000	-0.000
	TO BUS 4			0.347	-0.022				
	TO BUS 2			-0.113	-0.043				
	TO BUS 1			-0.157	-0.063				
BUS	4	1.055	-2.29	-0.000	0.000	0.400	0.050	-0.000	0.000
	TO BUS 5			0.116	-0.013				
	TO BUS 3			-0.346	0.003				
	TO BUS 2			-0.171	-0.043				
BUS	5	1.043	-3.68	0.000	0.000	0.600	0.100	-0.000	0.000
	TO BUS 4			-0.115	-0.039				
	TO BUS 2			-0.485	-0.061				

TOTAL SYSTEM LOSS = 0.016

SYSTEM VOLTAGE FLOATING FACTOR = 1.0169

This example showed that with the given cost data and the transmission loss specified at 1% of the given total

Poor Copy

demand, the optimum generation plan is generation bus 1 producing 0.448 P.U., bus 2 0.692 and bus 3 0.527. Furthermore the generations units will no longer operate at their nominal voltages but at 1.0169 times their nominal values.

6. CONCLUSION

The load flow methods in general use today require a generation bus in the system to be singled out as the slack bus. This is at variance with the physical system in which no one bus has this special characteristic. However the slack bus is essential to the conventional load flow calculations because of mathematical restrictions. Though the concept of a slack bus is satisfactory in ordinary load flow studies, when coupled with other power system anslyses, it may raise difficulties. For example, in economic dispatch, a slack bus load flow formulation will entail inelegant evaluations of the incremental transmission loss coefficients $\frac{\partial P_L}{\partial P_i}$.

This thesis proposes two new load flow methods that do not require a slack bus. In the Participation Factors Load Flow, all the generation buses real injections are allowed to change in proportion to their participation factors, to adjust the total generation to match the total demand plus transmission loss. With the Floating System Voltage Load Flow, on the other hand, all the real generations are specified, however the voltage magnitudes at the generation buses are allowed to vary from their nominal values so that the resulting transmission loss are as specified.

These two new load flow methods were coded into computer programs and tested with results obtained from the conventional N-R load method. In preparing the computer programs, sparsity of the matrices involved were exploited to conserve memory storage and improve execution time. It is to be noted that the conventional N - R load flow programs can be modified without much difficulty to solve load flow problems formulated under the new formulations. Thus it is hoped that others who wish to use these new load flow formulations will not be deterred from doing so by the the prospect of having to write their programs from scratch.

This work also examines the physical basis and possible applications of the two load flow methods developed. With the Participation Factors Load Flow, given the data on the droop characteristics, it is possible to determine how and by how much each individual generation unit will take up the imbalance should total generation not match the load.

The Floating System Voltage Load Flow as the name indicates, allows the voltage level of the system to vary around the nominal value. The voltages will not deviate appreciably from their nominal values if the total generation had not been seriously misjudged in matching generation to demand plus transmission loss. In this method all the real generations are independent variables and this enables a simple algorithm to be used in economic dispatch.

In addition to the two methods proposed, other no slack bus flow load methods are also possible. The load flow problem can be considered as a set of consistent algebraic equations relating the known operating conditions to the unknown nodal voltages. With this generalized perspective, as long as one is wary of overspecifying the operating conditions on the system, one can set up a load flow method which takes on as independent variables the operating conditions most relevant to the system under study or most applicable to the analysis to be undertaken.

REFERENCES

1. Gliman, A. F. and Stagg, G. W., "Automatic Calculation of Load Flows", AIEE Trans., Vol. 76, Oct 1957, pp. 817 - 828.
2. Stott, B., "Review of Load Flow Calculation Methods", Proc. IEEE, Vol. 62, No. 7, July 1977, pp. 916 - 929.
3. Yamane, K., "New Methods for Load Flow Calculation Without any Swing Bus", M. Eng. thesis, Massachusetts Institute of Technology, 1971.
4. Gomez, D. A. G., "Average System Frequency Approach to Slow Speed Power System Dynamics", M. Eng. thesis, Massachusetts Institute of Technology, 1972.
5. Iwamoto, S. and Tamura, Y., "A Fast Load Flow Method Retaining Nonlinearity", IEEE Trans., Vol. PAS-97, No. 5, Sept/Oct 1978, pp. 1586 - 1599.
6. Duran, H., Discussion on Reference [3].
7. Galiana, F. D., "Power-Voltage Limitations Imposed by the Network Structure of a Power System", PICA Proc., June 1975, pp. 356 - 363.
8. Strang, G., "Linear Algebra and its Applications", Academic Press, 1976, Chapter 1.
9. Tinney, W. F. and Walker, J. W., "Direct Solutions of

Sparse Network Equations by Optimally Ordered Triangular Factorization", Proc. IEEE, Vol. 55, No. 11, Nov 1967, pp. 1801 - 1809.

10. Eisenstat, S. C., Schultz, M. H. and Sherman, A. H., "Considerations in the Design of Software for Sparse Gaussian Elimination", Proceedings of the Symposium on Sparse Matrix Computations at Argonne National Laboratory Sept 1975.

11. Ogbuobiri, E. C., "Dynamic Storage and Retrieval in Sparsity Programming", IEEE Trans., Vol. PAS-89, No. 1, Jan 1970, pp. 150 -155.

12. Tewarson, R. P., "Sparse Matrices", Academic Press, 1973, Chapter 1.

13. Reference [8] Chapter 2.

14. Reference [8] Chapter 4.

15. Vojdani, A. F., "Analytic Approach to Economic Dispatch", M. Eng. thesis, McGill University, 1979.

16. Elgerd, O. I., "Electric Energy Systems Theory: An Introduction", McGrall-Hill, 1971, Chapter 9.

17. Gungor, R. B., Tsang, N. F. and Webb, B., "A Technique for Optimising Real and Reactive Power Schedules", IEEE Trans., Vol. PAS-90, 1971, pp. 1781 - 1790.

18. Stagg, G. W. and El-Abiad, A. H., "Computer Methods in

Power System Analysis", McGraw-Hill, 1968, Chapter 8.

APPENDIX A

Floating System Voltage Load Flow in Polar Coordinates

The floating system voltage load flow can be formulated in polar co-ordinates. At the PV buses the voltage magnitudes are permitted to vary by a factor ρ of their nominal value $|V|_i$. The specified quantities are the real powers $\underline{P}$ at all buses, the nominal voltages $\underline{V}_G$ at the PV buses and the reactive powers $\underline{Q}$ at the load buses. The unknowns are the voltage angles $\underline{\theta}$ at all buses except the reference bus, the factor ρ and the voltage magnitudes $\underline{V}_L$ at the load buses. The equations describing the system are:-

For each PV bus

$$P_{i\,spec} = P_i\,(\rho\underline{V}_G, \underline{V}_L, \underline{\theta})$$

For each PQ bus

$$P_{i\,spec} = P_i\,(\rho\underline{V}_G, \underline{V}_L, \underline{\theta})$$

$$Q_{i\,spec} = Q_i\,(\rho\underline{V}_G, \underline{V}_L, \underline{\theta})$$

In an n bus system with m PV buses, there are in all (2n - m) equations. At the same time there are (2n - m) unknowns: ρ, (n - m) elements of $\underline{V}_L$ and the (n - 1) elements of $\underline{\theta}$. Hence the above constitutes a set of (2n - m) equations in (2n - m) unknowns.

APPENDIX B

Sparse Matrix Storage and Retrieval

The sparse matrix storage schemes used in this thesis are described in this Appendix. Two schemes are used. The first scheme stores the non-zero elements sequentially, that is, all the non-zero elements of each row/column of the sparse matrix are placed adjacent to each other in the array and there are pointers to locate the first of each row/column entry. Thus a typical example would be as follows:

	1	2	3	4	5	6	...
Non-zero elements	a_{11}	a_{13}	a_{14}	a_{22}	a_{27}	a_{31}	...
Column position	1	3	4	2	7	1	...

	1	2	3	...
Pointers	1	4	6	...

With this scheme, if a particular element a needs to be retrieved one has to search only that part of the array, occupied by row i entries which are located between positions given by POINTER(i) and POINTER(i+1).

Excellent the above scheme may be, it has its limitations. Central to creating the storage of the sparse matrix with the above scheme, is the assumption that a priori knowledge of the structure of the sparsity of the matrix is available. All the non-zero elements of one row must be stored before the next row elements can be stored. This is so because non-zero elements of a particular row are stored adjacent to one another. Once storage assignment begins for elements of one row, all elements of that row must be assigned the following consecutive area in the array before assignment of storage for the next row. This may prove to be of no restriction for most matrices e.g. all the elements of the matrix are available or are generated row by row. However not all matrices are generated or processed row by row. A simple example is this: subsequent operations may dictate that the matrix be stored by columns but the matrix is generated row by row. Another example is the generation of the bus admittance matrix from the line parameters. Here, the non-zero elements are created in a random manner depending on the nodes the line is joining.

The sparse matrix storage scheme to use is a rather complicated one. The previous scheme has the restriction that all elements of a particular row must be stored adjacent to one another in the array. This scheme does away with this restriction. Elements of the matrix can be stored into the array in any order. However there must be means to indicate where all the elements of any particular row are, in the array. One way to do so is to have a pointer pointing to the first element of the row and the first element having a pointer pointing to the second element and so on until the last element which has a pointer with a null value indicating there is no other element of the same row stored in the array. Such a data structure is known as a linked list ---- all the elements of one group are linked together by pointers, the elements not necessarily being adjacent to one another in their locations. An example may further clarify the structure of a linked list. Suppose the matrix to be stored is such:-

a_{11}	a_{12}		
	a_{22}	a_{23}	
		a_{33}	a_{34}
	a_{42}		a_{44}

The elements of the above matrix may be stored in a linked list in the following fashion.

Non-zero elements	a_{11}	a_{12}	a_{22}	a_{42}	a_{23}	a_{33}	a_{34}	a_{44}
Pointers	2	0	5	8	0	7	0	0
Column position	1	2	2	2	3	3	4	4

	1	2	3	4
Starting pointers	1	3	6	4

In the above example the starting pointers show that the first element of row 1 is at location 1 in the array, first element of row 2 is at location 3 in the array and so on. With the starting pointer giving the lead, the chain could be traced on. From the starting pointer, location 1 of the non-zero elements array is the first entry of row one. Location 1 of the pointer array has a value of 2 i.e. the second non-zero entry of row 1 at location 2 of the non-zero elements array. Tracing further, location 2 in the pointer array has a value of 0, the null value; this value indicates the termination of the chain. Similar traces could be made for the entries of other rows.

With a clearer concept of a linked list we shall elaborate on how the storage scheme is implemented. The above example just illustrates how elements are accessed, they did not show how the linked list was created in the first place. The linked list in this work was implemented as n stacks (where n is the number of rows in the matrix). A stack is a last-in first-out data structure; the last data entry stored into the stack is the first element to be retrieved. The first elements of each row to be retrieved is pointed to by the corresponding pointers stored in the starting-pointers array. Hence, because of the stack structure implementation, the starting-pointers contain the locations of the last-stored elements of the matrix.

In the very beginning of the linked list creation, the staring-pointers are initiated to the null value. This is so because there are no last-stored elements at the very beginning. Next, the non-zero elements are filled into the array. As each non-zero element is filled into the non-zero elements array, the value in the appropriate starting-pointer array is copied into the corresponding location of the pointer array. This step links the newly stored element with the last entry from the same row of the matrix. Next, the starting pointer is updated to indicate the last entry is this presently stored element. A short section of FORTRAN code is included to illustrate concisely the operations involved.

```
      DO 1 I = 1,NROWS
1     LSTART(I) = 0
      I = 0
2     READ (5,*,END=7) ELEMENT, IROW, ICOLMN
      I = I + 1
      ARRAY(I) = ELEMENT
      LARRAY(I) = ICOLMN
      LINKPT(I) = LSTART(IROW)
      LSTART(IROW) = I
      GO TO 2
7     CONTINUE
```

APPENDIX C

Computer Program Listings

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                                                                    CC
CC                                                                    CC
CC                                                                    CC
CC                                                                    CC
CC                                                                    CC
CC    LOAD FLOW PROGRAM USING THE NEWTON RAPHSON METHOD                CC
CC                                                                    CC
CC                                                                    CC
CC                                                                    CC
CC                                                                    CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C     THIS PROGRAM IS DESIGNED TO ACCOMODATE UP TO
C
C         120 BUSES
C     AND 200 LINES
C
C
C     THE LINE DATA IS ENTERED FIRST, IN F10.5 FORMAT IN THE SEQUENCE
C     NODE NUMBER, NODE NUMBER, LINE RESISTANCE, REACTANCE AND ONE
C     HALF LINE CHARGING ADMITTANCE (ALL IN PER UNIT)
C
C     THE LAST CARD OF LINE DATA IS SEPARATED FROM THE FOLLOWING DECK
C     OF BUS DATA CARDS BY A BLANK CARD
C
C     NEXT, THE BUS DATA IS ENTERED IN F5.2 FORMAT IN THE SEQUENCE
C     BUS NUMBER, BUS TYPE, VOLTAGE MAGNITUDE, REAL POWER GENERATION,
C     REACTIVE POWER GENERATION, REAL POWER DEMAND, REACTIVE DEMAND,
C     INITIAL ESTIMATE OF BUS VOLTAGE MAGNITUDE AND ANGLE
C
C     THE LAST CARD OF BUS DATA MUST BE FOLLOWED BY A BLANK CARD
C
C     THE BUSES MUST BE NUMBERED CONSECUTIVELY STARTING FROM 1,
C     HOWEVER THERE IS NO RESTRICTION ON GROUPING OF THE TYPES OF BUSES
C     E.G. THE SLACK BUS CAN BE NUMBER THE FIRST BUS, THE LAST BUS OR
C     ANY NUMBER IN BETWEEN, THE SAME APPLIES TO PQ AND PV BUSES
C
C     THE BUS TYPES ARE CODED AS FOLLOWS;
C        1.0    PQ BUS
C        2.0    PV BUS
C        3.0    SLACK BUS WHICH IS ALSO THE REFERENCE BUS.
C
C
C
C
C
```

```
C     THE FOLLOWING ARE SOME PARAMETERS THAT CONTROL THE EXECUTION
C     OF THE PROGRAM        THESE PARAMETERS SHOULD BE READ IN THE
C     FOLLOWING SEQUENCE IN 5F10.5 FORMAT BEFORE OTHER DATA IS READ
C
C
C     IREAD  --- INPUT DEVICE NUMBER FOR LINE AND BUS DATA
C     IWRITE --- OUTPUT DEVICE NUMBER FOR RESULTS
C     CRITER --- ACCURACY TO WHICH THE MAXIMUM MISMATCH MUST SATISFY
C     JUPDAT --- HOW FREQUENTLY THE JACOBIAN MATRIX IS UPDATED
C           0  INDICATES THE INITIAL JACOBIAN MATRIX IS USED THROUGHOUT
C           1  INDICATES THE JACOBIAN MATRIX IS UPDATED EVERY ITERATION
C           2  INIDCATES JACOBIAN MATRIX UPDATED EVERY TWO ITERATIONS
C           N  INDICATES JACOBIAN MATRIX UPDATED EVERY  N  ITERATIONS
C     LOOP   --- MAXIMUM NUMBEF OF ITERATIONS ALLOWED
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      REAL*4 PARM(5)
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),NREP,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      COMMON /LUSOLV/ DELTAX(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,JUPDAT,LOOP
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      READ(5,1)PARM
      IREAD=PARM(1)
      IWRITE=PARM(2)
      CRITER=PARM(3)
      JUPDAT=PARM(4)
      LOOP=PARM(5)
      WRITE(IWRITE,5)
      CALL INTIAL
      CALL DINPUT
      CALL NEWTON(OK)
      CALL RESULT(OK)
      STOP
    1 FORMAT(5F10.5)
    5 FORMAT('1',T25,'CONVENTIONAL  LOAD  FLOW  PROGRAM  USING  NEWTON',
     +'-RAPHSON  ALGORITHM'// T28,'WITH L-U DECOMPOSITION OF THE JACOBIAN
     +N AND SPARSITY PROGRAMMING')
      END
```

```
      SUBROUTINE INTIAL
C
C
C     THIS SUBROUTINE INITIALISES VARIABLES
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      DO 1 I=1,120
      DIAYMR(I)=0.0
      DIAYMI(I)=0.0
      LKSTYM(I)=0
    1 KONECT(I)=0
      RETURN
      END
```

```
      SUBROUTINE DINPUT
C
C
C
C     THIS SUBROUTINE READS IN THE LINE DATA AND BUS DATA
C     THEN DOES A RE-NUMBERING OF THE BUSES ACCORDING TO THIS:
C
C     PV BUSES ARE GIVEN THE FIRST NUMBERS IN THE ORDER THEY ARE ENTERED
C     THEN PQ BUSES ARE NUMBERED IN ASCENDING ORDER OF THE NUMBER OF
C     LINES JOINING IT
C     THE SLACK BUS IS NUMBERED LAST
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      INTEGER*4 IREAD,IWRITE
      REAL*4 BUFLIN(5),BUFNOD(9),NODE1,NODE2,RPU,XLPU,YCPU,NODE,TYPE,
     +VM,PG,QG,PD,QD,VMG,ANGLE
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      COMMON /LUSOLV/ DELTAX(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,JUPDAT,LOOP
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      COMMON /CONTIG/ NODE1,NODE2,RPU,XLPU,YCPU
      COMMON /CONTIN/ NODE,TYPE,VM,PG,QG,PD,QD,VMG,ANGLE
      COMPLEX*16 REACTN,IMAG/(0.0D0,1.0D0)/
      EQUIVALENCE (BUFLIN,NODE1),(BUFNOD,NODE)
C
C     READING IN THE LINE DATA AND CALCULATING THE Y-MATRIX
C     ONLY NON-ZERO ELEMENTS OF THE Y-MATRIX ARE STORED
C
      LINE=0
      LIST=0
      WRITE(IWRITE,1111)
      WRITE(IWRITE,50)
   10 CONTINUE
         READ(IREAD,51)BUFLIN
         IF(NODE1.EQ.0.0)GO TO 20
         WRITE(IWRITE,52)BUFLIN
         LINE=LINE+1
         LIST=LIST+1
         N2=NODE2
         N1=NODE1
```

```
      KONECT(N1)=KONECT(N1)+1
      KONECT(N2)=KONECT(N2)+1
      REACTN=1D0/(RPU+XLPU*IMAG)
      SUSCEP=-REACTN*IMAG
      DIAYMR(N1)=DIAYMR(N1)+REACTN
      DIAYMR(N2)=DIAYMR(N2)+REACTN
      DIAYMI(N1)=DIAYMI(N1)+SUSCEP+YCPU
      DIAYMI(N2)=DIAYMI(N2)+SUSCEP+YCPU
      DATAYR(LINE)=-REACTN
      DATAYI(LINE)=-SUSCEP
      DATALN(LINE)=YCPU
      JCOLYM(LIST)=N2
      LINKYM(LIST)=LKSTYM(N1)
      LKSTYM(N1)=LIST
      LIST=LIST+1
      JCOLYM(LIST)=N1
      LINKYM(LIST)=LKSTYM(N2)
      LKSTYM(N2)=LIST
      GO TO 10
20    CONTINUE
      WRITE(IWRITE,55) LINE
C
C     READING IN THE BUS DATA
C
      WRITE(IWRITE,1111)
      NTOTAL=0
      NOGEN=0
      DTORAD=3.1415926D0/180.0D0
      WRITE(IWRITE,60)
30    CONTINUE
      READ(IREAD,61)BUFNOD
      IF(NODE.EQ.0.0) GO TO 40
      WRITE(IWRITE,62)BUFNOD
      NTOTAL=NTOTAL+1
      N=NODE
      NODBUS(N)=TYPE
      VREAL(N)=VMG*DCOS(ANGLE*DTORAD)
      VIMAG(N)=VMG*DSIN(ANGLE*DTORAD)
      REALG(N)=PG
      REALD(N)=PD
      REACTG(N)=QG
      REACTD(N)=QD
      DELTAG(N)=PG-PD
      DELTAQ(N)=QG-QD
      IF(TYPE.EQ.1.0) GO TO 30
        VMAGSQ(N)=VM**2
        KONECT(N)=0
        NOGEN=NOGEN+1
```

```
          IF(TYPE.EQ.3.0)NREF=N
          GO TO 30
40     CONTINUE
       NLESS1=NTOTAL-1
       NLESX2=NLESS1*2
       NTOTX2=NTOTAL*2
       NOGEN=NOGEN-1
       KONECT(NREF)=100
C
C      BUS RE-NUMBERING
C
       DO 45 I=1,NTOTAL
45     NORDER(I)=I
65     INTERC=0
         DO 70 I=2,NTOTAL
         IF(KONECT(I-1).LE.KONECT(I)) GO TO 70
           L=KONECT(I)
           KONECT(I)=KONECT(I-1)
           KONECT(I-1)=L
           L=NORDER(I)
           NORDER(I)=NORDER(I-1)
           NORDER(I-1)=L
           INTERC=1
70       CONTINUE
         IF(INTERC.NE.0)GO TO 65
       DO 80 I=1,NTOTAL
80     LORDER(NORDER(I))=I
       RETURN
50     FORMAT(' LINE DATA'/1X,9('=')///T8,'BUS NO. JOINS BUS NO.',T36,
      +'R  P.U.',T50,'XL  P.U.',T64,'YSH  P.U.'/'+',T8,7('_'),T22,7('_'),
      +T35,'___',T49,'____',T63,'_____'///)
51     FORMAT(8F10.5)
52     FORMAT(8X,F4.0,10X,F4.0,1X,3F14.4/)
55     FORMAT(//// T22,'THERE ARE ',I4,' LINES IN THE SYSTEM')
60     FORMAT(' BUS DATA'/1X,8('=')///T24,'VOLTAGE',T37,'GENERATION',T57,
      +'LOAD',T71,'STARTING VOLTAGE'//T8,'NUMBER',T16,'TYPE',T23,
      +'MAGNITUDE',T35,'REAL',T41,'REACTIVE',T54,'REAL',T60,'REACTIVE',
      +T71,'MAGNITUDE',T82,'ANGLE'/'+',T8,6('_'),T16,'____',T23,9('_'),
      +T35,'____',T41,8('_'),T54,'____',T60,8('_'),T71,9('_'),T82,
      +5('_')///)
61     FORMAT(16F5.2)
62     FORMAT(8X,F4.0,T17,F2.0,T25,F4.2,T34,F6.3,T42,F6.3,T53,F6.3,
      +T61,F6.3,T71,F6.2,T80,F6.2/)
1111   FORMAT(//////////)
       END
```

```
      SUBROUTINE NEWTON(OK)
C
C
C
C     THIS SUBROUTINE PERFORMS THE NEWTON RAPHSON ITERATIONS
C
C
C     IT RETURNS A LOGICAL*1 VARIABLE OF VALUE .TRUE. WHEN THE ROUTINE
C     WAS SUCCESSFULLY COMPLETED
C     OTHERWISE THE RETURNED VARIABLE IS .FALSE.
C     UNSUCCESSFUL COMPLETION IS WHEN THE REQUIRED ACCURACY IS NOT
C     ATTAINED IN THE MAXIMUM ALLOWED NUMBER OF ITERATIONS
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),NODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      COMMON /LUSOLV/ DELTAX(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,JUPDAT,LOOP
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      KOUNT=0
100   CONTINUE
C
C        ** CALCULATING CURRENT INJECTIONS
         DO 200 M=1,NTOTAL
            I=NORDER(M)
            CREAL(M)=DIAYMR(I)*VREAL(I)-DIAYMI(I)*VIMAG(I)
            CIMAG(M)=DIAYMI(I)*VREAL(I)+DIAYMR(I)*VIMAG(I)
            L=LKSTYM(I)
150         CONTINUE
               KDATA=(L+1)/2
               J=JCOLYM(L)
               CREAL(M)=CREAL(M)+DATAYR(KDATA)*VREAL(J)
     +                          -DATAYI(KDATA)*VIMAG(J)
               CIMAG(M)=CIMAG(M)+DATAYR(KDATA)*VIMAG(J)+
     +                          DATAYI(KDATA)*VREAL(J)
               L=LINKYM(L)
               IF(L.NE.0)GO TO 150
200      CONTINUE
```

```
C
C       ** EVALUATING THE MISMATCHES
        IF (NOGEN.EQ.0) GO TO 410
        DO 400 I=1,NOGEN
           J=NORDER(I)
           ERRORZ(I)=-VMAGSQ(J)+VREAL(J)**2+VIMAG(J)**2
           ERRORZ(I+NLESS1)=-DELTAG(J)+VREAL(J)*CREAL(I)
     +                          +VIMAG(J)*CIMAG(I)
400     CONTINUE
410     CONTINUE
        IF(NOGEN.EQ.NLESS1)GO TO 510
        M=NOGEN+1
        DO 500 I=M,NLESS1
           J=NORDER(I)
           ERRORZ(I)=-DELTAQ(J)-(VREAL(J)*CIMAG(I)-VIMAG(J)*CREAL(I))
           ERRORZ(I+NLESS1)=-DELTAG(J)+(VREAL(J)*CREAL(I)+
     +                     VIMAG(J)*CIMAG(I))
500     CONTINUE
510     CONTINUE
C
C       ** CHECKING AGAINST CONVERGENCE CRITERIA
        DO 600 I=1,NLESX2
           IF(DABS(ERRORZ(I)).GT.CRITER)GO TO 700
600     CONTINUE
C
C       ** ALL MISMATCHES ARE LESS THAN THE CRITERIA GIVEN
        WRITE(IWRITE,550)CRITER,KOUNT
        OK=.TRUE.
        RETURN
C
C       ** MORE ITERATIONS REQUIRED
C
700     CONTINUE
        IF(KOUNT.LT.LOOP)GO TO 710
        WRITE(IWRITE,560)CRITER,LOOP
        OK=.FALSE.
        RETURN
C
C       ** DETERMINE WHETHER TO UPDATE JACOBIAN MATRIX
710     CONTINUE
        IF(KOUNT.EQ.0)GO TO 730
        IF(JUPDAT.EQ.0)GO TO 750
        IF(KOUNT/JUPDAT.EQ.(KOUNT-1)/JUPDAT)GO TO 750
730     CONTINUE
        CALL JACOB
750     CONTINUE
        CALL BACKSB(NLESX2)
        DO 800 I=1,NLESS1
```

```
          J=NORDER(I)
          VREAL(J)=VREAL(J)+DELTAX(I)
          VIMAG(J)=VIMAG(J)+DELTAX(I+NLESS1)
800     CONTINUE
        KOUNT=KOUNT+1
      GO TO 100
550   FORMAT('1',T11,'CONVERGES TO WITHIN ',F8.5,' FOR THE MAXIMUM MISMATCH
     +TCH IN ',I3,' ITERATIONS' /////)
560   FORMAT('1',T11,'FAILS TO CONVERGE TO WITHIN ',F8.5,' FOR THE MAXIMUM
     +UM MISMATCH IN ',I3,' ITERATIONS' /////)
      END
```

```
      SUBROUTINE JACOB
C
C
C
C     THIS SUBROUTINE EVALUATES THE JACOBIAN MATRIX
C
C
C     THE JACOBIAN MATRIX IS NOT STORED BUT AS SOON AS ONE ROW IS
C     CALCULATED IT IS DECOMPOSED INTO THE CORRESPONDING ROWS OF THE
C     LOWER AND UPPER TRIANGULAR MATRICES
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      COMMON /LUSOLV/ DATAJB(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      DO 170 J=1,NLESS1
      I=NORDER(J)
      DO 179 J0=1,NTOTX2
         DATAJB(J0)=0.0
179      JCOLJB(J0)=0
      J0=1
      IF(MODBUS(I).NE.2) GO TO 175
         DATAJB(J)=-2.0*VREAL(I)
         JCOLJB(J0)=J+NLESS1
         DATAJB(J+NLESS1)=-2.0*VIMAG(I)
      GO TO 178
175      DATAJB(J)=-VIMAG(I)*DIAYMR(I)+VREAL(I)*DIAYMI(I)+CIMAG(J)
         L=LKSTYM(I)
171      CONTINUE
            M=JCOLYM(L)
            IF(M.EQ.NREF)GO TO 177
            M=LORDER(M)
            JCOLJB(J0)=M
            J0=J0+1
            KD=(L+1)/2
            DATAJB(M)=-VIMAG(I)*DATAYR(KD)+VREAL(I)*DATAYI(KD)
            JCOLJB(J0)=M+NLESS1
            J0=J0+1
            DATAJB(M+NLESS1)=VIMAG(I)*DATAYI(KD)+VREAL(I)*DATAYR(KD)
```

```
177         L=LINKYM(L)
            IF(L.NE.0)GO TO 171
        JCOLJB(J0)=J+NLESS1
        DATAJB(J+NLESS1)=VIMAG(I)*DIAYMI(I)+VREAL(I)*DIAYMR(I)-CREAL(J))
178     CALL LUNSYM(J,NLESX2)
170     CONTINUE
        DO 160 J=NTOTAL,NLESX2
        I=NORDER(J-NLESS1)
        DO 163 J0=1,NTOTX2
            DATAJB(J0)=0.0
163         JCOLJB(J0)=0
        J0=1
        DATAJB(J)=-VIMAG(I)*DIAYMR(I)+VREAL(I)*DIAYMI(I)-CIMAG(J-NLESS1)
        L=LKSTYM(I)
161         CONTINUE
            M=JCOLYM(L)
            IF(M.EQ.NREF)GO TO 166
            M=LORDER(M)
            JCOLJB(J0)=M
            J0=J0+1
            KD=(L+1)/2
            DATAJB(M)=-VREAL(I)*DATAYR(KD)-VIMAG(I)*DATAYI(KD)
            JCOLJB(J0)=M+NLESS1
            J0=J0+1
            DATAJB(M+NLESS1)=VREAL(I)*DATAYI(KD)-VIMAG(I)*DATAYR(KD)
166         L=LINKYM(L)
        IF(L.NE.0)GO TO 161
        JCOLJB(J0)=J-NLESS1
        DATAJB(J-NLESS1)=-VREAL(I)*DIAYMR(I)-VIMAG(I)*DIAYMI(I)
     +                  -CREAL(J-NLESS1)
        CALL LUNSYM(J,NLESX2)
160     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE LUNSYM(I,N)
C
C
C
C     THIS SUBROUTINE TRANSFORMS A GIVEN ROW OF A MATRIX INTO THE
C     CORRESPONDING ROWS OF A LOWER TRIANGULAR AND AN UPPER TRIANGULAR
C     MATRIX.  ONLY NON-ZERO ELEMENTS ARE STORED.
C
C     THE INPUT VARIABLES I GIVES WHAT ROW OF THE MATRIX IT IS TO BE
C     DECOMPOSED WHILE N GIVES THE ORDER OF THE MATRIX
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      COMMON /LUSOLV/ DATAJB(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      IF(I.NE.1)GO TO 22
C
C     ** INITIALIZATION AND CALCULATICN OF FIRST ROW OF UPPER
C        TRIANGULAR MATRIX
      KUT=1
      KLT=1
      IRSTUT(1)=1
      IRSTLT(1)=1
      DO 1 M=1,N
1     LKSTUT(M)=0
      DIAGUT(1)=DATAJB(1)
      J0=1
      L=JCOLJB(J0)
10    IF(L.EQ.0)GO TO 20
         IF(DATAJB(L).EQ.0.0)GO TO 15
            DATAUT(KUT)=DATAJB(L)
            JROWUT(KUT)=1
            JCOLUT(KUT)=L
            LINKUT(KUT)=LKSTUT(L)
            LKSTUT(L)=KUT
            KUT=KUT+1
15       CONTINUE
         J0=J0+1
         L=JCOLJB(J0)
      GO TO 10
20    RETURN
C
C     **DECOMPOSITION OF ROWS OTHER THAN THE FIRST
22    CONTINUE
      J0=1
      IRSTUT(I)=KUT
```

```
      IRSTLT(I)=KLT
      IR=IRSTLT(I)
C
C     ** SEEKING COLUMN ONE ENTRIES OF JACOBIAN MATRIX
      L=JCOLJB(J0)
130   IF(L.EQ.0)GO TO 110
         IF(L.EQ.1)GO TO 120
         J0=J0+1
         L=JCOLJB(J0)
      GO TO 130
C
C     ** EVALUATING ELEMENT OF COLUMN ONE OF LOWER TRIANGULAR MATRIX
120   JCOLLT(KLT)=1
      DATALT(KLT)=DATAJB(L)/DIAGUT(1)
      KLT=KLT+1
C
C     ** IF THIS IS SECOND ROW  NO MORE LOWER TRIANGULAR MATRIX ENTRIES
110   IF(I.EQ.2)GO TO 140
      I1=I-1
      DO 200 J=2,I1
      J0=1
      DATALT(KLT)=0.0
C
C     ** SEEKING NON-ZERO ELEMENT IN CORRESPONDING POSITION
         IN JACOBIAN MATRIX
      L=JCOLJB(J0)
220   IF(L.EQ.0)GO TO 230
         IF(L.EQ.J)GO TO 210
         J0=J0+1
         L=JCOLJB(J0)
      GO TO 220
210   DATALT(KLT)=DATAJB(L)
230   K1=KLT-1
C
C     ** IF THERE ARE NO PREVIOUS ENTRIES IN THIS ROW OF THE LOWER TRIAN-
C        GULAR MATRIX NO FURTHER PROCESSING IS NECESSARY FOR THIS ELEMENT
      IF(IR.GT.K1)GO TO 240
C
C     ** SCANNING THROUGH LIST OF ELEMENTS IN COLUMN J OF UPPER
C        TRIANGULAR MATRIX TO MATCH THE CORRESPONDING ENTRY IN THE
         LOWER TRIANGULAR MATRIX
      L=LKSTUT(J)
      MM=KLT
      DO 250 M=IR,K1
         MM=MM-1
270      IF(L.EQ.0.OR.(JROWUT(L).LT.JCOLLT(MM)))GO TO 250
            IF(JROWUT(L).EQ.JCOLLT(MM))GO TO 260
            L=LINKUT(L)
```

```
         GO TO 270
 260     DATALT(KLT)=DATALT(KLT)-DATAUT(L)*DATALT(MM)
 250  CONTINUE
C
C     ** IF ELEMENT IS ZERO DO NOT STORE INTO LIST
 240  IF(DATALT(KLT).EQ.0.0) GO TO 200
         JCOLLT(KLT)=J
         DATALT(KLT)=DATALT(KLT)/DIAGUT(J)
         KLT=KLT+1
 200  CONTINUE
C
C     ** CALCULATING THE DIAGONAL ELEMENT OF UPPER TRIANGULAR MATRIX
 140  DIAGUT(I)=DATAJB(I)
      K1=KLT-1
      IF(IR.GT.K1) GO TO 340
         L=LKSTUT(I)
         MM=KLT
         DO 300 J=IR,K1
            MM=MM-1
 330        IF(L.EQ.0.OR. (JROWUT(L).LT.JCOLLT(MM)))GO TO 300
               IF(JROWUT(L).EQ.JCOLLT(MM))GO TO 320
               L=LINKUT(L)
            GO TO 330
 320        DIAGUT(I)=DIAGUT(I)-DATALT(MM)*DATAUT(L)
 300     CONTINUE
 340  CONTINUE
C
C     ** IF IT IS THE LAST ROW  THERE IS NO NON-DIAGONAL ELEMENT
         IN UPPER MATRIX
      IF(I.EQ.N) GO TO 100
C
C     ** EVALUATING ELEMENTS IN THE UPPER TRIANGULAR
         MATRIX EXCLUDING DIAGONAL
      I1=I+1
      DO 400 J=I1,N
      J0=1
      DATAUT(KUT)=0.0
      L=JCOLJB(J0)
 430  IF(L.EQ.0) GO TO 410
         IF(L.EQ.J)GO TO 420
            J0=J0+1
            L=JCOLJB(J0)
         GO TO 430
 420     DATAUT(KUT)=DATAJB(L)
 410  IF(IR.GT.K1)GO TO 440
      L=LKSTUT(J)
      MM=KLT
      DO 450 M=IR,K1
```

```
      MM=MM-1
480      IF(L.EQ.0.OR.(JROWUT(L).LT.JCOLLT(MM)))GO TO 450
            IF(JROWUT(L).EQ.JCOLLT(MM))GO TO 470
            L=LINKUT(L)
         GO TO 480
470      DATAUT(KUT)=DATAUT(KUT)-DATAIT(MM)*DATAUT(L)
450   CONTINUE
440   IF(DATAUT(KUT).EQ.0.0) GO TO 400
         JROWUT(KUT)=I
         LINKUT(KUT)=LKSTUT(J)
         LKSTUT(J)=KUT
         JCOLUT(KUT)=J
         KUT=KUT+1
400   CONTINUE
      RETURN
100   CONTINUE
      IRSTLT(N+1)=KLT
      IRSTUT(N+1)=KUT
      RETURN
      END
```

```
      SUBROUTINE BACKSB(N)
C
C
C
C     THIS SUBROUTINE DOES A FORWARD THEN A BACKWARD SUBSTITUTION WITH
C     THE GIVEN LOWER AND UPPER TRIANGULAR MATRICES RESPECTIVELY.
C
C
C
C     THE INPUT VARIABLE N IS THE NUMBER OF ROWS IN THE MATRIX
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      COMMON /LUSOLV/ DELTAX(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      IEND=IRSTLT(2)-1
      DO 600 I=1,N
600   DELTAX(I)=ERRORZ(I)
      DO 700 I=2,N
         ISTART=IEND+1
         IEND=IRSTLT(I+1)-1
         IF(IEND.LT.ISTART)GO TO 700
         DO 750 J=ISTART,IEND
            M=JCOLLT(J)
            DELTAX(I)=DELTAX(I)-DATALT(J)*DELTAX(M)
750      CONTINUE
700   CONTINUE
      ISTART=IRSTUT(N+1)
      DO 800 I=1,N
         IR=N-I+1
         IEND=ISTART-1
         ISTART=IRSTUT(IR)
         IF(IEND.LT.ISTART)GO TO 800
         DO 850 J=ISTART,IEND
            M=JCOLUT(J)
            DELTAX(IR)=DELTAX(IR)-DATAUT(J)*DELTAX(M)
850      CONTINUE
800   DELTAX(IR)=DELTAX(IR)/DIAGUT(IR)
      RETURN
      END
```

```
      SUBROUTINE RESULT(OK)
C
C
C
C     THIS SUBROUTINE CALCULATES THE POWER INJECTIONS, THE POWER
C     FLOWS AND THE TOTAL TRANSMISSION LOSSES OF A SYSTEM
C     GIVEN THE NODAL VOLTAGES
C
C
C     IF THE INPUT VARIABLE IS .FALSE. IT PRINTS A WARNING MESSAGE
C     THAT THE NODAL VOLTAGES ARE NOT UP TO THE SUFFICIENT ACCURACY
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      INTEGER*4 IREAD,IWRITE
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      COMMON /LUSOLV/ DELTAX(240),ERRCRZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,JUPDAT,LOOP
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      LOGICAL*1 OK
      IF(.NOT.OK)WRITE(IWRITE,500)
      WRITE(IWRITE,520)
      PLOSS=0.0
      RTODEG=180D0/3.1415926D0
      DO 1000 I=1,NTOTAL
         VMAG=DSQRT(VREAL(I)**2+VIMAG(I)**2)
         ANGLE=DATAN2(VIMAG(I),VREAL(I))*RTODEG
         J=LORDER(I)
         IF(MODBUS(I).EQ.1) GO TO 300
            REACTG(I)=VIMAG(I)*CREAL(J)-VREAL(I)*CIMAG(J)+REACTD(I)
            IF(MODBUS(I).NE.3)GO TO 300
               REALG(I)=REALD(I)+VREAL(NREF)*CREAL(NTOTAL)
               WRITE(IWRITE,550)I,VMAG,ANGLE,REALG(I),REACTG(I),
     +         REALD(I),REACTD(I)
               GO TO 350
300      CONTINUE
         WRITE(IWRITE,550)I,VMAG,ANGLE,REALG(I),REACTG(I),
     +   REALD(I),REACTD(I),ERRORZ(J),ERRORZ(J+NLESS1)
350      CONTINUE
         PLOSS=PLOSS+REALG(I)-REALD(I)
```

```
      L=LKSTYM(I)
400   CONTINUE
        M=JCOLYM(L)
        KDATA=(L+1)/2
        VOLTRL=VREAL(I)-VREAL(M)
        VOLTIM=VIMAG(I)-VIMAG(M)
        CURREL=-DATAYR(KDATA)*VOLTRL+DATAYI(KDATA)*VOLTIM
        CURIMG=-DATAYR(KDATA)*VOLTIM-DATAYI(KDATA)*VOLTRL
        SHUNT=-DATALN(KDATA)*VMAG**2
        REAFLO=VREAL(I)*CURREL+VIMAG(I)*CURIMG
        REACTV=VIMAG(I)*CURREL-VREAL(I)*CURIMG+SHUNT
        WRITE(IWRITE,555)M,REAFLO,REACTV
        L=LINKYM(L)
        IF(L.NE.0)GO TO 400
1000  CONTINUE
      WRITE(IWRITE,559)PLOSS
      RETURN
500   FORMAT(T14,'THE FOLLOWING RESULTS ARE CALCULATED BASED ON THE YET
     +TO CONVERGE VOLTAGES' /////)
520   FORMAT(/ T14,'V O L T A G E',T35,'GENERATION',T56,'DEMAND',
     +T73,'MISMATCH' // T11,'MAGNITUDE   ANGLE',T33,'REAL   REACTIVE',
     +T52,'REAL   REACTIVE',T71,'Q/|V|  POWER' / )
550   FORMAT(// ' BUS ',I3,F10.3,F8.2,3(3X,2F8.3)/)
555   FORMAT(7X,'TO BUS ',I3,T30,2F8.3)
559   FORMAT(/////T20,'TOTAL SYSTEM LOSS = ',F8.3 / '1')
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE PARTICIPATION FACTORS LOAD FLOW PROGRAM                      C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      REAL*4 PARM(5)
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),SHARE(120),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),DEMAND,TOTALG,SUMSHA,NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120)
      COMMON /LUSOLV/ DELTAX(240),ERRCRZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,LOOP,JUPDAT
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      READ(5,1)PARM
      IREAD=PARM(1)
      IWRITE=PARM(2)
      CRITER=PARM(3)
      JUPDAT=PARM(4)
      LOOP=PARM(5)
      WRITE(IWRITE,5)
      CALL INTIAL
      CALL DINPUT
      CALL FACTOR(OK)
      CALL RESULT(OK)
      STOP
1     FORMAT(5F10.5)
5     FORMAT('1',T35,'NO  SLACK  BUS  LOAD  FLOW  PROGRAM  USING  NEWTON
     +  RAPHSON  ALGORITHM'// T38,'WITH L-U DECOMPOSITION OF THE JACOBIAN
     +N AND SPARSITY PROGRAMMING')
      END
```

```
      SUBROUTINE FACTOR(OK)
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),SHARE(120),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),NODBUS(120),DEMAND,TOTALG,SUMSHA,NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120)
      COMMON /LUSOLV/ DELTAX(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,LOOP,JUPDAT
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      KOUNT=0
  100 CONTINUE
C        ** CALCULATING CURRENT INJECTIONS AT ALL NODES
         DO 200 M=1,NTOTAL
            I=NORDER(M)
            CREAL(M)=DIAYMR(I)*VREAL(I)-DIAYMI(I)*VIMAG(I)
            CIMAG(M)=DIAYMI(I)*VREAL(I)+DIAYMR(I)*VIMAG(I)
            L=LKSTYM(I)
  150       CONTINUE
               KDATA=(L+1)/2
               J=JCOLYM(L)
               CREAL(M)=CREAL(M)+DATAYR(KDATA)*VREAL(J)
     +                          -DATAYI(KDATA)*VIMAG(J)
               CIMAG(M)=CIMAG(M)+DATAYR(KDATA)*VIMAG(J)+
     +                          DATAYI(KDATA)*VREAL(J)
               L=LINKYM(L)
               IF(L.NE.0)GO TO 150
  200    CONTINUE
C
C        ** EVALUATING THE TOTAL GENERATION
         TOTALG=0.0
         IF (NOGEN.EQ.0)GO TO 310
         DO 300 I=1,NOGEN
            J=NORDER(I)
  300       TOTALG=TOTALG+VREAL(J)*CREAL(I)+VIMAG(J)*CIMAG(I)+REALD(J)
  310    TOTALG=TOTALG+VREAL(NREF)*CREAL(NTOTAL)+VIMAG(NREF)*CIMAG(NTOTAL)
     +   L)+REALD(NREF)
         TOTALG=TOTALG/SUMSHA
C
C        ** EVALUATING THE MISMATCHES
         IF (NOGEN.EQ.0)GO TO 410
         DO 400 I=1,NOGEN
```

```
          J=NORDER(I)
          ERRORZ(I)=-VMAGSQ(J)+VREAL(J)**2+VIMAG(J)**2
          ERRORZ(I+NLESS1)=-SHARE(J)*TOTALG+REALD(J)+VREAL(J)*CREAL(I)
     +                   +VIMAG(J)*CIMAG(I)
400    CONTINUE
410    CONTINUE
       IF(NOGEN.EQ.NLESS1)GO TO 510
       M=NOGEN+1
       DO 500 I=M,NLESS1
          J=NORDER(I)
          ERRORZ(I)=-DELTAQ(J)-VREAL(J)*CIMAG(I)+VIMAG(J)*CREAL(I)
          ERRORZ(I+NLESS1)=-DELTAG(J)+VREAL(J)*CREAL(I)+
     +                             VIMAG(J)*CIMAG(I)
500    CONTINUE
510    CONTINUE
C
C      ** CHECKING AGAINST CONVERGENCE CRITERIA
       DO 600 I=1,NLESX2
          IF(DABS(ERRORZ(I)).GT.CRITER)GO TO 700
600    CONTINUE
       WRITE(IWRITE,550)CRITER,KOUNT
       OK=.TRUE.
       RETURN
700    CONTINUE
       IF(KOUNT.LT.LOOP)GO TO 710
       WRITE(IWRITE,560)CRITER,LOOP
       OK=.FALSE.
       RETURN
710    CONTINUE
       IF(KOUNT.EQ.0)GO TO 730
       IF(JUPDAT.EQ.0)GO TO 750
       IF(KOUNT/JUPDAT.EQ.(KOUNT-1)/JUPDAT)GO TO 750
730    CONTINUE
       CALL JACOB
750    CONTINUE
       CALL BACKSB
       DO 800 I=1,NLESS1
          J=NORDER(I)
          VREAL(J)=VREAL(J)+DELTAX(I)
          VIMAG(J)=VIMAG(J)+DELTAX(I+NLESS1)
800    CONTINUE
       KOUNT=KOUNT+1
     GO TO 100
550  FORMAT('1',T11,'CONVERGES TO WITHIN ',F8.5,' FOR THE MAXIMUM MISMATCH
    +TCH IN ',I3,' ITERATIONS' /////)
560  FORMAT('1',T11,'FAILS TO CONVERGE TO WITHIN ',F8.5,' FOR THE MAXIMUM
    +UM MISMATCH IN ',I3,' ITERATIONS' /////)
     END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C                                                                      C
C     THE FLOATING SYSTEM VOLTAGE LOAD FLOW PROGRAM                    C
C                                                                      C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      REAL*4 PARM(5)
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),NODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120)
      COMMON /LUSOLV/ DELTAX(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,LOOP,JUPDAT
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      READ(5,1)PARM
      IREAD=PARM(1)
      IWRITE=PARM(2)
      CRITER=PARM(3)
      JUPDAT=PARM(4)
      LOOP=PARM(5)
      WRITE(IWRITE,5)
      CALL INTIAL
      CALL DINPUT
      CALL SYSVOL(OK)
      CALL RESULT(OK)
      STOP
1     FORMAT(5F10.5)
5     FORMAT('1',T35,'NO SLACK BUS LOAD FLOW PROGRAM USING NEWTON
     + RAPHSON ALGORITHM'// T38,'WITH L-U DECOMPOSITION OF THE JACOBIAN
     +N AND SPARSITY PROGRAMMING')
      END
```

```
      SUBROUTINE SYSVOL(OK)
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120)
      COMMON /LUSOLV/ DELTAX(240),ERRORZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,LOOP,JUPDAT
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      COMMON /FLOAT/ RHO,RHOSQ
      RHO=1.0
      RHOSQ=1.0
      NEQTNS=NLESX2+1
      KOUNT=0
100   CONTINUE
C
C        ** CALCULATING CURRENT INJECTIONS AT ALL NODES
         DO 200 M=1,NTOTAL
            I=NORDER(M)
            CREAL(M)=DIAYMR(I)*VREAL(I)-DIAYMI(I)*VIMAG(I)
            CIMAG(M)=DIAYMI(I)*VREAL(I)+DIAYMR(I)*VIMAG(I)
            L=LKSTYM(I)
150         CONTINUE
               KDATA=(L+1)/2
               J=JCOLYM(L)
               CREAL(M)=CREAL(M)+DATAYR(KDATA)*VREAL(J)
     +                          -DATAYI(KDATA)*VIMAG(J)
               CIMAG(M)=CIMAG(M)+DATAYR(KDATA)*VIMAG(J)+
     +                          DATAYI(KDATA)*VREAL(J)
               L=LINKYM(L)
               IF(L.NE.0)GO TO 150
200      CONTINUE
C
C        ** EVALUATING THE MISMATCHES
         IF (NOGEN.EQ.0)GO TO 410
         DO 400 I=1,NOGEN
            J=NORDER(I)
            ERRORZ(I)=-VMAGSQ(J)+VREAL(J)**2+VIMAG(J)**2
            ERRORZ(I+NLESS1)=-      DELTAG(J)     +RHOSQ*(VREAL(J)*CREAL(I)
     +                          +VIMAG(J)*CIMAG(I))
400      CONTINUE
```

```
410     CONTINUE
        IF(NOGEN.EQ.NLESS1)GO TO 510
        M=NOGEN+1
        DO 500 I=M,NLESS1
           J=NORDER(I)
           ERRORZ(I)=-DELTAQ(J)-(VREAL(J)*CIMAG(I)-VIMAG(J)*CREAL(I))*
     +     RHOSQ
           ERRORZ(I+NLESS1)=-DELTAG(J)+(VREAL(J)*CREAL(I)+
     +                       VIMAG(J)*CIMAG(I))*RHOSQ
500     CONTINUE
510     CONTINUE
        ERRORZ(NLESX2+1)=-DELTAG(NREF)+VREAL(NREF)*CREAL(NTOTAL)*RHOSQ
C
C       ** CHECKING AGAINST CONVERGENCE CRITERIA
        DO 600 I=1,NEQTNS
           IF(DABS(ERRORZ(I)).GT.CRITER)GO TO 700
600     CONTINUE
        WRITE(IWRITE,550)CRITER,KOUNT
        OK=.TRUE.
        RETURN
700     CONTINUE
        IF(KOUNT.LT.LOOP)GO TO 710
        WRITE(IWRITE,560)CRITER,LOOP
        OK=.FALSE.
        RETURN
710     CONTINUE
        IF(KOUNT.EQ.0)GO TO 730
        IF(JUPDAT.EQ.0)GO TO 750
        IF(KOUNT/JUPDAT.EQ.(KOUNT-1)/JUPDAT)GO TO 750
730     CONTINUE
        CALL JACOB
750     CONTINUE
        CALL BACKSB(NEQTNS)
        DO 800 I=1,NLESS1
           J=NORDER(I)
           VREAL(J)=VREAL(J)+DELTAX(I)
           VIMAG(J)=VIMAG(J)+DELTAX(I+NLESS1)
800     CONTINUE
        RHO=RHO+DELTAX(NLESX2+1)
        RHOSQ=RHO*RHO
        KOUNT=KOUNT+1
      GO TO 100
550   FORMAT('1',T11,'CONVERGES TO WITHIN ',F8.5,' FOR THE MAXIMUM MISMATCH
     +TCH IN ',I3,' ITERATIONS' /////)
560   FORMAT('1',T11,'FAILS TO CONVERGE TO WITHIN ',F8.5,' FOR THE MAXIMUM
     +UM MISMATCH IN ',I3,' ITERATIONS' /////)
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C                                                                       C
C     THE FLOATING SYSTEM VOLTAGE LOAD FLOW PROGRAM                     C
C                                                                       C
C               THE ITERATIVE APPROACH                                  C
C                                                                       C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THE VARIABLE ITERAT IS THE MAXIMUM NUMBER OF TIMES VOLTAGE
C     ADJUSTMENTS IS TO BE PERFORMED
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      REAL*4 PARM(6)
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      COMMON /LUSOLV/ DELTAX(240),ERRCRZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,LOOP,JUPDAT,ITERAT
      COMMON /SIZE/ NLESS1,NTOTAL,NLESX2,NTOTX2
      READ(5,1)PARM
      IREAD=PARM(1)
      IWRITE=PARM(2)
      CRITER=PARM(3)
      JUPDAT=PARM(4)
      LOOP=PARM(5)
      ITERAT=PARM(6)
      WRITE(IWRITE,5)
      CALL INTIAL
      CALL DINPUT
      CALL FLLOAT(OK)
      CALL RESULT(OK)
      STOP
1     FORMAT(6F10.5)
5     FORMAT('1',T35,'NO  SLACK  BUS  FLOATING  SYSTEM  VOLTAGE ',
     +' LOAD  FLOW ')
      END
```

```
      SUBROUTINE FLLOAT(OK)
C
C
C     THIS SUBROUTINE PERFORMS THE LOAD FLOW ITERATIVELY, ADJUSTING THE
C     VOLTAGE MAGNITUDES UNTIL THE SPECIFIED TRANSMISSION LOSS IS ATTAINED
C
C
      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*2 (I-N)
      COMPLEX*16 V1,V2,VDIFF,DCMPLX
      INTEGER*4 IREAD,IWRITE
      LOGICAL*1 OK
      COMMON /LOADFL/ VREAL(120),VIMAG(120),VMAGSQ(120),CREAL(120),
     +CIMAG(120),CMAGLN(200),REALG(120),REACTG(120),
     +REALD(120),REACTD(120),MODBUS(120),NREF,NOGEN
      COMMON /NETWOK/ DIAYMR(120),DIAYMI(120),DATAYR(200),DATAYI(200),
     +DATALN(200),LKSTYM(120),JCOLYM(400),LINKYM(400),KONECT(120),
     +LORDER(120),NORDER(120),LINE
      COMMON /LUSOLV/ DELTAX(240),ERRCRZ(240),DELTAG(120),DELTAQ(120),
     +DIAGUT(240),DATAUT(6000),DATALT(6000),LKSTUT(240),JROWUT(6000),
     +LINKUT(6000),IRSTUT(240),JCOLUT(6000),IRSTLT(240),JCOLLT(6000),
     +JCOLJB(240)
      COMMON /ENABLE/ CRITER,IREAD,IWRITE,LOOP,JUPDAT,ITERAT
      COMMON /SIZE/NLESS1,NTOTAL,NLESX2,NTOTX2
C
C     CALCULATE THE IMPLICITLY SPECIFIED TRANSMISSION LOSS
C
      PLOSS=0.0
      DO 1 I=1,NTOTAL
      PLOSS=PLOSS+DELTAG(I)
1     CONTINUE
      IF(PLOSS.LE.0.0) WRITE(IWRITE,50)
      IF(PLOSS.LE.0.0) STOP
      KOUNT=1
9     CONTINUE
C
C     EXECUTES THE LOAD FLOW ITERATIVELY
C
         CALL NEWTON(OK)
         IF(.NOT.OK) RETURN
C
C     CALCULATE THE TRANSMISSION LOSS
C
         TRLOSS=0.0
         DO 2 I=1,LINE
            K=JCOLYM(2*I)
            L=JCOLYM(2*I-1)
            V1=DCMPLX(VREAL(K),VIMAG(K))
            V2=DCMPLX(VREAL(L),VIMAG(L))
```

```
        VDIFF=V1-V2
        VMAGDF=CDABS(VDIFF)
        TRLOSS=TRLOSS+VMAGDF*VMAGDF*(-DATAYR(I))
2     CONTINUE
      DPLOSS=DABS(PLOSS-TRLOSS)
      WRITE(IWRITE,52)DPLOSS
      IF(DPLOSS.LE.CRITER)WRITE(IWRITE,53)KOUNT
      IF(DPLOSS.LE.CRITER)RETURN
C
C   ADJUST THE VOLTAGES AT THE PV BUSES
C
      VLEVEL=TRLOSS/PLOSS
      IF(NOGEN.EQ.0)GO TO 4
      DO 3 I=1,NOGEN
        J=NORDER(I)
        VMAGSQ(J)=VLEVEL*VMAGSQ(J)
3     CONTINUE
4     CONTINUE
      VREAL(NREF)=VREAL(NREF)*DSQRT(VLEVEL)
      KOUNT=KOUNT+1
   IF(KOUNT.LT.ITERAT)GO TO 9
   WRITE(IWRITE,55)ITERAT
   OK=.FALSE.
   RETURN
50 FORMAT('1',T20,'INSUFFICIENT REAL GENERATIONS ALLOCATED TO',
  +' MEET DEMAND PLUS TRANSMISSION LOSSES')
52 FORMAT(/ T20,'DIFFERENCE BETWEEN CALCULATED & SPECIFIED TRANSMISSION
  +ON LOSS = ', F9.6 )
53 FORMAT('1',T20,'CONVERGES IN ',I2,' OUTER ITERATIONS'///)
55 FORMAT('1',T7,'AFTER',I3,' SYSTEM VOLTAGE CORRECTIONS THE ',
  +'SPECIFIED TRANSMISSION LOSSES IS STILL UNATTAINED'///)
   END
```

This thesis is prepared and produced on
the McGill University System of Interactive Computing
using the Context Editor and SCRIPT facilities