c.1

FAST LOAD FLOW ALGORITHMS

by

Hossein Jalali-Kushki

B.Sc., Arya-Mehr University of Technology, Tehran, Iran, 1971

M.A.Sc., University of British Columbia, Vancouver, Canada, 1973

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

THE FACULTY OF GRADUATE STUDIES

in the Department

of

Electrical Engineering

We accept this thesis as conforming to

the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

May, 1977

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering

The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date September 1, 1977

ABSTRACT

New, fast and reliable algorithms for solving load-flow problems are presented in this thesis. Each of these algorithms iteratively solves a set of linear equations in terms of voltage magnitude squared and phase angles, and converges onto the final solution in a few iterations. Although the line losses of the system are used in deriving the equations of the basic line-loss load-flow algorithm, knowledge of their (approximate) values is not a prerequisite to using the algorithm. The basic line-loss load-flow algorithm is slightly modified to give an incremental-change line-loss algorithm which proves to be always preferable to the basic algorithm. By exploiting the weak interdependence between active power and voltage magnitude, and between reactive power and phase angle, two decoupled versions of the incremental-change line-loss algorithm were also developed.

All these algorithms have constant gradient characteristics, and their storage requirements are, at most, the same as those of the standard Newton-Raphson algorithm. If need be, the storage requirements can be reduced to those of the triangularized Y-matrix iterative algorithms.

Tests on various systems indicate fast and reliable convergence characteristics better than those of the Newton-Raphson algorithm and comparable to those obtained by Stott and Alsac with their decoupled Newton-Raphson load-flow algorithm.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF ILLUSTRATIONS

NOMENCLATURE

Voltage

V_i	complex voltage of bus i
U_i	magnitude of V_i
δ_i	phase angle of V_i
e_i	complex voltage of bus i in the $(S-U^2)$ system

Power

S_{ik}	complex power flowing from bus i towards bus k
S_i	total complex power injection at bus i
S^{ℓ}_{ik}	complex power loss in line (i,k)
$P_{ik}, P_i, P^{\ell}_{ik}$	real parts of S_{ik}, S_i, S^{ℓ}_{ik}, respectively
$Q_{ik}, Q_i, Q^{\ell}_{ik}$	imaginary parts of S_{ik}, S_i, S^{ℓ}_{ik}, respectively

Admittance

y_{ik}	value of admittance connecting buses i and k
y_i	total shunt admittance at bus i
Y	nodal admittance matrix
Y_{ij}	elements (i,j) of Y
G	real part of Y
B	imaginary part of Y
G_{ij}, B_{ij}	elements (i,j) of G and B, respectively
g_i	total shunt conductance at bus i
b_i	total shunt susceptance at bus i
x_{ik}	reactance of branch (i,k)

General

(i,k)	branch connecting buses i and k
*	complex conjugate
j	complex operator, $\sqrt{-1}$
Im	imaginary part of a complex number
Re	real part of a complex number
ε_{ik}	correction term corresponding to branch (i,k)
n	number of buses
n_{PV}	number of P-V buses
n_{PQ}	number of P-Q buses
$i \in$ P-Q or P-V	i is a P-Q (or P-V) bus
Δ	differential operator
superscript (ν)	indicates iteration number

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my supervisor, Dr. M.D. Wvong, for his continued encouragement, help and guidance throughout the research and writing of this thesis.

I would also like to thank Dr. Y.N. Yu, for his constant encouragement during the research work and for proof-reading the thesis and offering valuable suggestions.

I am also grateful to Dr. H.W. Dommel for proof-reading the thesis and also for providing me the excellent opportunity to visit the Bonneville Power Administration and have valuable discussions with the people there.

Thanks are also due to Miss Sannifer Louie for typing the thesis.

The financial support from the National Research Council of Canada is gratefully acknowledged.

1. INTRODUCTION

1.1 The Load-Flow Problem

"A power system continuously experiences changes in its operating state. These changes can be due to load demand variations, rescheduling of power generation, redistribution of reactive power generation, changing tap settings of transformers or phase shifters and disconnecting lines, transformers and generators for maintenance or as a consequence of system faults. The effect of these variations is investigated both during system planning and operation. One of the objectives is to ensure that the quality of service provided to the consumers is acceptable. Another is to confirm that these operations do not cause overloading of system elements. Yet another objective during system planning is to determine the range of tap settings of transformers and the capacities of the excitors required for proper steady-state and transient operation. The real and reactive power flow patterns are also studied as an important part of system security monitoring process .... A large number of load flow solutions will have to be obtained for a present day power system." [G16]

"Power flow studies are the backbone of the design of a power system. They are the means by which the future operation of the system is known ahead of time .... The continual expansion of the demand for electrical energy due to the growth of industries, commercial centers, and residential sections, require never-ending additions to existing power systems. The system engineer must decide what components must be added to the system many years before they are put into operation and he does this by means of power flow studies." [R5]

"The power flow program is indispensible for the analysis of ac power systems. Although improved solution methods and faster computers have extended its capabilities, this gain has been largely offset by the increased number and size of studies that must be made." [D7]

"An increasing use is being made of Power Flow algorithms in the electric utility industry, and recently the interest has been in applying a form of the algorithm to real time situations. The new uses of Power Flow imply that faster and more efficient algorithms are desired." [N7]

"The load-flow program is certainly the most frequently used computer program for power system applications. It consumes the largest computer time per year. This time has been growing as power system planners see the need for solving ever larger cases in succession of base case and many contingent solutions .... The great demand that load-flow applications place on computing resources accounts for the continued interest in the development of techniques that are faster, exhibit quicker convergence for a variety of system conditions, and make efficient use of core." [D8]

These statements of the "authorities" in the field - from American Electric Power Services Corporation, to IBM Scientific Center, to Bonneville Power Administration - should suffice to indicate the great importance of load-flow studies and the necessity of developing even faster and more efficient load-flow algorithms in order to deal with the huge interconnected power systems of today.

The problem is to specify a set of complex voltages for the system nodes (or buses) that satisfy certain requirements. Depending upon these requirements, the buses of the system can be divided into three main categories:

(a) The load or P-Q buses; whose active and reactive power injections are given and whose voltage magnitudes and phase angles have to be determined. A system may have any number of these buses.

(b) The voltage regulated, generator or P-V buses; whose voltage magnitudes and active power injections are given and whose phase angles have to be specified. A system may have any number of these buses.

(c) The reference, swing or slack buses; whose voltage magnitudes and phase angles are pre-determined and have no constraints on their power injections. A system can have any number of these buses but must at least have one.

The problem can be formulated in terms of node or mesh variables. The most commonly used formulation is the one using the nodal equations:

$$I = Y \cdot V \tag{1-1}$$

where I is the vector of injected nodal currents, Y is the nodal admittance matrix and V is the vector of nodal voltages. However, since neither the voltages nor the currents are known, the nodal equations alone will not be adequate. Additional equations are required to relate the known quantities, the active and reactive power injections, to the nodal voltages and currents. These are:

$$S_i = V_i \, I_i^* \qquad i = 1, \ldots, n \tag{1-2}$$

where S_i is the total complex power injection at node i and * is used to indicate the conjugate of a complex number. The nonlinearity of the load-flow problem arises from the fact that:

(1) equation (1-2) is nonlinear, and

(2) for P-V buses, the problem is further complicated because, in equation (1-2), only the real part of S_i and the magnitude of V_i are known.

Equations (1-1) and (1-2) represent the basic formulation of the load-flow problem. All the load-flow algorithms that use nodal analysis, use some combination of these equations to correct the voltages - in steps - from a set of initial estimates onto the final solution. The differences in the convergence characteristics and the computing speed of various algorithms are due to the fact that each algorithm uses (1-1) and (1-2) differently.

Normally the per unit quantities of the variables are used and certain practical assumptions are usually made in developing the load-flow algorithms. Among these are the facts that for any practical power system:

(1) The voltage magnitudes are approximately 1 P.U. and all the phase differences across various branches are small;

(2) The transmission lines are highly inductive;

(3) The interdependence between active powers and voltage magnitudes and between reactive powers and phase angles are weak and may be neglected; and

(4) The losses of the system constitute a small percentage of the total generated power.

The first assumption is made not only in developing load-flow algorithms, but also in choosing a reliable starting point. This is normally referred to as a "flat-start", and is obtained by setting all P-Q bus voltage magnitudes to 1 P.U. and all the phase angles to zero*.

Other practical considerations, such as the required speed and accuracy of the solution, play a role both in developing load-flow algorithms and in choosing a particular algorithm for a specific study.

1.2 Review of Available Load-Flow Algorithms

Before digital computers became available, the load-flow problem was solved on network analysers. The first totally automatic algorithm for solving the load-flow problem on a digital computer was presented by Ward and Hale in 1956 [Y4]. Since then, considerable attention has been paid to the development of faster, more efficient and more reliable load-flow algorithms and enormous progress has been made in this direction**. This section is intended to give a general account of the development of load-flow algorithms. It is by no means a complete and exhaustive survey of the literature.

The Ward-Hale algorithm [Y4], as it is normally referred to, is one of a group of load-flow algorithms classified as the "Y-matrix Iterative Methods" [Y1-Y4]. The reason for this classification is that all these algorithms are based on the iterative solution of the set of linear equations in (1-1). Their storage requirements are minimal and, for that reason, they were very suitable to the early generation of computers. Furthermore, the Y-matrix iterative methods are easy to

---

* Other starting points are also used. For example, in assessing the system outages, the base case solution may be preferable to the flat-start values.

**See "Bibliography" for a partial listing.

program and perform satisfactorily on many problems. The changes in the system configuration can be easily accounted for and the automatic controls can be easily included in these algorithms.

On the other hand, the Y-matrix iterative methods converge very slowly and sometimes do not converge at all. In practice, acceleration techniques are invariably used to speed up the convergence of these algorithms. However, even with the use of the "best" acceleration factors, for large systems, the total computation time of these algorithms is far greater than that of the newer methods. Nowadays, with the constant growth of the power systems and the increase in the number of load-flow problems that have to be solved for each system, and also due to the appearance of the modern and very powerful digital computers with storage capabilities far beyond those of the earlier computers, the Y-matrix iterative methods are becoming less and less attractive.

In the early 1960's another group of load-flow algorithms, classified as the "Z-matrix Methods" [Z1-Z4], were introduced. These algorithms, which are based on the direct solution of the linear set of equations in (1-1), have more reliable convergence characteristics than the Y-matrix iterative methods. On the other hand, their storage requirements and computation times are considerably more and grow enormously with system size. They are rarely competitive with the Y-matrix iterative methods and still less competitive with the newer load-flow algorithms. The Z-matrix methods did not become very popular.

In 1959, the Newton-Raphson method of solving the load-flow problem was shown to have powerful convergence characteristics [N13, N14]. Due to its large storage and computer time requirements, however, it was not competitive with the Y-matrix iterative methods. It was not until

1967, when, by using sparsity programming and ordered Gaussian elimination these problems were overcome [G23,N11], that the algorithm became practical and widely used.

Because of its quadratic convergence, the Newton-Raphson algorithm converges very rapidly to very accurate solutions. For practical accuracies, it always converges in a few iterations, irrespective of the system size. It is far superior to any of the Y-matrix iterative methods but it is also far more complicated to program. Its storage requirements, even with sparsity programming and ordered elimination, are much more than those of the Y-matrix iterative methods, but impose no serious problem on most present-day computers.

Soon after the appearance of the paper by Tinney and Hart in 1967 [N11], the Newton-Raphson algorithm became very popular and replaced the "classical" Ward-Hale algorithm. Subsequently, a great deal of work was done in the direction of developing the Newton-Raphson load-flow algorithm and improving upon certain aspects of it. The polar form of this algorithm became to be widely regarded as the standard method of solving load-flow problems. Even today, the Newton-Raphson method of solving load-flow problems is the algorithm which is most widely used by the industry. Also, this algorithm is almost invariably used as the basis of comparison for the newly developed load-flow algorithms.

In 1967, Bonaparte and Maslin [D1] presented a new load-flow algorithm which was based on exploiting the weak interdependence between real power flows and voltage magnitudes. In this algorithm - the DC load-flow as it is normally called - a set of linear equations relating the real power injections to voltage phase angles is directly solved at each iteration. The matrix of coefficients remains constant throughout

the process and, therefore, the computation time per iteration is greatly reduced. The algorithm was proposed as a means of obtaining fast and approximate real power flow solutions and as such, it became very popular and widely used.

The idea of exploiting the weak interdependence between the real powers and voltage magnitudes and between the reactive powers and voltage phase angles of any practical power system was used in several subsequent papers. In the proposed decoupled algorithms, the P-U and Q-δ couplings are altogether neglected and the load-flow equations are simplified. The idea of decoupling the load-flow equations did not become very popular until, in a recent paper by Stott and Alsac [D11], it was shown that a decoupled version of the Newton-Raphson algorithm has better characteristics than the original undecoupled algorithm. The Fast Decoupled (Newton-Raphson) Load Flow, as the authors called it, is becoming increasingly popular [D5,D8] and is expected to replace the Newton-Raphson algorithm.

Many other load-flow algorithms have been developed during the past two decades which do not belong to any of the above categories. In fact, with the incredible growth of power systems, the scope of the problem has become so wide and the number of publications related to the subject so numerous that it would be practically impossible for this survey to give a complete coverage. A more complete account of the development in this field can be found in several review papers [R1-R6] that have appeared on the subject.

1.3 The Newton-Raphson Algorithm(*) [N11]

Combining equations (1-1) and (1-2) we obtain:

$$S_i = V_i \sum_{k=1}^{n} Y_{ik}^* V_k^* \tag{1-4}$$

where Y_{ik} is the (i,k)th element of the nodal admittance matrix, Y. Using polar coordinates:

$$\begin{aligned} V_i &\triangleq U_i \, e^{j\delta_i} \\ Y_{ik} &\triangleq |Y_{ik}| \, e^{j\theta_{ik}} \end{aligned} \tag{1-5}$$

where j is the complex operator, $\sqrt{-1}$, equation (1-4) becomes:

$$S_i = P_i + jQ_i = \sum_{k=1}^{n} U_i U_k |Y_{ik}| \, e^{j(\delta_i - \delta_k - \theta_{ik})} \tag{1-6}$$

The Newton-Raphson technique is then used to solve the set of nonlinear equations in (1-6).

The elements of the Jacobian matrix are the partial derivatives of P_i and Q_i with respect to voltage magnitudes and phase angles:

$$\Delta P_i = \sum_{k=1}^{n} (H_{ik} \, \Delta\delta_k + N_{ik} \frac{\Delta U_k}{U_k}) \tag{1-7a}$$

and

$$\Delta Q_i = \sum_{k=1}^{n} (J_{ik} \, \Delta\delta_k + L_{ik} \frac{\Delta U_k}{U_k}) \tag{1-7b}$$

where:

$$\begin{aligned} H_{ik} &\triangleq \frac{\partial P_i}{\partial \delta_k} \quad ; \quad & N_{ik} &\triangleq \frac{\partial P_i}{\partial U_k} U_k \\ J_{ik} &\triangleq \frac{\partial Q_i}{\partial \delta_k} \quad ; \quad & L_{ik} &\triangleq \frac{\partial Q_i}{\partial U_k} U_k \end{aligned} \tag{1-8}$$

(*) Throughout this thesis, the phrase "the Newton-Raphson algorithm" refers to the application of this well-known algorithm for solving a set of nonlinear equations to the polar formulation of the load-flow problem.

and

$$\Delta P_i \triangleq P_i(\text{scheduled}) - P_i(\text{calculated})$$

$$\Delta Q_i \triangleq Q_i(\text{scheduled}) - Q_i(\text{calculated})$$

The partial derivatives defined in (1-8) are real functions of the admittance matrix and node voltages. Even though the problem is formulated in polar coordinates, these partial derivatives should be calculated by rectangular complex arithmetic. Assuming the following rectangular expressions for the admittances and voltages:

$$\begin{aligned} V_i &\triangleq e_i + jf_i \\ Y_{ik} &\triangleq G_{ik} + jB_{ik} \end{aligned} \tag{1-9}$$

and defining:

$$a_k + jb_k = (e_k + jf_k)(G_{ik} + jB_{ik}) \tag{1-10}$$

we have for $i \neq k$:

$$\begin{aligned} H_{ik} = L_{ik} &= a_k f_i - b_k e_i \\ N_{ik} = -J_{ik} &= a_k e_i + b_k f_i \end{aligned} \tag{1-11a}$$

and for $i = k$:

$$\begin{aligned} H_{ii} &= -Q_i - B_{ii} U_i^2 \\ L_{ii} &= Q_i - B_{ii} U_i^2 \\ N_{ii} &= P_i + G_{ii} U_i^2 \\ J_{ii} &= P_i - G_{ii} U_i^2 \end{aligned} \tag{1-11b}$$

For a system of n nodes (not including the slack buses) of which n_{PQ} are P-Q type, the number of unknowns is $n + n_{PQ}$. The number of equations is likewise $n + n_{PQ}$ because for every P-V or P-Q bus we can write (1-7a) and for every P-Q bus we can also write (1-7b). In this way, for every P-Q bus, there is a "double row" and a "double column" in the Jacobian matrix; for every P-V bus there is a single row and a single column; and there are no rows or columns corresponding to the slack buses. The element (i,k) of the Jacobian matrix is therefore a submatrix whose dimensions are 2-by-2, 2-by-1, 1-by-2, or 1-by-1, depending upon the types of buses i and k. The structure of the Jacobian matrix would then be the same as that of the nodal admittance matrix, but the former will not be symmetrical even if the latter is.

At every step of the iterative process, the Jacobian matrix is evaluated and the following set of linear equations is directly solved:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ J & L \end{bmatrix} \cdot \begin{bmatrix} \Delta\delta \\ \frac{\Delta U}{U} \end{bmatrix} \tag{1-12}$$

using Gaussian elimination; i.e., through the process of triangularization and back substitution. Although the original Jacobian matrix is highly sparse, because of the new elements that are created in the course of the triangularization process, its triangularized form may not be sparse any more. In order to prevent this, Tinney and Hart [N11] suggested several ordering schemes for the nodes of the system that tend to reduce the number of newly accumulated elements to a minimum. One of these schemes, which the authors used in their program and which is also used

in LFP*, calls for numbering the nodes "so that at each step of the elimination the next node to be eliminated is the one having the fewest connected branches. This method requires simulation of the elimination process to take into account the changes in the node-branch connections effected at each step".

Using the above scheme, the rows of the Jacobian matrix are ordered only once in the beginning of the process. The same order is then used at every iteration step. The augmented Jacobian matrix is formed and triangularized, row by row in that order, and the result is stored in compact form. The sign of integer row and column pointers indicate types of rows and columns and, therefore, number and order of the elements of the Jacobian submatrices. This storage scheme, which is explained in detail in [N11] was also used in programming the Newton-Raphson algorithm in LFP.

1.4 The Fast Decoupled (Newton-Raphson) Load-Flow [D11]

This algorithm was first presented in 1973 in a paper by Stott and Alsac [D11]. The polar form of the Newton-Raphson algorithm is taken as the starting point. This means starting with equation (1-12), repeated here for convenience:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ J & L \end{bmatrix} \cdot \begin{bmatrix} \Delta\delta \\ \frac{\Delta U}{U} \end{bmatrix} \qquad (1\text{-}12)$$

The algorithm takes advantage of the weak interdependence between real powers and voltage magnitudes and between the reactive powers and phase

---

* LFP is a collection of programs for solving the load-flow problem using various algorithms. The programs were written for the purpose of comparing the performance of various algorithms. See Appendix A.

angles (for practical power systems operating in steady state) to decouple P-δ and Q-U problems. "The first step" in this direction is to neglect the coupling submatrices [N] and [J] in (1-12), obtaining two separated sets of equations:

$$[\Delta P] = [H][\Delta\delta] \tag{1-13a}$$

$$[\Delta Q] = [L][\frac{\Delta U}{U}] \tag{1-13b}$$

where the elements of [H] and [L], for $i \neq k$, are

$$H_{ik} = L_{ik} = U_i\ U_k[G_{ik}\ \sin(\delta_i - \delta_k) - B_{ik}\ \cos(\delta_i - \delta_k)] \tag{1-14a}$$

and:

$$\begin{aligned} H_{ii} &= -\ B_{ii}\ U_i^2 - Q_i \\ L_{ii} &= -\ B_{ii}\ U_i^2 + Q_i \end{aligned} \tag{1-14b}$$

However, it was found that the decoupled equations of (1-13) represent an unstable algorithm unless other practical assumptions are made and an alternative equation is used instead of (1-13b) [D9]. The "best" stable algorithm was derived in [D11] using further physically justifiable simplifications as follows:

"In practical power systems the following assumptions are almost always valid:

$$\cos(\delta_i - \delta_k) \approx 1\ ;$$

$$G_{ik}\ \sin(\delta_i - \delta_k) << B_{ik}\ ;$$

$$Q_i << B_{ii}\ U_i^2$$

so that good approximations to (1-13a) and (1-13b) are:

$$[\Delta P] = [U \cdot B' \cdot U][\Delta\delta] \qquad (1\text{-}15a)$$

$$[\Delta Q] = [U \cdot B'' \cdot U][\frac{\Delta U}{U}] \qquad (1\text{-}15b)$$

where [B'] and [B"] are matrices whose elements are strictly elements of [-B] and [U·B'·U] represents a matrix whose element (i,k) has the expression $U_i \cdot B'_{ik} \cdot U_k$. The decoupling process and the final algorithmic forms are now completed by:

(a) omitting from [B'] the representation of those network elements that predominantly affect MVAR flows; i.e., shunt reactances and off-nominal in-phase transformer taps

(b) omitting from [B"] the angle shifting effects of phase shifters

(c) taking the left-hand U terms in (1-15a) and (1-15b) onto the left-hand sides of the equations, and then in (1-15a) removing the influence of MVAR flows on the calculation of [Δδ] by setting all the right-hand U terms to 1 P.U. Note that the U terms on the left-hand sides of (1-15a) and (1-15b) affect the behaviour of the defining functions and not the coupling

(d) neglecting series resistances in calculating the elements of [B'], which then becomes the DC approximation load-flow matrix. This is of minor importance, but is found experimentally to give slightly improved results."

With the above modifications the final fast decoupled load-flow equations become:

$$[\frac{\Delta P}{U}] = [B'][\Delta\delta] \qquad (1\text{-}16a)$$

$$[\frac{\Delta Q}{U}] = [B''][\Delta U] \qquad (1\text{-}16b)$$

In the above algorithm, both [B'] and [B"] are constant, real and sparse matrices that need to be triangularized only once at the beginning of the study. Equations (1-16a) and (1-16b) are solved separately and repeatedly until all the power flows are within the desired tolerance of their scheduled values. Starting with (1-16a), equations (1-16a) and (1-16b) are solved alternatively unless all the active powers or all the reactive powers are within the desired tolerance of their final values in which case the solution of the corresponding set of equations is skipped. The solution of either (1-16a) or (1-16b) is considered to be one half iteration. Experiments have shown that the convergence of this algorithm is very fast and reliable [D5,D8,D11] and that the algorithm is much preferred to the undecoupled Newton-Raphson algorithm.

In LFP, (1-16a) and (1-16b) are solved using ordered Gaussian elimination. The buses are numbered such that at each stage of the elimination process of [B'], the next row to be eliminated is the one having the fewest number of off-diagonal elements. The ordering is assumed to be the same for [B'] and [B"]. Both these matrices are formed and triangularized row by row and the results are separately stored in compact form. Negative pointers are used to indicate P-V type buses.

1.5 The Line-Loss Algorithms [09,014,D4]

Applied numerical methods are most efficient when they take advantage of the physical properties of the system being solved. For example, exploiting the weak (P-U) and (Q-δ) interdependencies and other physical properties of the practical power systems, resulted in the development of the Fast Decoupled (Newton-Raphson) Load-Flow, which is

shown to have much better characteristics than the undecoupled Newton-Raphson algorithm.

In any practical electrical power transmission network, the line losses of the system form a very small percentage of the system's total power generation. In the following chapters, this physical property of the system is exploited and a number of very fast and reliable load-flow algorithms are developed. The idea of using the line losses of the system in a load-flow algorithm is new and results in a set of linear equations in terms of voltage magnitudes squared and voltage phase angles. The solution of the load-flow problem is obtained by repeatedly solving this set of equations.

Although knowledge of the approximate values of line losses can be effectively used with the line-loss algorithms, such knowledge is not a pre-requisite to using the algorithms. Since losses are usually small, their values may be initially set to zero. Alternatively, a flat start may be used to start the algorithms.

Two slightly different formulations of the line-loss algorithm, as well as two decoupled versions of this algorithm, are presented in this thesis. In Chapter 2, the basic line-loss algorithm is derived. Chapter 3 contains a modified version of this algorithm which is faster and has better convergence characteristics. The decoupled versions of the algorithm are introduced in chapter 4. The performance of these algorithms are compared with those of the Newton-Raphson algorithm and the Fast Decoupled (Newton-Raphson) Load-Flow in respective chapters.

2. THE LINE-LOSS ALGORITHM

During the last two decades, remarkable progress has been made in developing better algorithms for the digital solution of the load-flow problem. On the other hand, the same period has witnessed a tremendous growth in the size of power systems and the emergence of huge interconnected electrical power transmission networks involving many utilities, which has resulted in a sizeable increase in the number and size of load-flow problems to be solved. This growth has been so enormous that it has largely offset the gains of the aforementioned development. Consequently, due to the ever-increasing system size and the growing complexity of the problems associated with it, even today, there is still a need for faster, more efficient and more reliable load-flow algorithms. This is why the constant-gradient fast-converging algorithms, such as the Fast Decoupled (Newton-Raphson) Load-Flow [D11] are becoming very popular. The line-loss load-flow algorithm presented here, is a constant-gradient algorithm with very fast and reliable convergence.

This algorithm was developed as the result of an attempt to transform the nonlinear load-flow problem into an ordinary linear circuit problem. It was noticed that, in a lossless* system, the power flow through a line remains constant and that, in any system, the algebraic sum of powers entering any node is zero. Both of these are characteristics of current. Therefore, in a lossless system, the nodal power injections may be assumed nodal currents. The solution to the load-flow

---

* Throughout this thesis the word "loss" always refers to "complex power loss". A lossless system, likewise, refers to an ideal system with zero impedances.

problem for the former case would then be the same as the solution of the current flow problem for the latter.

At the same time we know that, in any practical power system, the losses represent only a small percentage of the generated power. Therefore, we expect the above power/current analogy to be approximately valid for a practical power system. Furthermore, we expect this approximation to be even better if we know the (approximate) values of the line losses for our system. As a matter of fact it can be easily shown that, for a DC system, knowing the exact values of line losses means that we can transform the nonlinear load-flow problem into an exact linear equivalent. The variables in the linear equivalent model, are the square of the voltages in the original DC system.

Of course, for an ordinary AC system, the problem is far more complicated. On the one hand, for P-V buses, only the real power injections are specified and, on the other hand, the magnitudes of voltages for these buses are constrained. The problem is further complicated by the fact that there is not a one to one relationship between the voltages of our original AC network and the voltages which will be obtained by assuming nodal power injections to be currents. Hence, for an AC system, even if the exact values of line losses are known, the problem cannot be solved in one step. Using certain approximations, however, would result in an iterative algorithm for the digital solution of the load-flow problem. This algorithm, which may be derived using the expression for the line losses of the system, is called the line-loss algorithm.

From a mathematical point of view, the line-loss algorithm transforms the load-flow equations into a set of "almost linear" equations in terms of voltage magnitudes squared and voltage phase angles.

The nonlinear terms in the resulting formulation are very small. These terms are grouped together and taken to the known side of the equations. Their values are set to some initial estimates (usually zero) in the beginning of the process and are calculated using the updated nodal voltages at each subsequent iteration.

2.1 Derivation of the Algorithm

The algorithm can be derived in two different ways. The first is to use circuit analysis concepts and power/current analogy, while the second is to derive the algorithm by manipulating the power-flow equations from a mathematical point of view. The former approach led to the development of the algorithm while the latter approach was later used to indicate the relationship between the equations used in the line-loss algorithm and the exact load-flow equations. The following presentation will be made in the same order. The strictly mathematical derivation is given in section 3.1.

The idea, as mentioned earlier, is to extend the analogy that exists between the power flows and the currents of a lossless system, to practical systems for which the losses are usually very small. Assume a branch (i,k), represented - as shown in Fig. 2.1 - by an admittance y_{ik} connecting nodes i and k.

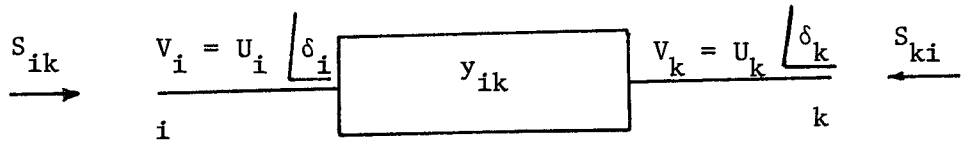


Fig. 2.1. A Simple Branch Representation.

We know that, in a lossy system, the power "sent" from node i through the branch (i,k) is different from the power "received" by node k. The difference between the two (or more precisely, the algebraic sum of the power flow from node i to node k and the power flow from node k to node i) is equal to the complex power losses in the branch (i,k):

$$S^{\ell}_{ik} = [U_i^2 + U_k^2 - 2U_i\, U_k \cos(\delta_i - \delta_k)]y^*_{ik} \qquad (2\text{-}1)$$

where U_i and δ_i (U_k and δ_k) represent the magnitude and the phase angle of voltage at node i (node k) and S^{ℓ}_{ik} is the complex power loss in the branch (i,k).

Now, if we subtract the losses from the actual power flow in branch (i,k), the branch can be considered lossless. In other words, after subtracting the line losses, the power flow in the branch remains constant and can be treated as current. In order to subtract the line losses, which are in reality distributed along the branch, we can either introduce a new node at the center of the branch (i,k) whose total injected power equals negative of S^{ℓ}_{ik} in (2-1), or we can subtract $S^{\ell}_{ik}/2$ from the powers flowing into both nodes i and k. The first approach resembles the way the T-equivalent of a line is obtained while the second approach would be similar to finding the π-equivalent of the line. The two methods are identical in every way except that the former requires introduction of new nodes into the system while the latter does not. For this reason, the second approach was chosen.

Subtracting half of the line losses, S^{ℓ}_{ik} as given in (2-1), from the powers flowing <u>into</u> nodes i and k, would leave two identical values. The fact that these values, which still have the dimension of power, are equal indicate that we can use them as the power flowing

through a lossless line and, subsequently, use them as currents(*). We represent the resulting power flow from node i to node k, after subtracting $\frac{S^{\ell}_{ik}}{2}$, by the symbol S'_{ik} to show that: 1) it is power, and, 2) it is different from the actual power flow (before subtracting the losses) which we call S_{ik}. Using equations (1-1), (1-2) and (2-1), the expression for S'_{ik} can be easily found:

$$S'_{ik} \triangleq S_{ik} - \frac{S^{\ell}_{ik}}{2}$$

$$= \tfrac{1}{2}\,[U_i^2 - U_k^2 - 2j\,U_i\,U_k\,\sin(\delta_i - \delta_k)]\,y^*_{ik} \qquad (2\text{-}2)$$

where j is the complex operator, $\sqrt{-1}$.

As mentioned earlier, we can now treat S'_{ik} as a current flowing through the branch. Equation (2-2) indicates that the best results would be obtained when the admittance of the branch is conjugated, as shown in Fig. 2.2.

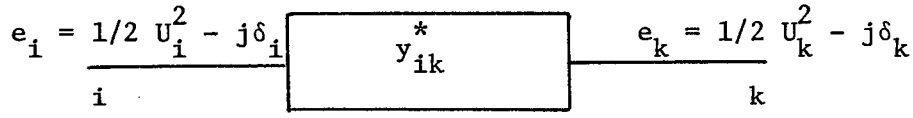


Fig. 2.2. The Branch Corresponding to that of Fig. 2.1.

If current S'_{ik} flows through the branch (i,k) of Fig. 2.2, the voltage across the branch would be

$$e_i - e_k = \frac{U_i^2}{2} - \frac{U_k^2}{2} - j\,U_i\,U_k\,\sin(\delta_i - \delta_k) \qquad (2\text{-}3)$$

(*) The value of this "power flow" through the lossless line is exactly equal to the power that was flowing through the middle point of the original branch.

where e_i and e_k represent the nodal voltages for the branch of Fig. 2.2.

The real part of each voltage, e_i, can be readily equated to $\frac{U_i^2}{2}$. This would be consistent with the results obtained for a DC network. The imaginary parts of the voltages cannot be so easily related to the voltages of the original branch, as the imaginary part of (2-3) represents a coupling between the voltages of nodes i and k. However, we can write:

$$U_i \; U_k \sin(\delta_i - \delta_k) = \delta_i - \delta_k - \varepsilon_{ik} \qquad (2\text{-}4)$$

where ε_{ik} is a correction term which is usually very small since in practical power systems all the voltage magnitudes are approximately 1 P.U. and all the phase differences across the branches are small.

Using (2-3) and (2-4), we can assume the imaginary part of e_i to be $-\delta_i$ and modify the current S'_{ik} according to the value of the correction term ε_{ik}. In other words, using

$$e_i = \tfrac{1}{2} U_i^2 - j\,\delta_i \qquad i = 1, \ldots, n \qquad (2\text{-}5)$$

would result in a current equal to $S'_{ik} - j\,\varepsilon_{ik}\,y^*_{ik}$ flowing through the branch (i,k) of Fig. 2.2. For convenience, from now on, we refer to the original system as the (I-V) system and to the system whose admittances and voltages are as shown in Fig. 2.2, as the $(S\text{-}U^2)$ system.

Writing the voltage/current relationship - equation (1-1) - for the $(S\text{-}U^2)$ system, and considering the fact that, Y_{ik}, the element (i,k) of the nodal admittance matrix has the value of $-y_{ik}$, we obtain:

$$S_i - \sum_{k=1}^{n} \frac{S^{\ell}_{ik}}{2} + j \sum_{k=1}^{n} Y^*_{ik}\,\varepsilon_{ik} = \sum_{k=1}^{n} Y^*_{ik} \left(\frac{U_k^2}{2} - j\delta_k\right) \qquad (2\text{-}6)$$

$$i = 1, \ldots, n$$

The above equation has to be slightly modified when the branches of the (I-V) system are represented, as shown in Fig. 2.3, by their π-equivalents.

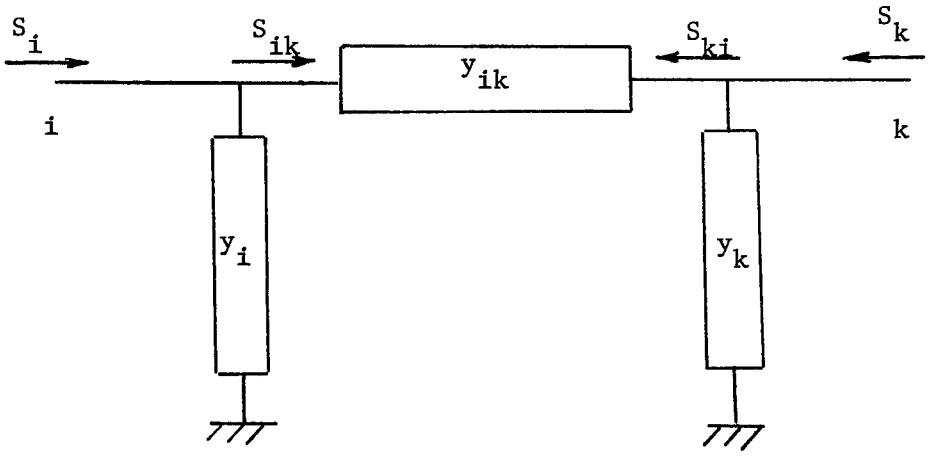


Fig. 2.3. π-Representation of Branch (i,k).

The problem can be handled by treating each "leg" of the π-equivalent as the simple branch in Fig. 2.1, but there is an easier approach: first we subtract the power consumed by each π-equivalent "leg" from the net power injection at the corresponding node. The remaining values would represent power flows through simple branches, like that in Fig. 2.1. Therefore, equation (2-6) becomes:

$$S_i - U_i^2\, y_i^* - \sum_{k=1}^{n} \frac{S_{ik}^{\ell}}{2} + j \sum_{k=1}^{n} Y_{ik}^*\, \varepsilon_{ik} = \sum_{k=1}^{n} Y_{ik}^* \left(\frac{U_k^2}{2} - j\delta_k\right) \qquad (2\text{-}7)$$

where y_i represents the total shunt admittance at node i:

$$y_i = \sum_{k=1}^{n} Y_{ik} \qquad (2\text{-}8)$$

Using (2-8), equation (2-7) can be rewritten as:

$$S_i - \sum_{k=1}^{n} \frac{S_{ik}^{\ell}}{2} + j \sum_{k=1}^{n} Y_{ik}^{*} \varepsilon_{ik} = \sum_{\substack{k=1 \\ k\neq i}}^{n} Y_{ik}^{*} \left(\frac{U_k^2}{2} - j\,\delta_k\right)$$

$$+ \frac{U_i^2}{2} (Y_{ii}^{*} + y_i^{*}) - j\,\delta_i (Y_{ii}^{*} - y_i^{*}) \qquad (2\text{-}9)$$

$i = 1, \ldots, n$

The above equation represents what we call the line-loss algorithm. It is the current flow equation for the $(S\text{-}U^2)$ system. At the same time, it has to (and does(*)) correspond to the power-flow equation for the (I-V) system. Note that irrespective of the fact that the branches of the (I-V) system may have been represented by their π-equivalents, the corresponding branches in the $(S\text{-}U^2)$ system are always represented as in Fig. 2.2.

By separating the real and imaginary equations in (2-9) we obtain the final form of the line-loss algorithm:

$$P_i - \sum_{k=1}^{n} \frac{P_{ik}^{\ell}}{2} + \sum_{k=1}^{n} \varepsilon_{ik} B_{ik} = \sum_{\substack{k=1 \\ k\neq i}}^{n} \left(G_{ik} \frac{U_k^2}{2} - B_{ik}\,\delta_k\right)$$

$$+ (G_{ii} + g_i) \frac{U_i^2}{2} - (B_{ii} - b_i)\,\delta_i \qquad (2\text{-}10a)$$

$$Q_i - \sum_{k=1}^{n} \frac{Q_{ik}^{\ell}}{2} + \sum_{k=1}^{n} \varepsilon_{ik} G_{ik} = \sum_{\substack{k=1 \\ k\neq i}}^{n} \left(-B_{ik} \frac{U_k^2}{2} - G_{ik}\delta\delta_k\right)$$

$$- (B_{ii} + b_i) \frac{U_i^2}{2} - (G_{ii} - g_i)\,\delta_i \qquad (2\text{-}10b)$$

(*) See section 3.1.

where:

$$S_i \triangleq P_i + jQ_i \quad ; \quad S^{\ell}_{ik} \triangleq P^{\ell}_{ik} + jQ^{\ell}_{ik}$$

$$Y_{ik} \triangleq G_{ik} + jB_{ik} \quad ; \quad y_i \triangleq g_i + jb_i$$

For each P-Q bus, there are two unknowns, U_i and δ_i, and both equations (2-10a) and (2-10b) are written. For each P-V bus, there is only one unknown, δ_i, and only equation (2-10a) can be written. There are no equations written for slack buses whose voltages are known ahead of time. In the resulting set of equations, all the U terms corresponding to P-V buses and all the U and δ terms corresponding to the slack buses are calculated and moved to the known side of the equations.

2.2 The Iterative Process

Once the set of equations in (2-10) is formed, it will be used in the following iterative process:

(1) Initially, all the ε_{ik}'s are set to zero. The line losses are set to their initial estimates when such estimates are available. Otherwise, they are set to zero.

(2) The resulting set of linear equations is directly solved using ordered Gaussian elimination and back substitution. The results will provide an initial set of voltages.

(3) If the present set of voltages satisfy the power constraints the problem is solved. Otherwise, using the present values of voltages and equations (2-1) and (2-4), the new values of ε_{ik} and S^{ℓ}_{ik} are calculated and the known side of the equations (2-10a) and (2-10b) are updated.

(4) The linear set of equations is solved again, using the updated known vector. Note that since the matrix of coefficients remains constant, the triangularization/back substitution process need to be performed on the known vector only.

(5) The new values of voltages obtained as the result of step 4 replace the previous values of voltages. The process then continues with step 3.

The algorithm is very fast and has reliable convergence. It is very fast because of the fact that the matrix of coefficients remains constant throughout the process; it is triangularized only once. The statement with respect to reliability of the convergence is not easy to prove. Indeed, the usual practice for confirming the reliability of a particular algorithm is to test it by using several numerical examples. With respect to the line-loss algorithm, the test results confirm the reliability of convergence. The following explanation is added to indicate why consistently good convergence characteristics can be expected from this algorithm.

The basic equations of the algorithm, equations (2-10a) and (2-10b), represent the following recursion formula:

$$S + F(X^{(\nu)}) = A \cdot X^{(\nu+1)} \qquad \nu = 0, 1, \ldots, n$$

$$F(X^{(o)}) = 0 \tag{2-11}$$

where S is the vector of injected active and reactive powers, F(X) is the nonlinear vector comprising all the line loss terms and ε terms in (2-10), A is the matrix of coefficients in (2-10) and X is the vector of unknown voltage magnitudes squared and voltage phase angles. Superscript

(ν) indicates the iteration number. As mentioned earlier, for any practical power system, the loss terms and the ε terms are very small. In other words, for practical power systems we have

$$\frac{||F(X)||}{||S||} << 1 \qquad (2\text{-}12)$$

where $||\cdot||$ indicates the norm of a vector.

From (2-11) and (2-12) we can conclude that the condition for (2-11) to converge onto the final solution is that matrix A should be "well-conditioned". Although there is no definition for what is a "well-conditioned" system and what is not, the matrices with dominant diagonal elements are normally considered to be well conditioned. Hence, since the matrix of coefficients, A in (2-11), which is directly related to the nodal admittance matrix, has dominant diagonal terms, it is well conditioned and, therefore, (2-11) converges.

We can also use the condition number, μ, of A to show that it is a well conditioned matrix. Here again, there is not a set of values of μ corresponding to the "well conditioned" matrices and another set corresponding to the "ill conditioned" ones. However, the matrices whose condition numbers are of about the same order as the matrix itself are considered to be well conditioned. The condition number of A was calculated for a few of the test systems and it was found that the above condition was satisfied. The condition numbers for Test Systems 3 (IEEE 14-bus system) and 5 (IEEE 30-bus system) were 10.89 and 24.46, respectively.

Although the condition numbers may be used to calculate an upper bound for the voltage mismatches at the end of each iteration, such bounds would be far greater than the actual mismatches and, therefore, would have no practical significance.

2.3 Programming and Storage Requirements

In matrix form, we have a double-row and a double-column for each P-Q bus; a single-row and a single-column for each P-V bus; and no row and column for slack buses. The "element" (i,k) of the matrix of coefficients would be a submatrix whose dimensions are given in Table 2.1.

k bus type / i bus type	P-Q	P-V
P-Q	2 x 2	2 x 1
P-V	1 x 2	1 x 1

Table 2.1. Dimensions of "Element" (i,k) of the Matrix of Coefficients.

The elements of these submatrices are the elements of the (modified) nodal admittance matrix. Therefore, if there is no branch connecting buses i and k, the corresponding (i,k) element would be null. It follows that:

(1) The matrix of coefficients is highly sparse;

(2) The matrix of coefficients has exactly the same structure as the Jacobian matrix of the Newton-Raphson algorithm in polar coordinates;

(3) The matrix of coefficients remains constant throughout the algorithm; and

(4) Using $(-\delta_i)$ and $\frac{U_i^2}{2}$ as our variables and assuming symmetry for the nodal admittance matrix, the matrix of coefficients would also be symmetrical except in some of its diagonal "elements".

The last proposition can be verified by examining equations (2-10a) and (2-10b): The "element" (i,k) of the matrix of coefficients, when both i and k are P-Q buses and $i \neq k$, would be:

$$E_{ik} = \left[\begin{array}{c|c} B_{ik} & G_{ik} \\ \hline G_{ik} & -B_{ik} \end{array}\right] \qquad \begin{array}{l} i \in P\text{-}Q \\ k \in P\text{-}Q \\ i \neq k \end{array} \tag{2-13}$$

which is equal to the transpose of "element" (k,i). It can be easily seen that this is still valid when one or both of the buses are of P-V type. Therefore, we have symmetry as far as the off-diagonal "elements" are concerned.

The diagonal "elements" of P-V buses are scalar $(B_{ii} - b_i)$ and, as such, do not alter the symmetry of the matrix. For P-Q type buses, however, the diagonal "elements" are:

$$D_{ii} = \left[\begin{array}{c|c} B_{ii} - b_i & G_{ii} + g_i \\ \hline G_{ii} - g_i & -(B_{ii} + b_i) \end{array}\right] \qquad i \in P\text{-}Q \tag{2-14}$$

which would not be symmetric if $g_i \neq 0$. It must be mentioned, however, that g_i is usually equal to zero. Even in cases where $g_i \neq 0$, it can be accounted for in other ways such that the matrix of coefficients becomes completely symmetrical.

The above features of the matrix of coefficients play an important role in programming the algorithm: Since this matrix is highly sparse, the ordered Gaussian elimination can be effectively utilized to minimize the storage requirements; and since it remains unchanged

throughout the process, the triangularization process need be carried out only once. Furthermore, we can exploit the (almost) symmetry of this matrix and further reduce the storage requirements: It can be easily shown that the upper half of a triangularized symmetrical matrix contains all the information necessary for carrying out forward and back-substitution processes on any R.H.S. vector. In this case, however, due to slight asymmetry, we have to store an additional n_{PQ} elements.

Taking advantage of the fact that the matrix of coefficients contains all the information about the nodal admittance matrix, the storage requirements of the algorithm, which are equal to those of the standard Newton-Raphson algorithm, can be further reduced. This results in a considerable increase in computation time but may be desirable for smaller computers.

2.4 The Numerical Results

A load-flow program (LFP(*)) was written - mostly in FORTRAN IV language - to compare the performance of the various algorithms. The program, exploiting the sparsity of the matrices, can handle very large systems. It contains the line-loss algorithm (including the variations of this algorithm to be presented in the following chapters), as well as the well known Newton-Raphson and Fast Decoupled (Newton-Raphson) Load-Flow algorithms.

All the algorithms share the same input/output as well as some other routines which are common to all of them. Furthermore, since all the programs were written by the author and all the tests were performed on the same digital computer - UBC's IBM 370/168 - the results are not biased by programming or computer efficiency.

(*) See Appendix A for details.

Several test systems of various sizes and configurations were used to compare the algorithms. These are listed in Table 2.2.

System No.	No. of Buses	No. of Lines	No. of Transf.	Source
1	5	6	0	[G21]
2	6	7	2	[Y4]
3*	14	17	3	[R2]
4	21	29	3	[D1]
5*	30	37	4	[R2]
6	33	23	11	[G14]
7	38	39	9	[G16]
8*	57	63	17	[R2]
9	93	99	57	[G13]
10	138	219	75	[G6]

Table 2.2. Test Systems Used.

* IEEE Test Systems

All the recorded computation times are CPU time spent in the main part of the algorithm; in other words, the time taken for input/output operations, formation of the nodal admittance matrix, dynamic storage allocations, etc., are not included. This ensures that the recorded times can be taken as very good measures of comparing the speed of various algorithms.

Tables 2.3 and 2.4 compare the line-loss algorithm with the well-known Newton-Raphson algorithm and the Fast Decoupled Load-Flow.

System	N-R	FDL	line loss alg.
1	2	2	2
2	2	2½	3
3*	2	2½	3
4	2	2½	2
5*	2	2	2
6	3	4	NC
7	3	2½	6
8*	2	3	3
9	3	6	12
10	4	4½	13

Table 2.3. Comparison of Number of Iterations.
(NC: No Convergence)

System	N-R	FDL	line loss alg.
1	7	9	7
2	13	12	13
3*	36	31	32
4	79	54	49
5*	112	75	70
6	119	101	NC
7	162	98	153
8*	336	224	218
9	808	563	836
10	2337	1091	1751

Table 2.4. Comparison of Computation Times.
(Times in milliseconds; NC: No Convergence)

All the results were obtained by specifying a tolerance of .01 P.U. (1 MW/1 MVAR) on maximum power mismatch. In the case of the Newton-Raphson algorithm and the Fast Decoupled Load-Flow, a flat-start was used. For the line-loss algorithm two different alternatives were tested: (1) setting the initial estimates of line losses to zero and; (2) using a flat-start to calculate the initial values of line losses and ε's. The two approaches proved to be identical*.

It can be seen from Tables 2.3 and 2.4 that, even when the (approximate) values of line losses are not specified initially, the line-loss algorithm performs very satisfactorily. For most systems, the algorithm converges in a few iterations and its overall computation time is less than that of the Newton-Raphson algorithm. Of course, the algorithm performs even better when some initial estimates of line losses are available.

Figures 2.4, 2.5 and 2.6 show the typical voltage magnitude and phase angle variations for various test systems. It is interesting to note the similarity of these patterns to that of an underdamped, critically damped or overdamped second-order system.

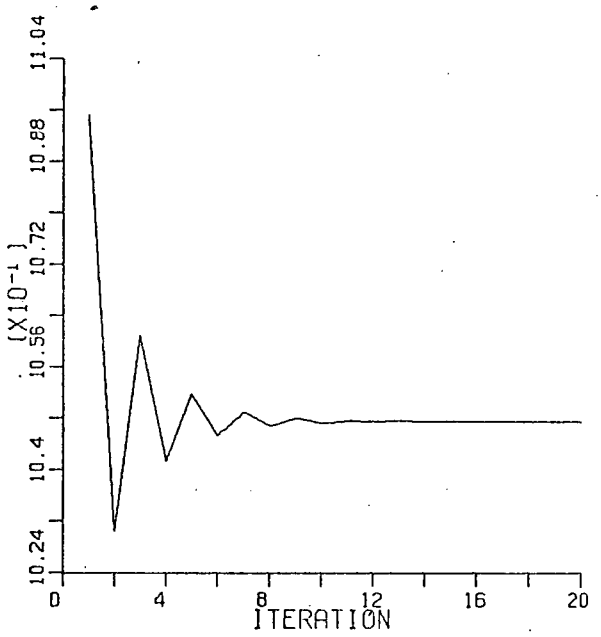
Nevertheless, the line-loss algorithm, in this present form, may have a considerable computational round-off error which, in some cases, can delay or even prevent convergence. This can be seen by examining equations (2-10a) and (2-10b). At a point fairly close to the

---

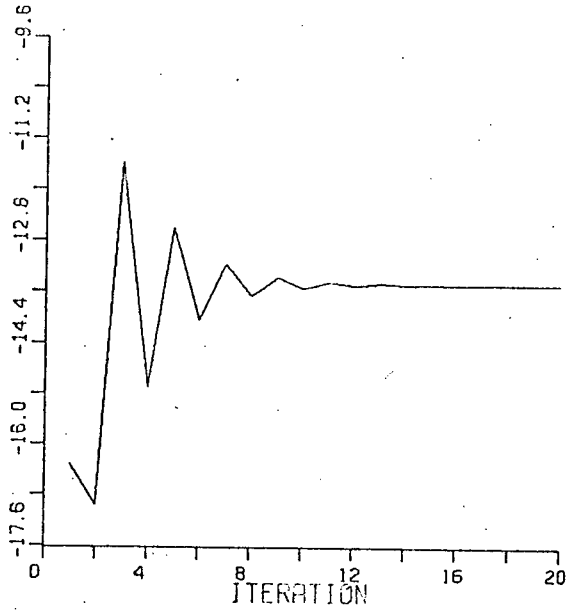
* The number of iterations were exactly the same while, because of the increased computation during the initial stage of the latter case, the computing times increased slightly.

final solution, the corrected R.H.S. values of equations differ by a very small percentage from their previous values. The resulting corrections to voltage magnitudes and phase angles would also be a small percentage of their previous values. Since the equations are in terms of the voltage magnitudes and phase angles and not in terms of corrections to these values, the roundoff error may completely offset the effect of these corrections. In other words, at some point in the process, the corrections are lost due to numerical errors and it would be impossible to get any closer to the solution point. This is precisely what happens in the case of Test System 5.

The line-loss algorithm, however, can be reformulated to overcome this problem. In the next chapter, first the strictly mathematical derivation of the line-loss algorithm is presented. Then the equations are slightly modified and reformulated in terms of incremental changes to variables. The new formulation does not have the roundoff error problem and as well has several other advantages over the basic line-loss algorithm.

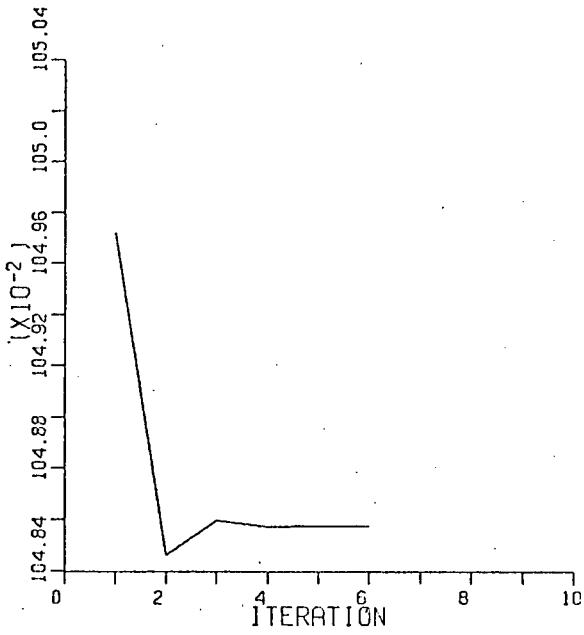


(a)

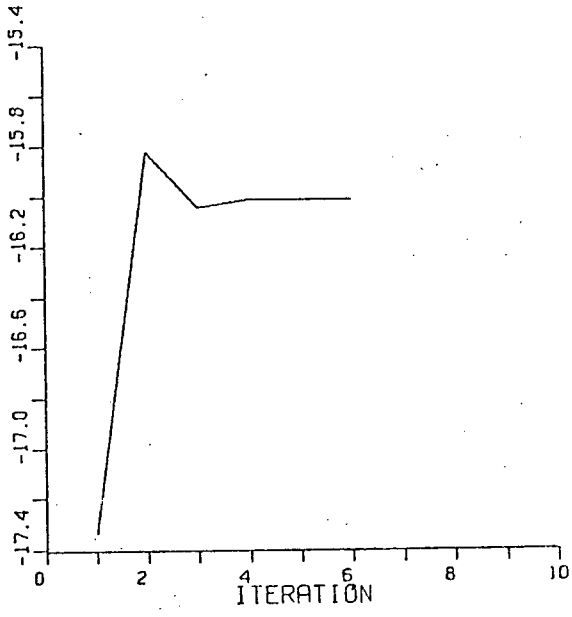


(b)

Fig. 2.4. Typical Voltage Magnitude (a) and Phase Angle (b) Variations for Test System 9.

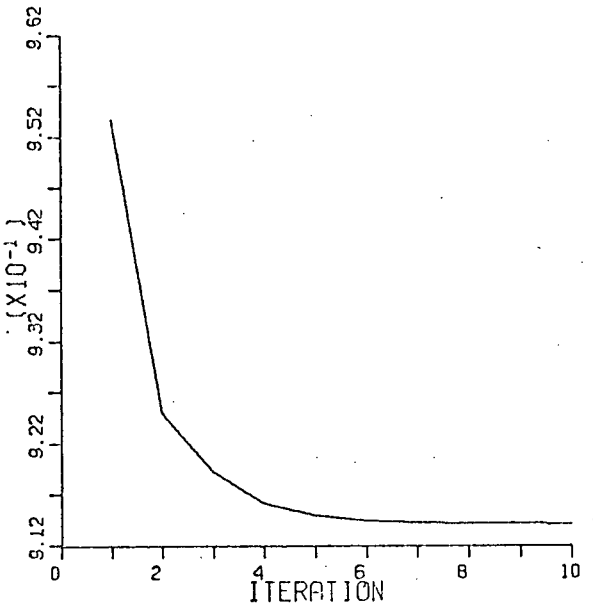


(a)

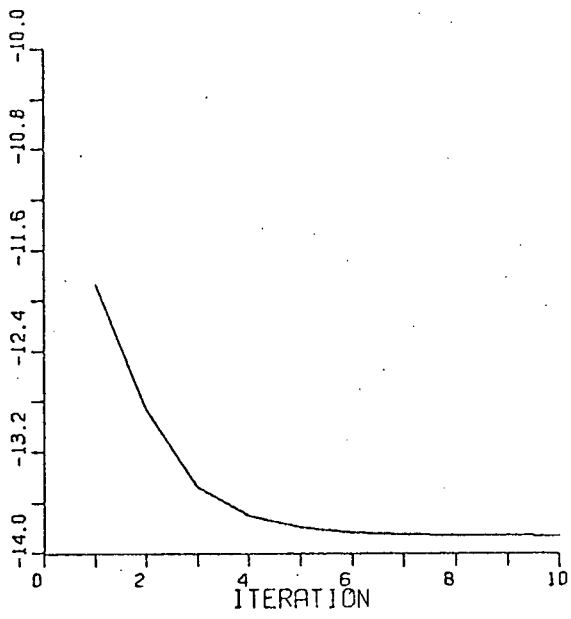


(b)

Fig. 2.5. Typical Voltage Magnitude (a) and Phase Angle (b) Variations for Test System 3.



(a)



(b)

Fig. 2.6. Typical Voltage Magnitude (a) and Phase Angle (b) Variations for Test System 6.

3. THE INCREMENTAL-CHANGE LINE-LOSS ALGORITHM

In previous chapter the line-loss algorithm was derived using circuit theory concepts and current/power analogy. Since the algorithm represents the exact expression of the power flows in the system, we must be able to derive it directly from the power flow equations. It must be mentioned, however, that we are able to do so precisely because we know what we are looking for and why. Without such knowledge, there would have been no reason to "regroup" the terms in this particular way while there are almost endless other possibilities.

In this chapter, first we present the new way of deriving the basic equation of the line-loss algorithm. This derivation enables us to proceed to reformulate the equations in terms of the incremental changes to variables. The modified version, which we call the incremental-change line-loss algorithm, has all the advantages of the basic line-loss algorithm while it does not have the serious drawback of being vulnerable to roundoff errors. It also takes less computation time per iteration and less storage as compared with the basic line-loss algorithm.

3.1 Derivation of the Algorithm

For an n-bus system the total complex power, S_i, injected at any bus, i, is given by:

$$S_i = V_i \sum_{k=1}^{n} Y_{ik}^* V_{ik}^* = \sum_{\substack{k=1 \\ k\neq i}}^{n} Y_{ik}^* (V_i V_k^*) + Y_{ii}^* U_i^2 \qquad (3\text{-}1)$$

where V_i represents the complex voltage at bus i, Y_{ik} represents the (i,k)th element of the nodal admittance matrix, U_i is the magnitude of V_i and * indicates the conjugate of a complex number. At the same time,

if y_{ik} is the value of admittance connecting buses i and k in the π-equivalent representation of the branch (i,k), we have the following expression for the complex power losses in that branch:

$$S^{\ell}_{ik} = (V_i - V_k) y^*_{ik} (V_i - V_k)^*$$

$$= y^*_{ik}[U_i^2 + U_k^2 - 2\ \mathrm{Re}(V_i\ V_k^*)] \qquad (3\text{-}2)$$

where Re( ) represents the real part of a complex number. Now, from the fact that the element (i,k) of the nodal admittance matrix is equal to the negative of y_{ik}; i.e.:

$$Y_{ik} = -y_{ik} \qquad (3\text{-}3)$$

it follows that:

$$\mathrm{Re}(V_i\ V_k^*) = \tfrac{1}{2}[U_i^2 + U_k^2 + \frac{S^{\ell}_{ik}}{Y^*_{ik}}] \qquad (3\text{-}4)$$

We also know that the imaginary part of $V_i\ V_k^*$ is:

$$\mathrm{Im}(V_i\ V_k^*) = U_i\ U_k\ \sin(\delta_i - \delta_k) \qquad (3\text{-}5)$$

which we assume to be

$$U_i\ U_k\ \sin(\delta_i - \delta_k) = \delta_i - \delta_k - \varepsilon_{ik} \qquad (3\text{-}6)$$

where ε_{ik} is the correction, or error term. Note that equation (3-6) presents no approximation.

From equations (3-4), (3-5) and (3-6) it follows that:

$$V_i\ V_k^* = \tfrac{1}{2}[U_i^2 + U_k^2 + \frac{S^{\ell}_{ik}}{Y^*_{ik}}] + j(\delta_i - \delta_k - \varepsilon_{ik}) \qquad (3\text{-}7)$$

Replacing for $V_i\ V_k^*$ in equation (3-1) and rearranging the terms we obtain:

$$S_i - \sum_{\substack{k=1\\k\neq i}}^{n} \frac{S_{ik}^{\ell}}{2} + j \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*}\, \varepsilon_{ik} = \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*} \left(\frac{U_k^2}{2} - j\, \delta_k\right)$$

$$+ \left(\frac{U_i^2}{2} + j\, \delta_i\right) \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*} + Y_{ii}^{*}\, U_i^2 \qquad (3\text{-}8)$$

Considering the fact that:

$$\sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik} + Y_{ii} = y_i \qquad (3\text{-}9)$$

equation (3-8) becomes

$$S_i - \sum_{\substack{k=1\\k\neq i}}^{n} \frac{S_{ik}^{\ell}}{2} + j \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*}\, \varepsilon_{ik} = \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*} \left(\frac{U_k^2}{2} - j\, \delta_k\right)$$

$$+ \frac{U_i^2}{2} (Y_{ii}^{*} + y_i^{*}) - j\, \delta_i (Y_{ii}^{*} - y_i^{*}) \qquad (3\text{-}10)$$

which is exactly the same as equation (2-9), the basic equation of the line-loss algorithm. The recursion formula, (2-11) in the line-loss algorithm, uses the above equation in the following way:

$$S_i - \sum_{\substack{k=1\\k\neq i}}^{n} \frac{S_{ik}^{(\nu)}}{2} + j \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*}\, \varepsilon_{ik}^{(\nu)} = \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*} \left(\frac{U_k^2}{2} - j\, \delta_k\right)^{(\nu+1)}$$

$$+ \frac{U_i^{2\,(\nu+1)}}{2} (Y_{ii}^{*} + y_i^{*}) - j\, \delta_i^{(\nu+1)} (Y_{ii}^{*} - y_i^{*}) \qquad (3\text{-}11)$$

where superscripts (ν) and $(\nu+1)$ denote the iteration cycle. At the same time, we know that equation (3-10) was obtained without any approximation and, therefore, it expresses the <u>exact</u> relationship between the

nodal power injections and nodal voltages. It can be written for any set of voltages and their corresponding powers. In other words, if we use $S_i^{(\nu)}$ to represent the calculated value of injected power at bus i at the end of iteration cycle ν, we have:

$$S_i^{(\nu)} - \sum_{\substack{k=1\\k\neq i}}^{n} \frac{S_{ik}^{\ell\,(\nu)}}{2} + j \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*}\, \varepsilon_{ik}^{(\nu)} = \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*} \left(\frac{U_k^2}{2} - j\,\delta_k\right)^{(\nu)}$$

$$+ \frac{U_i^{2(\nu)}}{2} \left(Y_{ii}^{*} + y_i^{*}\right) - j\,\delta_i^{(\nu)} \left(Y_{ii}^{*} - y_i^{*}\right) \qquad (3\text{-}12)$$

Equation (3-12) does not represent any recursion formula. On the contrary it expresses the exact power-flow equations at the end of iteration cycle ν. Now, subtracting equations (3-11) and (3-12) we obtain:

$$\Delta S_i^{(\nu)} = \sum_{\substack{k=1\\k\neq i}}^{n} Y_{ik}^{*} \left[\Delta \frac{U_k^{2(\nu)}}{2} - j\,\Delta\,\delta_k^{(\nu)}\right] + \Delta \frac{U_i^{2(\nu)}}{2} \left(Y_{ii}^{*} + y_i^{*}\right)$$

$$- j\,\Delta\,\delta_i^{(\nu)} \left(Y_{ii}^{*} - y_i^{*}\right) \qquad (3\text{-}13)$$

where:

$$\Delta S_i^{(\nu)} \triangleq S_i - S_i^{(\nu)}$$

= power mismatch at the end of iteration cycle ν;

$$\Delta \frac{U_i^{2(\nu)}}{2} \triangleq \frac{U_i^{2(\nu+1)}}{2} - \frac{U_i^{2(\nu)}}{2}$$

and

$$\Delta\delta_i^{(\nu)} \triangleq \delta_i^{(\nu+1)} - \delta_i^{(\nu)}$$

Note that in the R.H.S. of equations (2-10) and (3-10) we included <u>all</u> the buses connected to bus i. However, the constant terms, corresponding to the voltage magnitudes of P-V buses and to the voltage magnitudes and phase angles of slack buses, cancel out in the subtraction and do not appear in the R.H.S. of equation (3-13).

Since equation (3-13) applies to any (ν), we drop the superscript (ν) from this equation. Then, separating the real and imaginary equations in (3-13), we obtain the final equations used in the incremental-change line-loss algorithm:

$$\Delta P_i = \sum_{\substack{k=1 \\ k\neq i}}^{n} \left(G_{ik}\, \Delta \frac{U_k^2}{2} - B_{ik}\, \Delta\delta_k\right) + (G_{ii} + g_i)\, \Delta \frac{U_i^2}{2} - (B_{ii} - b_i)\, \Delta\delta_i \qquad (3\text{-}14a)$$

$$\Delta Q_i = \sum_{\substack{k=1 \\ k\neq i}}^{n} \left(-B_{ik}\, \Delta \frac{U_k^2}{2} - G_{ik}\, \Delta\delta_k\right) - (B_{ii} + b_i)\, \Delta \frac{U_i^2}{2} - (G_{ii} - g_i)\, \Delta\delta_i \qquad (3\text{-}14b)$$

For any P-Q bus, both equations (3-14a) and (3-14b) can be written and two correction terms have to be calculated. For a P-V type bus, only equation (3-14a) can be written and only one correction term has to be calculated. Slack buses have no unknowns and none of the above equations can be written for them.

3.2 The Iterative Process

The set of equations in (3-14), in matrix form, have exactly the same matrix of coefficients as the set of equations in (2-10). The reason for this is that although in the R.H.S. of equation (2-10) the

terms corresponding to any bus connected to i are included, nevertheless, all the U terms corresponding to P-V buses and all the U and δ terms corresponding to slack buses are known. The matrix of coefficients is not affected by these terms and, hence, there is the complete similarity between the matrices of coefficients.

Another point to be mentioned here is that, contrary to the basic line-loss algorithm of equation (2-10), the incremental-change line-loss algorithm cannot be started without having an initial set of estimates for the voltages. Once a set of estimates are available, they can be corrected using equation (3-14). If the approximate values of line losses are available, we can obtain these initial estimates by performing one cycle of the basic line-loss algorithm. Note that, since the matrices of coefficients are exactly identical, this step can be very easily implemented. If no approximate values of losses are initially specified, we can either use a flat-start or perform one cycle of the line-loss algorithm with losses set to zero. Both approaches were found to be equally good.

The iterative process can be summarized as follows:

(1) If some initial estimates for the line losses of the system are available, a cycle of the basic line-loss algorithm is performed to obtain the initial values of voltages. Otherwise, a flat-start is used(*).

(2) Using the present estimates of voltages, the power flows of the system are calculated. If they are within the desired tolerance of their specified values, the problem is solved. Otherwise,

---

(*) Alternatively, one cycle of the basic line-loss algorithm with initial estimates of line losses set to zero may be performed. LFP permits both options. See Appendix A.

(3) the power mismatches calculated in step 2 are used in equation (3-14) to calculate the corrections to the voltage magnitudes and phase angles. The set of equations is solved by Gaussian elimination and back substitution. Note that since the matrix of coefficients remains constant it need be triangularized only once. At each step of the process, the triangularization/back substitution process is only performed on the vector of power mismatches.

(4) Using the answers obtained as the result of step 3, the present estimates of the voltages are corrected. The new values of voltages, then, would be the new estimates and the process continues with step 2.

The algorithm has a very fast convergence since the matrix of coefficients remains constant and does not have to be formed and triangularized at every step. Furthermore, the algorithm is very reliable. The test results confirm this fact. Furthermore, all the comments made about the reliability of the basic line-loss algorithm apply here as well. In fact, this version of the line-loss algorithm is more reliable since it is formulated in terms of incremental changes to variables and, therefore, is not vulnerable to roundoff errors.

3.3 Programming and Storage Requirements

As mentioned earlier, the matrix of coefficients in this algorithm is exactly the same as that in the basic line-loss algorithm of Chapter 2. Therefore, all the comments about programming and storage requirements of the latter apply here as well; viz, if we use $(-\Delta\delta_i)$ and $\Delta \frac{U_i^2}{2}$ as our variables:

(a) The matrix of coefficients is a highly sparse, constant and almost symmetrical matrix which can be very easily constructed; therefore,

(b) sparsity programming techniques and ordered Gaussian elimination can be effectively used to minimize the storage requirements. The matrix of coefficients is triangularized only once. Only the upper half of the triangularized matrix of coefficients plus an additional n_{PQ} elements need be stored; and,

(c) if need be, the storage requirements of the algorithm can be reduced to those of the Y-matrix algorithms, at the expense of increased computation time.

The storage requirements of the algorithm are, at most, equal to those of the standard Newton-Raphson algorithm. This statement can be verified by comparing the structure of the matrix of coefficients in this algorithm and the Jacobian matrix in the Newton-Raphson algorithm. The "element" (i,k) in both of them is a submatrix whose dimensions depend on the type of buses i and k. For the same bus types, these submatrices will have the same dimensions in both cases. Also, in both cases, the slack buses are not included in these matrices. This means that the structure of the matrix of coefficients in (3-14) (and also in (2-10)) is exactly the same as that of the Jacobian matrix in the Newton-Raphson algorithm with polar coordinates.

However, the Newton-Raphson algorithm does not store the entire Jacobian matrix, but only its triangularized half. In other words, the Jacobian matrix is formed and triangularized, row by row, and the triangularized Jacobian is the only thing that is stored. Exactly the

same procedure can be followed in case of the incremental-change line-loss algorithm. In the Newton-Raphson algorithm this is possible because the Jacobian has to be formed at every step of the iterative process and, hence, no information other than the upper half of the triangularized matrix is required. In the incremental-change line-loss algorithm this is possible because the matrix of coefficients is symmetrical. Thanks to this symmetry, only the upper half of the triangularized matrix of coefficients would contain sufficient information to carry out triangularization and back substitution processes on any right hand side vector. However, we need to store all the diagonal elements, as well as n_{PQ} additional elements if the matrix is not completely symmetrical. These elements can be stored in place of the active and reactive power mismatches in the Newton-Raphson algorithm.

The storage scheme used (in LFP) for calculating and storing the elements of the matrix of coefficients can be summarized as follows:

All the triangularized elements of the matrix of coefficients, along with proper pointers, are stored in a vector, from now on referred to as the vector of the triangularized elements. In another vector the pointers to the starting location of each row are stored. These pointers are positive for P-Q rows (double rows) and negative for P-V rows (single rows). The elements of the triangularized row are stored in locations starting with the one specified by the pointer for that row and ending with the one specified by the next pointer. The first one or three locations, depending upon whether the row is P-V or P-Q, are used for storing the diagonal elements. This is followed by an integer pointer indicating the column number of an "element" in the row. This pointer, too, is positive for P-Q columns (double columns) and negative for P-V

columns (single columns). The number and the order of the elements that follow this pointer are dependent upon, and determined by the sign of both row and column pointers. In the location following the last item for that particular column (which may be one, two or four locations away) the column pointer for the next element in the row is stored, followed by the value of its elements. And so it continues.

Each row is formed and triangularized in a vector, from now on referred to as the working-row, prior to being stored in the vector of the triangularized elements. Each element of the working-row consists of four locations in which the four possible elements of the submatrix (i,k) are stored. It also has a column pointer which follows the same rule as any other pointer with respect to P-Q and P-V elements. The length of the working-row is sufficient to store a full row of the matrix of coefficients. Once the row is formed, a linear combination of the previously stored rows is added to it such that it will not have any elements in the column range that has been processed before. During this process, new elements may be created and some of the elements may be deleted and modified. New elements are added to the end of the working-row and deleted elements are indicated by a zero column pointer. At the end of the process, only the elements with non-zero column pointers will be stored in the compact vector of the triangularized elements.

It should be mentioned that the storage requirements of the basic line-loss algorithm is roughly the same as those of the incremental-change line-loss algorithm. The former requires slightly more space for storing additional terms corresponding to P-V and slack buses and other information necessary for updating the known vector.

3.4 The Numerical Results

The incremental-change line-loss algorithm was programmed using sparse matrix techniques and tested upon the test systems listed in Table 2.2. Tables 3.1 and 3.2 compare the required number of iterations and the CPU time taken by this algorithm with those of the Newton-Raphson algorithm and the Fast Decoupled (Newton-Raphson) Load-Flow. The number of iterations and the CPU times taken by the basic line-loss algorithm are also included for comparison. All the results were obtained by specifying a tolerance of .01 P.U. (1 MW/ MVAR) on maximum power mismatch. Furthermore, for the Newton-Raphson algorithm and the Fast Decoupled (Newton-Raphson) Load-Flow, a flat-start was used while, the incremental-change line-loss algorithm was started by using equations (2-10a) and (2-10b) during the first iteration and switched to equations (3-14a) and (3-14b) thereafter. The correction terms and the line-losses were assumed to be initially zero.

It is evident from Tables 3.1 and 3.2 that the incremental-change line-loss algorithm has all the advantages of the basic line-loss algorithm while, at the same time, it is not vulnerable to computational roundoff error. In case of Test System 6, where the original line-loss algorithm failed to converge, the new version converges in 5 iterations. Table 3.2 also shows that the incremental-change line-loss algorithm is faster than the basic line-loss algorithm, because the line losses and correction terms need not be computed at each iteration. Even without any approximate values of line losses specified, the incremental-change line-loss algorithm converges considerably faster than Newton-Raphson algorithm and, in most cases, even faster than the Fast Decoupled (Newton-Raphson) Load-Flow.

System	N-R	FDL	line loss alg.	Inc. change line loss alg.
1	2	2	2	2
2	2	2½	3	3
3*	2	2½	3	3
4	2	2½	2	2
5*	2	2	2	2
6	3	4	NC	5
7	3	2½	6	6
8*	2	3	3	3
9	3	6	12	12
10	4	4½	13	13

Table 3.1. Comparison of Number of Iterations.
(NC: No Convergence)

System	N-R	FDL	line loss alg.	Inc. change line loss alg.
1	7	9	7	6
2	13	12	13	10
3*	36	31	32	25
4	79	54	49	42
5*	112	75	70	61
6	119	101	NC	80
7	162	98	153	113
8*	336	224	218	188
9	808	563	836	616
10	2337	1091	1751	1247

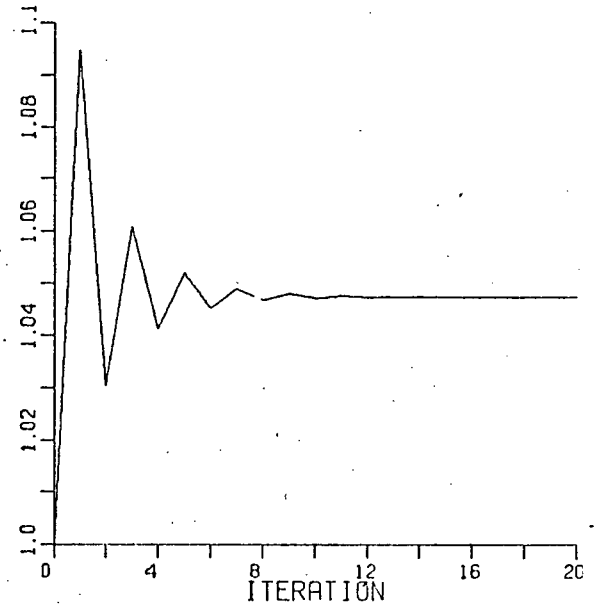
Table 3.2. Comparison of Computation Times.
(Times in milliseconds; NC: No Convergence)

The algorithm always converges in a few iterations, except for Test Systems 9 and 10, where the number of iterations required are 12 and 13, respectively. This was found to be caused by very large phase differences (up to about 70°) across some of the branches of these systems. Such large phase differences do not occur in practice. However, even in these cases, the computation time is significantly reduced as compared with the Newton-Raphson algorithm. It is interesting to note that the pattern of voltage magnitude and phase angle variations for these systems - as shown in Figures 3.1 and 3.2(*) - resemble the pattern of an under-damped second-order system.

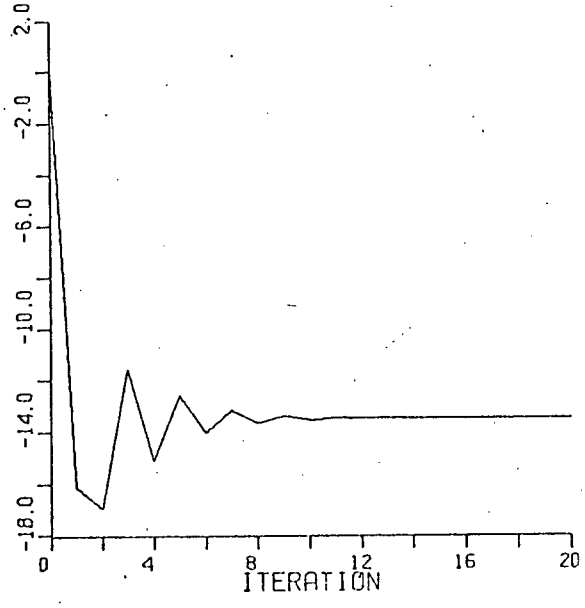
Several other modifications were tried to see if the performance of this algorithm, especially with respect to the number of iterations taken for Test Systems 9 and 10, can be further improved. Some of these actually did reduce the number of iterations considerably. Unfortunately, however, these modifications were those that required changing the matrix of coefficients at every step and, consequently, also increased the total computation time considerably. Nevertheless, their improved performance with respect to the number of iterations makes them attractive alternatives. They are also of special interest to those who may want to pursue this research work further. A few of these alternatives which showed the greatest improvement are briefly mentioned below:

One such alternative was to use another equation instead of (3-6), which would provide us with better values at the end of first iteration. Since we know that:

(*) For all the plots a flat-start was used. If one cycle of the line-loss algorithm with zero initial estimates was performed, the plots would start at the points corresponding to iteration 1.



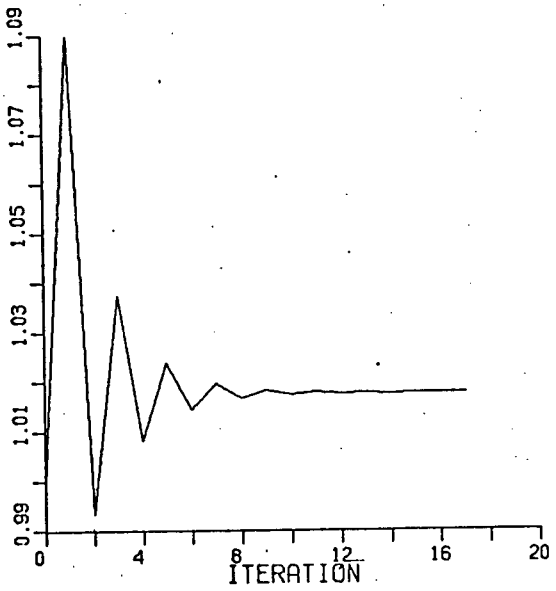
(a)



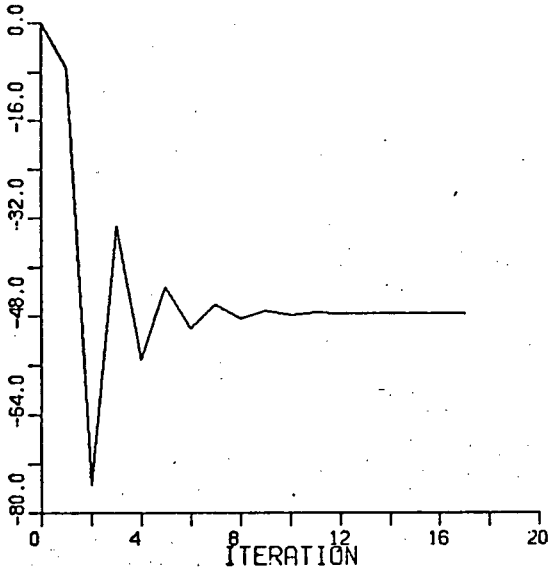
(b)

Fig. 3.1. Typical Voltage Magnitude and Phase-Angle Variations for Test System 9.

(Incremental-Change Line-Loss Algorithm with a Flat-Start)



(a)



(b)

Fig. 3.2. Typical Voltage Magnitude (a) and Phase-Angle Variations for Test System 10.

(Incremental-Change Line-Loss Algorithm with a Flat-Start)

$$\sin x \approx x - \frac{x^3}{3!} \approx x[\frac{2}{3} + \frac{\cos x}{3}] \tag{3-15}$$

we can use the following equation instead of (3-6):

$$U_i\ U_k \sin(\delta_i - \delta_k) = (\delta_i - \delta_k)[\frac{2}{3} + \frac{1}{3}\cos(\delta_i - \delta_k)] - (\text{correction term}) \tag{3-16}$$

Furthermore, we can replace for $\cos(\delta_i - \delta_k)$ an expression of the line losses. Assuming that the magnitudes of voltages are almost 1 P.U., we obtain:

$$\frac{S^{\ell}_{ik}}{2Y^*_{ik}} \approx \cos(\delta_i - \delta_k) - 1 \tag{3-17}$$

Therefore, (3-16) becomes

$$U_i\ U_k \sin(\delta_i - \delta_k) = (\delta_i - \delta_k)[1 + \frac{S^{\ell}_{ik}}{6Y^*_{ik}}] - \varepsilon_{ik} \tag{3-18}$$

where ε_{ik} represents the new correction term which is different from (and much smaller than) the correction term used in (3-6). Using (3-18) instead of (3-6), and following the same procedure as explained in section 3.1, we get:

$$S_i - \sum_{\substack{k=1\\k\neq i}}^{n} \frac{S^{\ell}_{ik}}{2} + j \sum_{\substack{k=1\\k\neq i}}^{n} Y^*_{ik}\ \varepsilon_{ik} = \sum_{\substack{k=1\\k\neq i}}^{n} \frac{U_k^2}{2} Y^*_{ik} - j \sum_{\substack{k=1\\k\neq i}}^{n} \delta_k(Y^*_{ik} + \frac{S^{\ell}_{ik}}{6})$$

$$+ \tfrac{1}{2} U_i^2(Y^*_{ii} + y^*_i) - j\ \delta_i(Y^*_{ii} - y^*_i - \sum_{k=1}^{n} \frac{S^{\ell}_{ik}}{6}) \tag{3-19}$$

which should replace the basic equation (3-10). It can be seen that the matrix of coefficients is no longer constant. Using the above equation

reduced the number of iterations taken for Test System 9 by one third. On the other hand, the total computation time was much more than that of the incremental-change line-loss algorithm. When the variable terms in the matrix of coefficients were transferred to the other side of the equations and evaluated on the basis of the present values of voltages the performance of the algorithm became worse than that of the incremental-change line-loss algorithm.

Another alternative which showed a great reduction in the number of iterations was to leave the sine terms in (3-6) intact. The derivation of the incremental-change line-loss algorithm, would then include the difference between two sine terms. In other words, $\Delta(\sin(\delta))$ will appear in equation (3-13) instead of $\Delta\delta$. However, assuming that the change in δ from one step to the next is small(*), we can write:

$$\sin[(\delta_i - \delta_k) + \Delta(\delta_i - \delta_k)] \approx \sin(\delta_i - \delta_k) + \Delta(\delta_i - \delta_k)\cos(\delta_i - \delta_k) \tag{3-20}$$

Here again, due to the cosine term in the right-hand side of (3-20), we will get a matrix of coefficients which changes from one iteration step to the next. Using this approach, the number of iterations taken for Test System 9, was reduced to 6. The computation time, on the other hand, increased threefold. Once again, taking the variable terms in the matrix of coefficients to the other side of the equations, and evaluating them on the basis of the present values of the voltages resulted in a worse performance.

(*) This assumption is indeed valid. Test results showed that after one cycle of the basic line-loss algorithm, even when the initial estimates were set to zero, the calculated voltages were within less than 10% of their final values.

Yet another approach was to use the following expression, instead of equation (2-5), for the voltages of the ($S-U^2$) system:

$$e_i = \frac{1}{2} U_i^2 - j \Delta_i \tag{3-21}$$

such that:

$$\Delta_i - \Delta_k = U_i U_k \sin(\delta_i - \delta_k) \tag{3-22}$$

for all i and k. The problem with this approach is that the phase angle for each bus can be calculated from several different routes; and the values do not necessarily agree with each other. The approach would be unsuccessful if the first route that becomes available is chosen and the phase angle is calculated from that route. On the other hand, when all the possible values for each bus are calculated and a least squares hyper-plane is fitted to these values the algorithm improves tremendously (with respect to the number of iterations). This latter approach, however, involves finding the parameters of the least squares hyper-plane, which requires the direct solution of another set of equations. Although the matrix of coefficients for the new set of equations is the same as the system's incidence matrix, nevertheless, the increased storage and computation time requirements offset the gains achieved in reducing the number of iterations from 12 to 6 (in case of Test System 9).

Many other alternatives were tried but proved to be unsuccessful. In particular, the use of various acceleration factors, including the ones that are related to power mismatches and vary from one iteration to the next, did not improve the performance of the algorithm. Except for decoupling, which improved the overall results, the best algorithm remained the one derived in section 3.1 - the incremental-change line-loss algorithm. Figures 3.3, 3.4 and 3.5 compare the performance of this

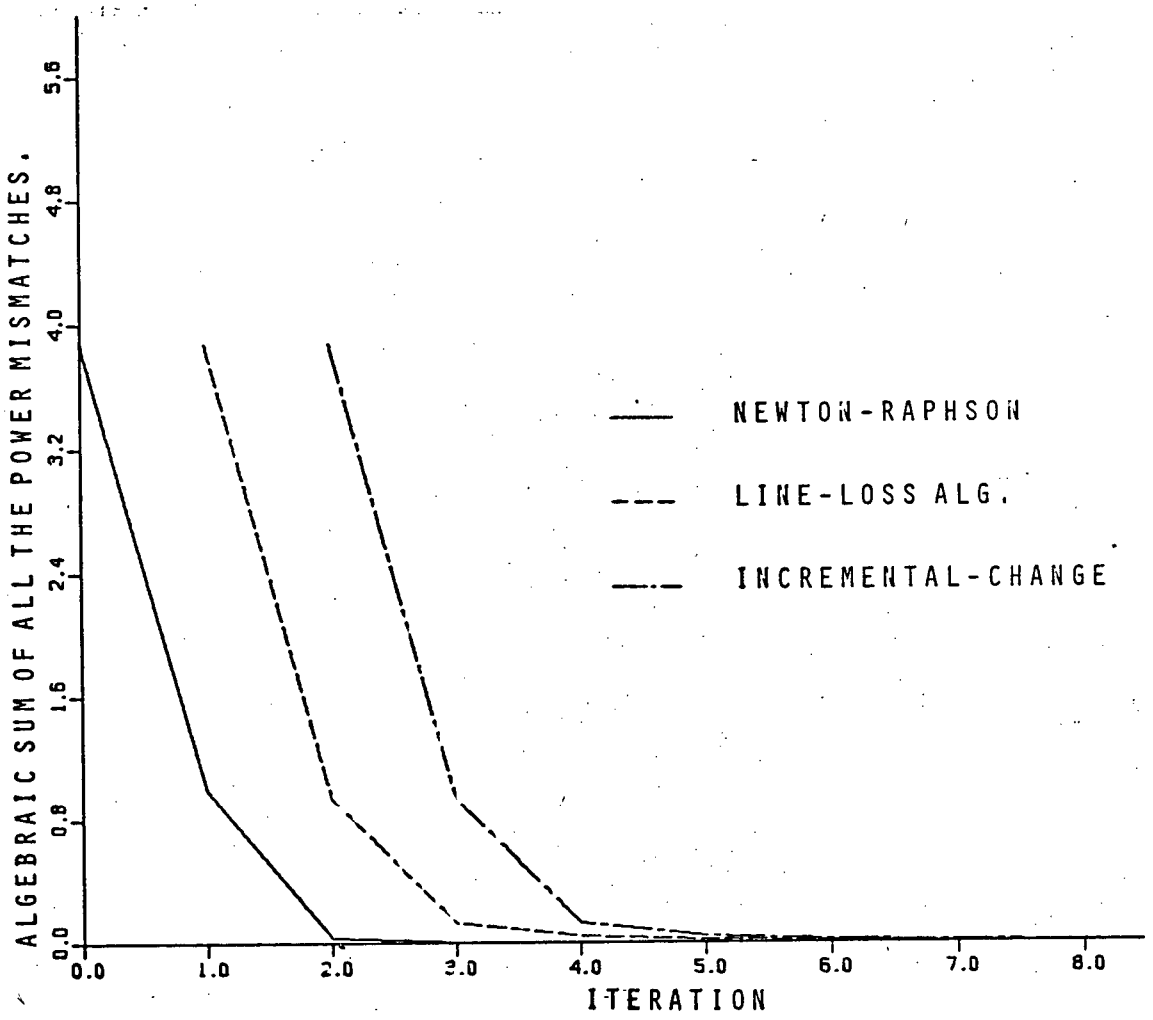


Fig. 3.3. Comparison of the Algebraic Sum of all the Power Mismatches.

(Test System 8)

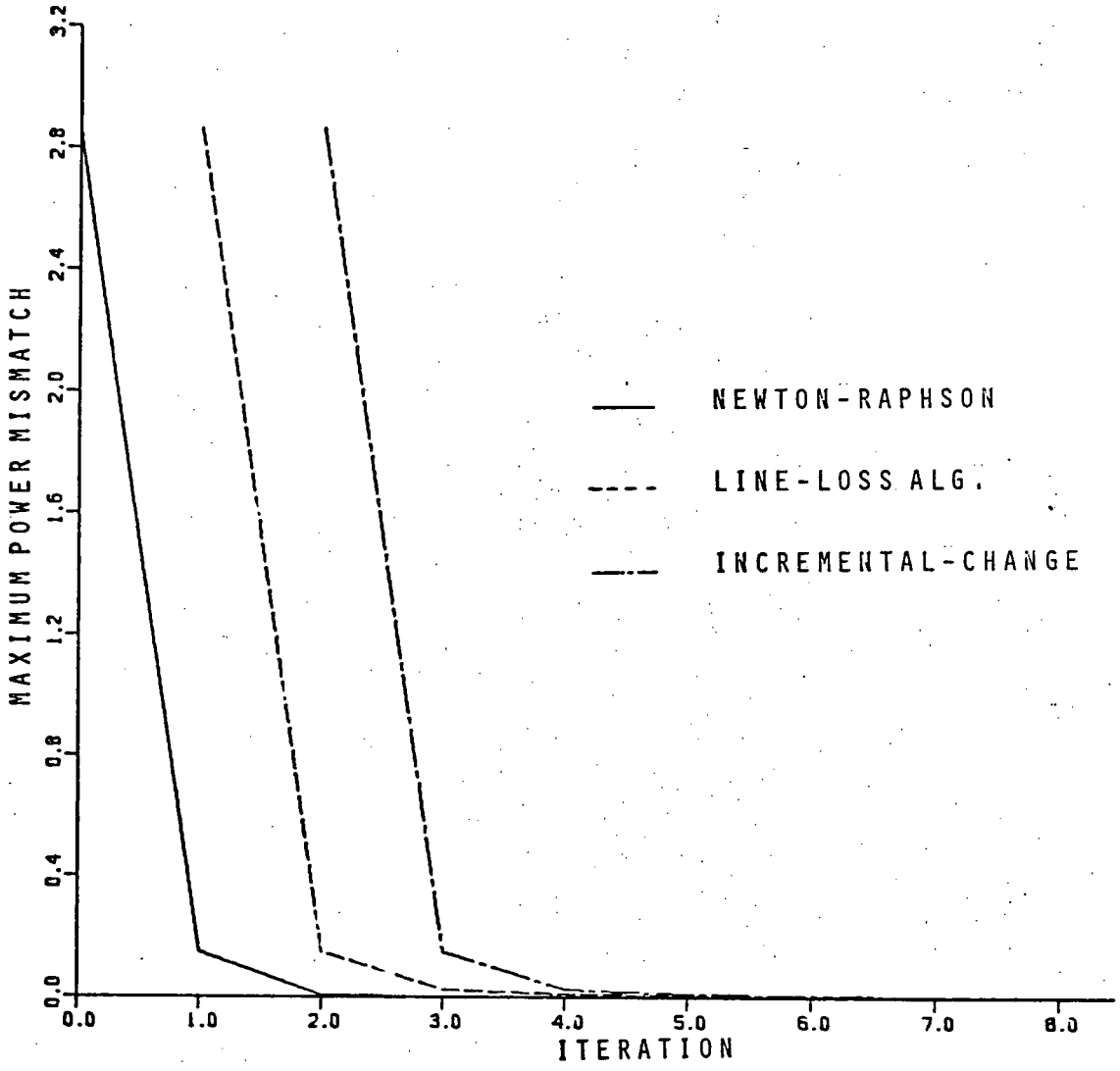


Fig. 3.4. Comparison of the Maximum Power Mismatches.

(Test System 8)

58

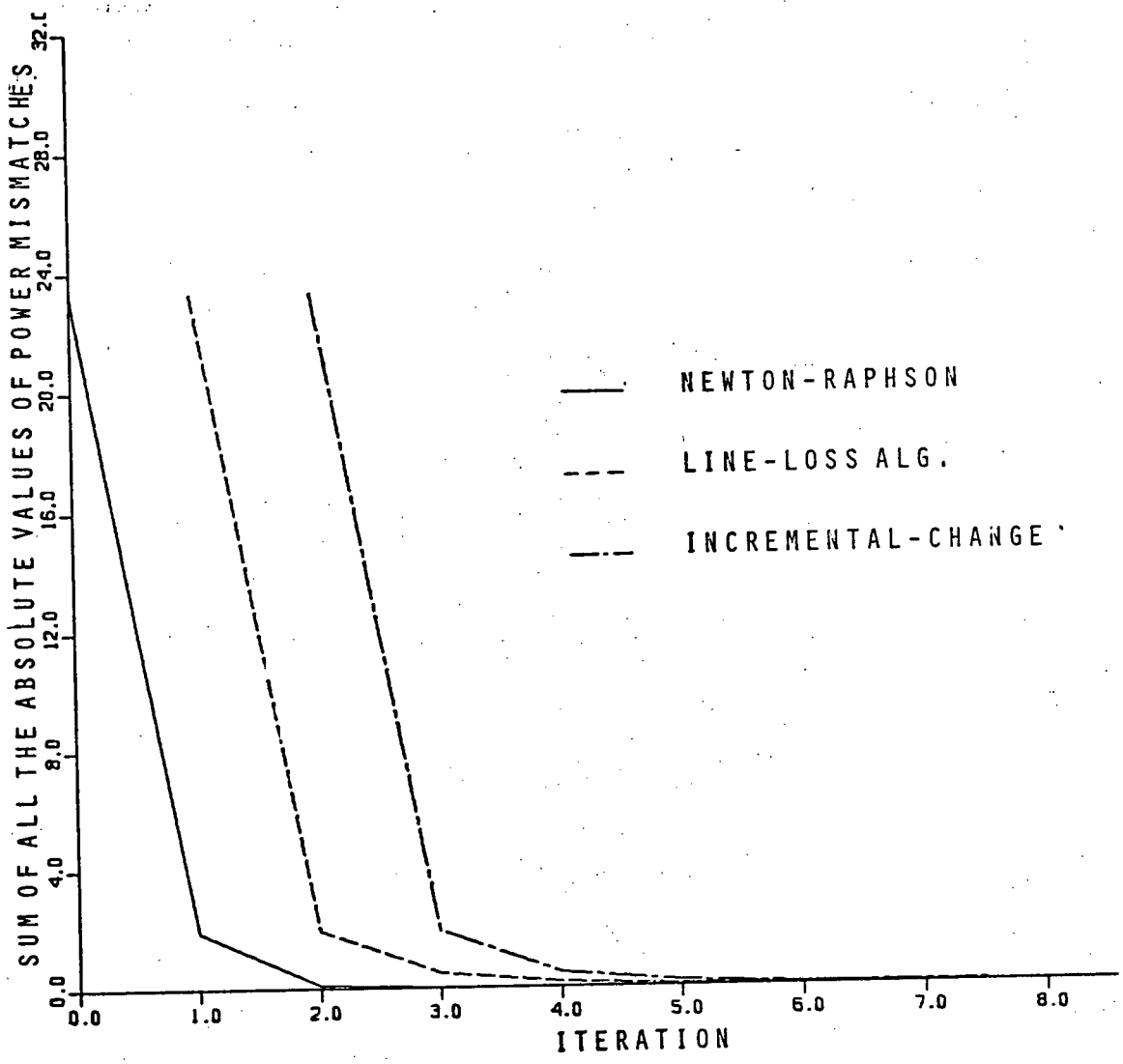


Fig. 3.5. Comparison of Sum of all the Absolute Values of Power Mismatches. (Test System 8)

algorithm, and that of the basic line loss algorithm, with the performance of the Newton-Raphson algorithm. Note that, for the sake of clarity, the three curves are displaced by one iteration step from one another. The figures show that the line-loss algorithms have a much greater rate of convergence during the first iterations than during the later iterations. This is due to the constant-gradient characteristic of these algorithms.

The overall performance of the line-loss algorithms was greatly improved when the weak (P-U) and (Q-δ) interdependencies were exploited and the incremental-change line-loss algorithm was decoupled. The decoupled version, as we will see in the next chapter, has a remarkable resemblance to the Fast Decoupled (Newton-Raphson) Load-Flow.

4. FAST DECOUPLED LINE-LOSS ALGORITHMS

In a practical power system, the interdependence between active powers and voltage magnitudes, and between reactive powers and phase angles are weak and may be neglected. This is usually referred to as "the decoupling principle". In this chapter, we exploit this feature of practical power systems and decouple the incremental-change line-loss algorithm. It will be seen that, contrary to the Newton-Raphson algorithm, applying the decoupling principle to the incremental-change line-loss algorithm will result in a stable decoupled algorithm.

Two slightly different decoupled algorithms will be presented and compared with the Fast Decoupled (Newton-Raphson) Load-Flow. Both these algorithms are very fast and reliable. Test results indicate that the performance of these decoupled versions is even better than that of the undecoupled incremental-change line-loss algorithm. Indeed, the decoupled formulations are nothing but approximations to the original formulation. The reason for their improved performance seems to be that, in the decoupled versions, the corrections are calculated and applied at the end of every "half-iteration" rather than every iteration.

4.1 Derivation of the Algorithm

The incremental-change line-loss algorithm uses equations (3-14a) and (3-14b), rewritten below,

$$\Delta P_i = \sum_{\substack{k=1 \\ k\neq i}}^{n} \left(G_{ik}\, \Delta \frac{U_k^2}{2} - B_{ik}\, \Delta\delta_k\right) + (G_{ii} + g_i)\, \Delta \frac{U_i^2}{2} - (B_{ii} - b_i)\, \Delta\delta_i \qquad (3\text{-}14a)$$

$$\Delta Q_i = \sum_{\substack{k=1 \\ k \neq i}}^{n} \left(-B_{ik}\, \Delta \frac{U_k^2}{2} - G_{ik}\, \Delta\delta_k\right) - (B_{ii} + b_i)\, \Delta \frac{U_i^2}{2}$$

$$- (G_{ii} - g_i)\, \Delta\delta_i \qquad (3\text{-}14b)$$

which, in matrix form, can be written as:

$$\begin{bmatrix} \Delta P \\ \hline \Delta Q \end{bmatrix} = \left[\begin{array}{c|c} -B' & G' \\ \hline -G'' & -B'' \end{array}\right] \begin{bmatrix} \Delta\delta \\ \hline \Delta \frac{U^2}{2} \end{bmatrix} \qquad (4\text{-}1)$$

(a) Algorithm A

Assuming that submatrices [G'] and [G"] in the above equation are null, we get:

$$[\ \Delta P\ ] = [\ -B'\ ] \cdot [\ \Delta\delta\ ] \qquad (4\text{-}2a)$$

$$[\ \Delta Q\ ] = [\ -B''\ ] \cdot [\ \Delta \frac{U^2}{2}] \qquad (4\text{-}2b)$$

This follows the decoupling technique commonly used [D11]; namely, that of neglecting P-U and Q-δ interdependencies altogether.

From equations (3-14a) and (3-14b), it is seen that the submatrices [B'] and [B"] are identical with the nodal susceptance matrix entries, except for the main diagonal elements.

(b) Algorithm B

In this algorithm we also neglect the effect of the series resistances in calculating the elements of [B']. The equations used are still (4-2a) and (4-2b) but the elements of [B'] are different:

$$B'_{ik} = \frac{1}{X_{ik}} \qquad (4\text{-}3a)$$
$$i \neq k$$

and

$$B'_{ii} = - \sum_{\substack{k=1 \\ k \neq i}}^{n} \frac{1}{X_{ik}} \qquad (4\text{-}3b)$$

In other words, the DC load-flow algorithm [D1] is used instead of equation (4-2a) for calculating the voltage phase angles. Equation (4-2b) remains unchanged. This was found, experimentally, to slightly improve the results. Stott and Alsac experienced the same results when they applied the above approximation to the decoupled Newton-Raphson algorithm [D11].

It is very interesting to compare the equations (4-2a) and (4-2b), which are the basic equations for the decoupled line-loss algorithms, with equations (1-16a) and (1-16b), which represent the Fast Decoupled (Newton-Raphson) Load-Flow. The two sets of equations are remarkably similar. Indeed, the matrices of coefficients in equations (4-2a) and (1-16a) are exactly identical. Equation (4-2b) is only slightly different from equation (1-16b); but if the approximation $\Delta \frac{U^2}{2} \approx U \, \Delta U$ is made, then the two will become almost identical. It must be mentioned, however, that equations (1-16a) and (1-16b) were obtained by making a number of practical assumptions after applying the decoupling principle while, equations (4-2a) and (4-2b) are obtained simply by neglecting P-U and Q-δ couplings. Between the two decoupled line-loss algorithms, algorithm B is more similar to the Fast Decoupled (Newton-Raphson) Load-Flow.

4.2 The Iterative Process

Like the incremental-change line-loss algorithm, the decoupled algorithms A and B need a set of initial estimates for the voltages of the system. However, unlike the former algorithm, the matrices of coefficients in the latter are not the same as that of the basic line-loss algorithm. Therefore, although it is possible to start these algorithms by first performing one cycle of the basic line-loss algorithm, this would require forming and solving a different set of equations during the first iteration step. This is a very time consuming process. For this reason, the decoupled line-loss algorithms are always started using a flat-start.

The solution of either (4-2a) or (4-2b) is considered to be one "half iteration". Prior to solving (4-2a) or (4-2b), the active or reactive power mismatches are checked against the desired tolerance. If all these mismatches are within the desired limits, the solution of the corresponding set of equations is skipped. Hence, depending upon the convergence characteristics of different systems, one set of equations may have to be solved more often than the other. The algorithm always starts by trying to solve (4-2a), because this was found (experimentally) to be always preferable. The flow-chart of Figure 4.1 summarizes the iterative process for decoupled algorithm A and B. Note that since the matrices of coefficients, [B'] and [B"], are symmetrical and constant, they need to be triangularized only once. As before, the solution of (3-2a) or (3-2b) is always obtained by using the information in the upper triangularized half of these matrices.

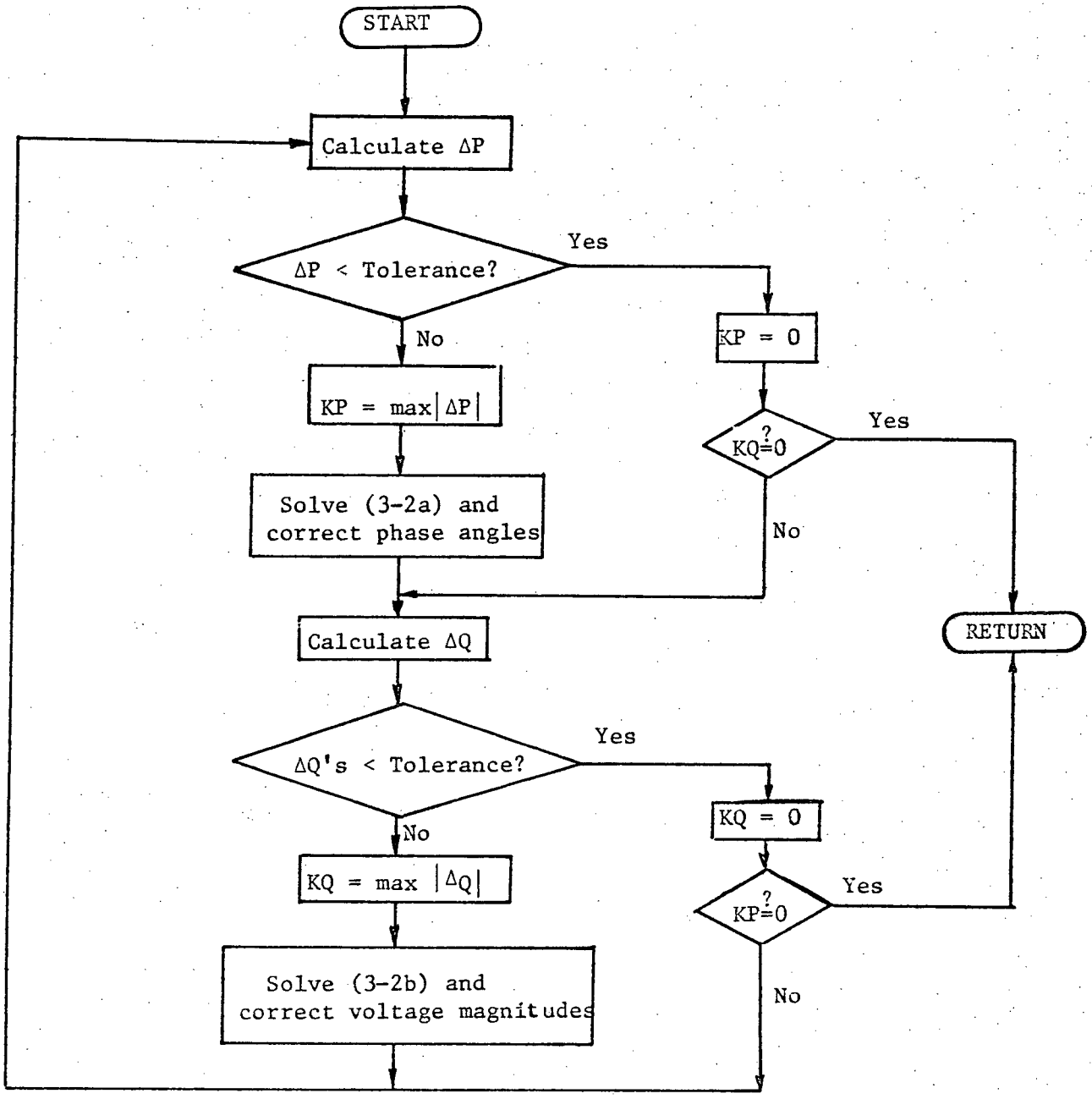


Fig. 4.1. Flow-Chart Indicating the Logic of Decoupled Algorithms.

4.3 Programming and Storage Requirements

Both [B'] and [B"], in equations (4-2a) and (4-2b), are constant, symmetrical and highly sparse matrices. Due to the sparsity of these matrices, the decoupled algorithms A and B can benefit a great deal from ordered Gaussian elimination and sparsity programming techniques. The storage requirements of these algorithms, then, would be much less than those of the Newton-Raphson algorithm and exactly the same as those of the Fast Decoupled (Newton-Raphson) Load-Flow. Furthermore, if storage is a very serious limitation, the storage requirements of these algorithms can be substantially reduced, of course, at the expense of increased computation time per iteration.

Since both matrices are symmetrical and constant, they need to be triangularized only once and only the upper half of their triangularized results need be stored. The ordering of the rows is exactly the same for [B'] and for the Jacobian matrix in the Newton-Raphson algorithm and the matrix of coefficients in the incremental-change line-loss algorithm. The same ordering is assumed for the rows of [B"]. With such ordering of the rows, however, the decoupled algorithms require just slightly over half the computer storage needed by the other algorithms mentioned above.

The storage scheme used for these algorithms, which is the same as the one used(*) for programming the Fast Decoupled (Newton-Raphson) Load-Flow, is very similar to that of the incremental-change line-loss algorithm, as explained in section 3.3. Both [B'] and [B"]

(*) This is the storage scheme used in LFP. Stott and Alsac did not mention what storage scheme they used in [D11].

are formed and triangularized, row by row, and stored in compact form in two separate vectors. There are two vectors of pointers, indicating the starting locations of each row in [B'] and [B"]. The compact forms of [B'] and [B"] consist of integer column pointers followed by respective entries of the triangularized matrices. The scheme is considerably simpler than that used for the incremental-change line-loss algorithm (and also for the Newton-Raphson algorithm) since there is no necessity of distinguishing between P-Q and P-V pointers. As was the case with the incremental-change line-loss algorithm, however, each row is formed in a working-row, combined with the previously processed rows and triangularized, and then stored in compact form. Here again, only the entries in the upper half of the triangularized matrices (plus the diagonal elements) are stored. This information is sufficient for carrying out triangularization/back substitution process on any right-hand side vector.

4.4 The Numerical Results

The results of comparison between decoupled algorithms A and B, and other algorithms are shown in Tables 4.1 and 4.2. A flat-start and a tolerance of .01 P.U. (1 MW/1 MVAR) on maximum power mismatch was specified for all the runs. The test systems are those listed in Table 2.2. The rate of convergence of these algorithms are compared with that of the Fast Decoupled (Newton-Raphson) algorithm in Figures 4.1, 4.2 and 4.3.

Since [B'] and [B"] are of smaller dimensions than the matrix of coefficients of the incremental-change line-loss load-flow algorithm, a reduction in computation time per iteration is expected as a result of

decoupling. However, Table 4.2 shows that this is not the case. The reason for this is that in the decoupled algorithms almost all the search operations are duplicated. This cannot be avoided if the flexibility of performing each half iteration separately is to be preserved. The only way to eliminate this problem would be to solve (4-2a) and (4-2b) simultaneously which, however, greatly degrades the performance of the decoupled algorithms.

Tables 4.1 and 4.2 indicate that the performance of the decoupled line-loss algorithms is comparable to that of the Fast Decoupled (Newton-Raphson) Load-Flow. This was to be expected since there is a remarkable similarity between the equations used by the two algorithms. However, in order to better compare the reliability of the decoupled line-loss algorithms A and B with that of the Fast Decoupled (Newton-Raphson) Load-Flow, it was decided to check their behaviour with respect to heavily overloaded systems. Test System 5 (IEEE 30-bus system) was overloaded, step by step, and the load-flow problem was solved for each loading level using various algorithms. The Newton-Raphson algorithm, the basic line-loss algorithm and the incremental-change line-loss algorithm all diverged for the same loading level. This breaking point was when all the power injections (both loads and generations) were 1.8 times greater than their original values.

On the other hand, all the decoupled algorithms converged for this loading level: the Fast Decoupled (Newton-Raphson) Load Flow converged in 27 half-iterations (216 milliseconds); algorithm A converged in 16 half iterations (164 milliseconds); and algorithm B converged in 20 half iterations (187 milliseconds). The differences in computation times and the number of iterations are noticeably in favour of the

System	N-R	FDL	line loss alg.	Inc. change line loss alg.	Decoupled line loss alg.	
					Alg. A	Alg. B
1	2	2	2	2	1½	1½
2	2	2½	3	3	2½	2½
3*	2	2½	3	3	7	2½
4	2	2½	2	2	3½	2½
5*	2	2	2	2	3	2
6	3	4	NC	5	3½	4½
7	3	2½	6	6	3	3
8*	2	3	3	3	3½	2½
9	3	6	12	12	4½	4½
10	4	4½	13	13	6½	4½

Table 4.1. Comparison of Number of Iterations (NC: No Convergence).

System	N-R	FDL	line loss alg.	Inc. change line loss alg.	Decoupled line loss alg.	
					Alg. A	Alg. B
1	7	9	7	6	8	7
2	13	12	13	10	12	12
3*	36	31	32	25	57	30
4	79	54	49	42	63	53
5*	112	75	70	61	87	73
6	119	101	NC	80	94	107
7	162	98	153	113	108	108
8*	336	224	218	188	235	210
9	808	563	836	616	505	495
10	2337	1091	1751	1247	1263	1063

Table 4.2. Comparison of Computation Times.
(Times in milliseconds)
(NC: No Convergence)

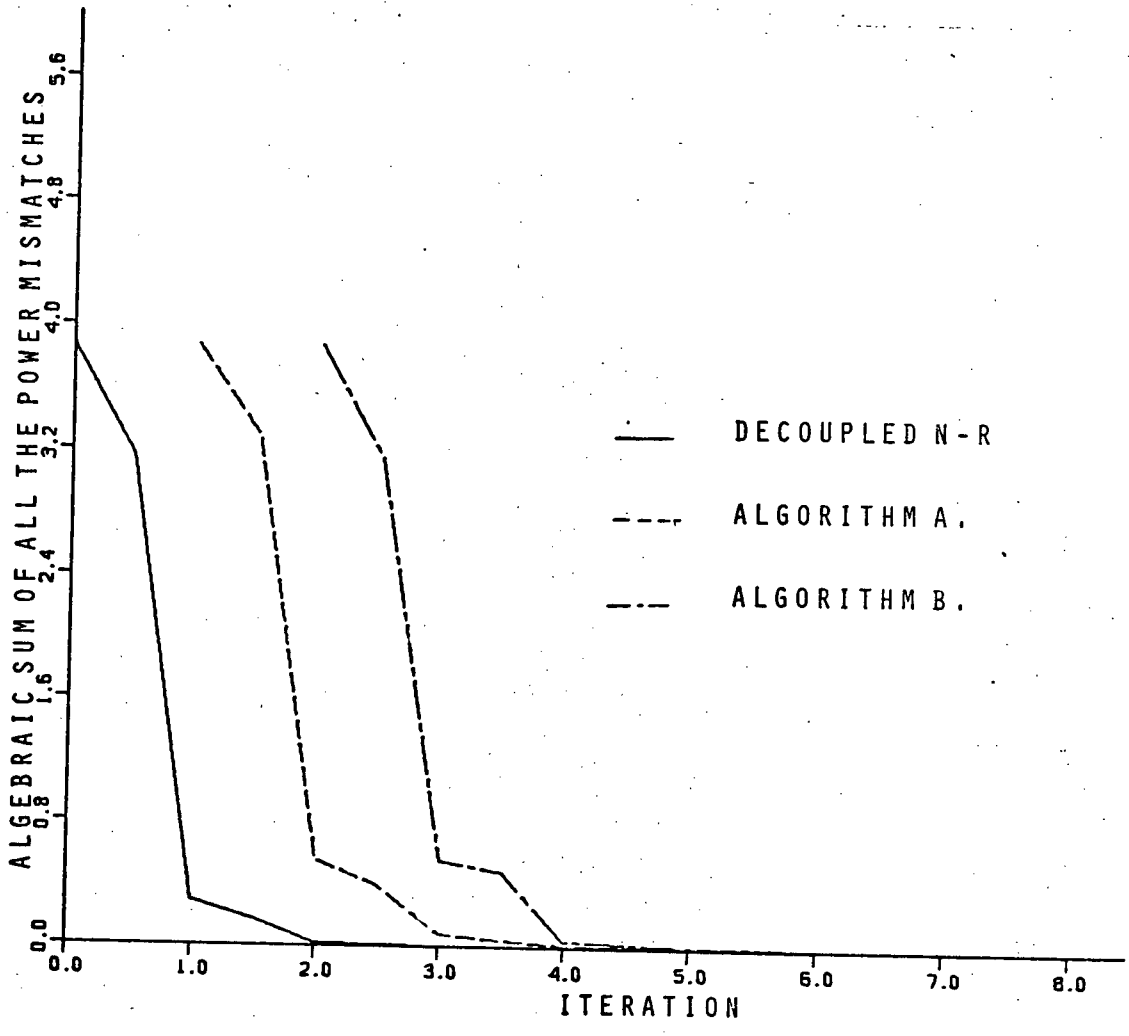


Fig. 4.2. Comparison of the Algebraic Sum of all the Power Mismatches. (Test System 8)

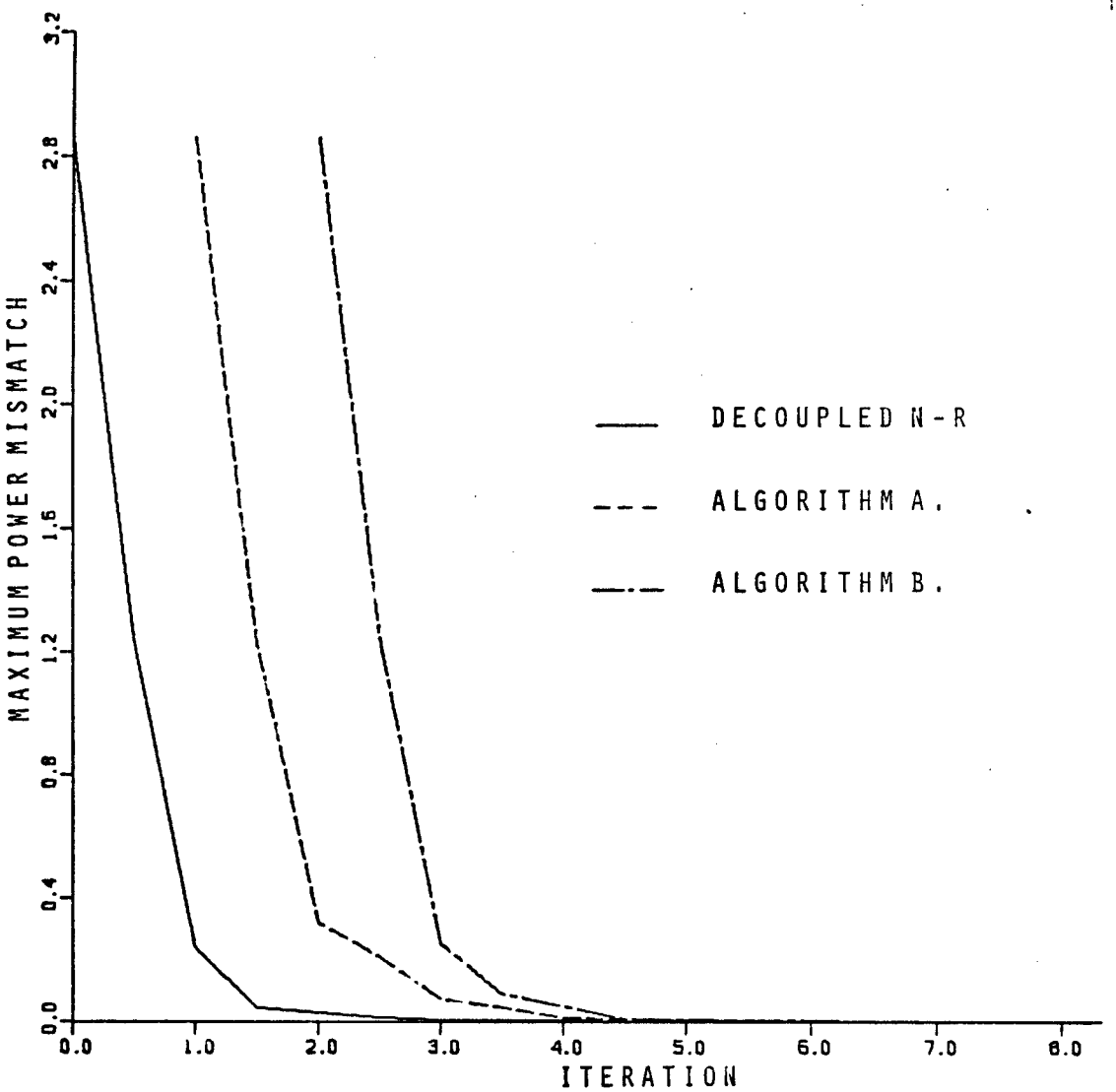


Fig. 4.3. Comparison of the Maximum Power Mismatches.

(Test System 8)

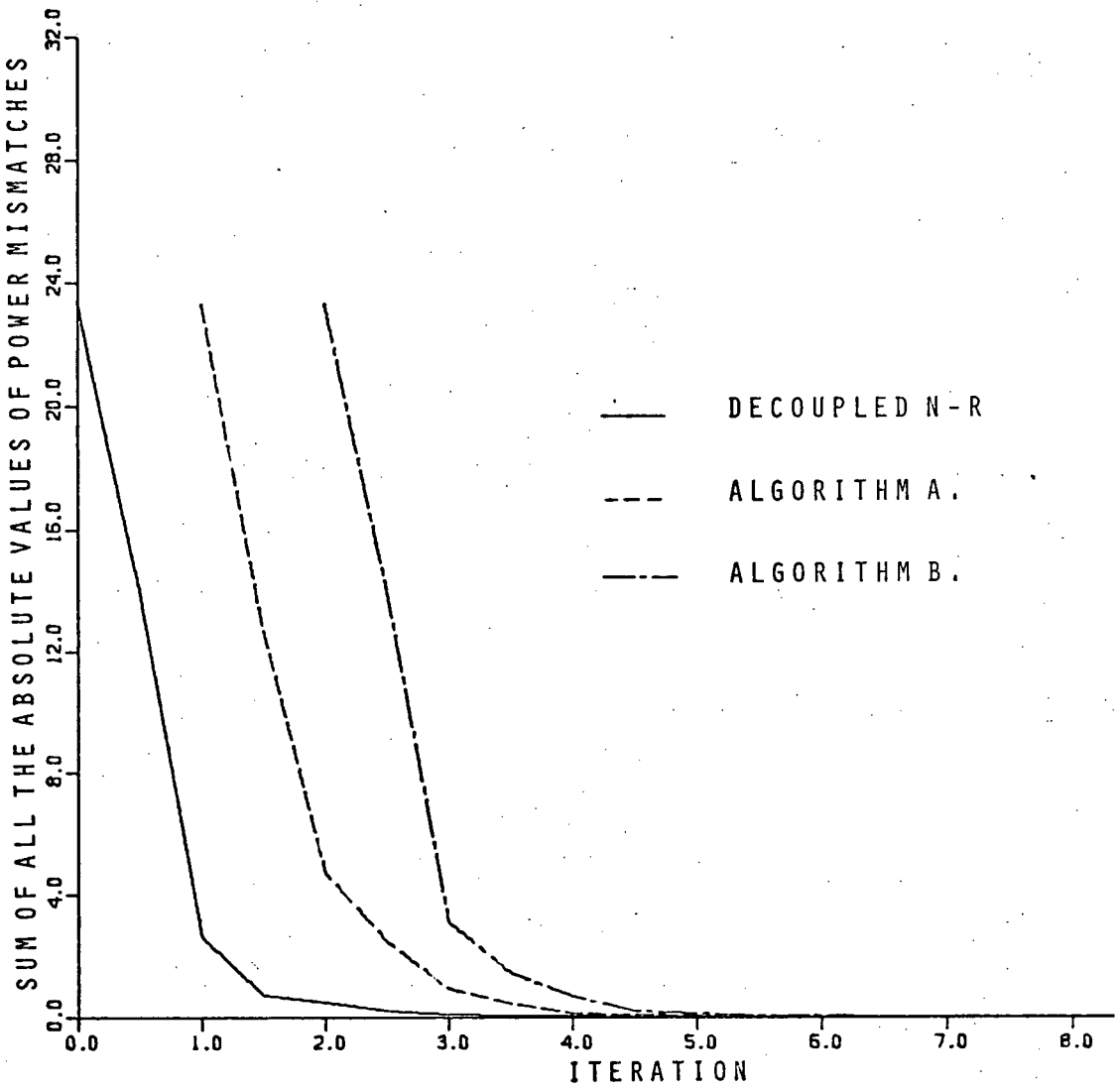


Fig. 4.4. Comparison of Sum of all the Absolute Values of Power Mismatches. (Test System 8)

decoupled line-loss algorithms, especially algorithm A, which has less similarity with the Fast Decoupled (Newton-Raphson) Load-Flow.

When the system was slightly (one per cent) more overloaded, the Fast Decoupled (Newton-Raphson) Load-Flow diverged while both decoupled line-loss algorithms converged, each taking 20 half iterations. As the overloading continued, in small one per cent steps, both decoupled algorithms converged for three more steps until, finally, algorithm B diverged. For this loading level, algorithm A was the only algorithm that solved the problem. After this point, none of the load-flow algorithms converged.

The above test indicates that while the decoupled line-loss algorithms have the same performance as the Fast Decoupled (Newton-Raphson) Load-Flow, they are more reliable. The assumption made in deriving algorithm B, slightly improves the performance of this algorithm while, at the same time, makes it slightly less reliable.

5. CONCLUSIONS

In a power system, the line losses can be used to derive the constant gradient, fast converging line-loss load-flow algorithm. The algorithm iteratively solves a set of linear equations in terms of voltage magnitudes squared and phase angles, and converges onto the final solution in a few iterations. Although the performance of the algorithm improves if the (approximate) values of line-losses are initially known, such knowledge is not a pre-requisite to using the algorithm. For practical accuracies, even when the initial estimates of line-losses are assumed zero, the line-loss algorithm converges considerably faster than the standard Newton-Raphson algorithm.

A slightly modified version of the line-loss algorithm, written in terms of the incremental changes to the voltage magnitudes squared and phase angles, performs even better than the original version. For all the test systems used, it converges considerably faster than the Newton-Raphson algorithm - in some cases it is almost twice as fast.

The decoupled versions of the incremental change line-loss algorithm have convergence characteristics similar to those of the Fast Decoupled (Newton-Raphson) Load-Flow [D11]. As compared with the undecoupled version of this algorithm, they show definite improvement with respect to the required number of iterations; but not with respect to computation time. In the decoupled algorithms, the corrections are calculated and applied at the end of every half iteration; resulting in improved overall performance and the increased computation time per iteration.

All the line-loss algorithms have very reliable convergence. In particular, the results of using these algorithms for solving the load-flow problem on heavily overloaded systems showed that the convergence of the line-loss algorithms is at least as reliable as that of the Newton-Raphson algorithm. The results also showed that the decoupled line loss algorithms are more reliable than the Newton-Raphson algorithm and the Fast Decoupled (Newton-Raphson) Load-Flow.

The storage requirements of the line-loss algorithms are, at most, equal to those of the standard Newton-Raphson algorithm. The decoupled versions of the line-loss algorithm have the same storage requirements as the Fast Decoupled (Newton-Raphson) Load-Flow. However, if storage is a serious limitation, in both cases, the storage requirements can be reduced to those of the Y-matrix algorithms.

The matrix of coefficients in the line-loss algorithms is very easy to construct and is directly related to the system nodal admittance matrix. Hence, slight changes in the system configuration can be easily reflected in the algorithm. Furthermore, approximate load-flow solutions can be obtained after performing only one iteration of the line-loss algorithm. These characteristics make the line-loss algorithms particularly attractive for study of system outages, where numerous load-flow problems, only slightly different from one another, have to be solved and only approximate answers are required.

BIBLIOGRAPHY

General:

[G1] Arsov, D. and Hammam, M.S.A.A.; "Sensitivity of Load Flow"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan. 27-Feb. 1, 1974, Paper C74 018-8.

[G2] Branin Jr., F.H.; "Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations"; IBM Journal of Research and Development; Sept. 1972; pp. 504-522.

[G3] Brown, K.M. and Conte, S.D.; "The Solution of Simultaneous Nonlinear Equations"; Proc. of A.C.M. National Meeting; 1967; pp. 111-114.

[G4] Daniels, H. and Mo-Shing Chen; "An Optimization Technique and Security Calculation for Dispatching Computers"; IEEE Trans. on PAS; Vol. PAS-91, no 3; May-June 1972; pp. 883-888.

[G5] Dayal, G., Grigsby, L.L. and Hasdroff, L.; "Quadratic Programming for Optimal Active and Reactive Power Dispatch Using Special Techniques for Reducing Storage Requirements"; Presented at the IEEE PES Summer Meeting; Portland, Oreg.; July 18-23, 1976; Paper A76 388-9.

[G6] Drakos, J.; B.C. Hydro Sample Test Data;1973.

[G7] El-Abiad, A.H., Watson, M. and Stagg, G.W.; "The Load Flow Problem - its Formulation and Solution"; Part I; AIEE Conf. Paper; 1961; Paper 61-1054.

[G8] Green Jr., R.K., Brown, J.L. and Mo-Shing Chen; "Load-Flow Fundamentals"; 21st Annual Southwestern IEEE Conference and Exhibition; San Antonio, Texas; April 23-25,1969; pp. 10A1-10A8.

[G9] Isaacson, E. and Keller, H.B.; Analysis of Numerical Methods; John Wiley & Sons, Inc; New York; 1966; Chapter 3.

[G10] IEEE Work Group; "Common Format for Exchange of Solved Load-Flow Data"; Presented at the IEEE PES Summer Meeting; San Francisco, Cal.; July 9-14, 1972; Paper C72 450-5.

[G11] Juricek, M.J., Fukutome, A. and Mo-Shing Chen; "Transportation Analysis of an Electric Power Distribution System"; Paper Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan. 25-30, 1976; Paper A76 052-1.

[G12] Kesavan, H.K. and Bhat, M.V.; "Sensitivity Analysis of Load-Flow Solutions"; Presented at the IEEE PES Summer Meeting & EHV/UHV Conf.; Vancouver, B.C., Canada; July 15-20, 1973; Paper C73 465-2.

[G13] Nordstrom, T.; B.C. Hydro Sample Test Data; 1973.

[G14] Nordstrom, T.; "Digital Load-Flow"; Report for EE 553 Course; Department of Electrical Engineering, UBC; 1967.

[G15] Ogbuobiri, E.C.; "Dynamic Storage and Retrieval Programming"; 1969 PICA Conference Record; pp. 226-232.

[G16] Sachdev, M.S. and Ibrahim, S.A.; "A Simulation Technique for Studying Real and Reactive power Flow Patterns"; Presented at the IEEE PES Summer Meeting & Energy Resources Conf.; Anaheim, Calif.; July 14-19, 1974; Paper C74 389-3.

[G17] Sasson, A.M.; "Nonlinear Programming Solutions for Load-Flow, Minimum Loss, and Economic Dispatching Problems"; IEEE Trans. on PAS; Vol. PAS-88, no 4; April 1969; pp. 399-409.

[G18] Sato, N. and Tinney, W.F.; "Techniques for Exploiting the Sparsity of the Network Admittance Matrix"; AIEE Trans. on PAS; Vol. 81; Dec. 1963; pp. 944-950.

[G19] Schlaepfer, F., Kelly, T.C. and Dewey, A.G.; "An Interactive Load-Flow Program"; PICA Conf. Paper; 1971; pp. 110-117.

[G20] Stagg, G.W. and El-Abiad, A.H.; Computer Methods in Power System Analysis; New York; McGraw Hill Book Co.; 1968.

[G21] Stevenson Jr, W.D.; Elements of Power System Analysis; 2nd ed.; McGraw Hill Book Co.; 1962.

[G22] Tinney, W.F.; "Compensation Methods for Network Solution by Optimally Ordered Triangular Factorization"; PICA Conf. Paper; 1971; pp. 377-382.

[G23] Tinney, W.F. and Walker, J.W.; "Direct Solution of Sparse Network Equations by Optimally Ordered Triangular Factorization"; Proc. IEEE; Vol. 55, no 11; Nov. 1967; pp. 1801-1809.

Load-Flow Review Papers:

[R1] Conner, D.A.; "Representative Bibliography on Load-Flow Analysis and Related Topics"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan. 28-Feb. 2, 1973; Paper C73 104-7.

[R2] Freris, L.L. and Sasson, A.M.; "Investigation of the Load-Flow Problem"; Proc. IEE; Vol. 115, no 10; Oct. 1968; pp. 1459-1470.

[R3] Laughton, M.A.; "Numerical Methods of Load Flow Analysis"; The Use of Digital Computers in Electric Power Systems; edited by J.L. Dingeley; Oriel Press; 1967.

[R4] Laughton, M.A. and Humphrey Davis, M.W.; "Numerical Techniques in the Solution of Power SystemsLoad-Flow Problems"; IEE Proc.; Vol. 111, no 9; Sept. 1964; pp. 1575-1588.

[R5] Sasson, A.M. and Jaimes, F.J.; "Digital Methods Applied to Power-Flow Studies"; IEEE Trans. on PAS; Vol. PAS-86, no 7; July 1967; pp. 860-867.

[R6] Stott, B.; "Review of Load-Flow Calculation Methods"; IEEE Proc.; Vol. 62, no 7; July 1974; pp. 916-929.

Convergence & Uniqueness:

[C1] El-Shibini, M.A.; "Effect of Problem Formulation on the Convergence of the Load-Flow Solution"; Presented at the IEEE PES Summer Meeting & Energy Resources Conf.; Anaheim, Calif.; July 14-19,1974; Paper C74 322-4.

[C2] Johnson, B.K.; "Extraneous and False Load-Flow Solutions"; IEEE Trans. on PAS; Vol. PAS-96, No. 2, March/April 1977; pp.524-534.

[C3] Klos, A. and Kerner, A.; "The Non-Uniqueness of Load-Flow Solution"; Presented at the PSCC Conference; Sept. 1-5, 1975.

[C4] Korsak, A.J.; "On the Question of Uniqueness of Stable Load-Flow Solutions"; IEEE Trans. on PAS; Vol. PAS-91; May-June 1972; pp. 1093-1100.

[C5] Rao, N.R. and Subrahmanyam, B.; "Load-Flow Studies by Newton-Raphson Method: Convergence Characteristics"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan. 28-Feb. 2, 1973; Paper C73 077-5.

[C6] Van Ness, J.E.; "Convergence of iterative Load-Flow Studies"; AIEE Trans. on PAS; Vol. 78; Feb. 1960; pp.1590-1597.

[C7] Wu, F.F.; "Theoretical Study of the Convergence of the Fast Decoupled Load-Flow"; IEEE Trans. on PAS; Vol. PAS-96, No. 1; Jan./Feb. 1977, pp. 268-275

[C8] Vetter, W.J., Moore, E. and Ganesan, K.; "On Convergence of Newton's Method for Load Flow Problems"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan. 28-Feb. 2, 1973; Paper C73 103-9.

Y-matrix Algorithms:

[Y1] Brown, R.J. and Tinney, W.F.; "Digital Solutions for Large Power Networks"; AIEE Trans, (PAS); Vol. 76; June 1957; pp. 347-355.

[Y2] Glimn, A.F. and Stagg, G.W.; "Automatic Calculation of Load-Flows"; AIEE Trans. (PAS); Vol. 76; Oct. 1957; pp.817-828.

[Y3] Podmore,R. and Undrill, J.M.; "Modified Nodal Iterative Load-Flow Algorithm to Handle Series Capacitive Branches"; IEEE Trans. on PAS, Vol. PAS-92, No. 4, July/Aug. 1973, pp. 1379-87.

[Y4] Ward, J.B. and Hale, H.W.; "Digital Computer Solution of Power-Flow Problems"; AIEE Trans. (PAS); Vol. 75; June 1956; pp. 398-404.

Z-matrix Algorithms:

[Z1] Brameller, A. and Denmead, J.K.; "Some Improved Methods for Digital Network Analysis"; Proc. IEE; Vol. 109A; Feb. 1962; pp. 109-116.

[Z2] Brown, H.E., et al; "Power Flow Solution by Impedance Matric Iterative Methods"; IEEE Trans. on PAS; Vol. PAS-82; April 1963; pp. 1-10.

[Z3] Brown, H.E., et al; "Z-matrix Algorithms in Load Flow Programs"; IEEE Trans. on PAS; Vol. PAS-87; March 1968; pp. 807-814.

[Z4] Gupta, P.P. and Humphrey Davies, M.W.; "Digital Computers in Power System Analysis"; IEE Proc.; Vol. 108A; Jan. 1961; pp. 383-404.

Newton-Raphson Algorithm:

[N1] Billinton, R., Bollinger, K.E. and Dhar, S.B.; "A New Modified Newton's Method for Load-Flow Analysis"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan., 1972; Paper C72 004-5.

[N2] Braunagel, D.A., Kraft, L.A. and Whsong, J.L.; "Inclusion of DC Converter and Transmission Equations Directly in a Newton-Raphson Power Flow"; IEEE Trans. on PAS: Vol. PAS-95, No. 1; Jan/Feb. 1976, pp. 76-78.

[N3] Britton, J.P.; "Improved Area Interchange Control for Newton's Method Load Flows"; IEEE Trans. on PAS; Vol PAS-88, no 10; Oct. 1969; pp. 1577-1581.

[N4] Britton, J.P.; "Improved Load Flow Performance Through a More General Equation Form"; IEEE Trans. on PAS; Vol. PAS-90, no 1; Jan./Feb. 1971; pp. 109-115.

[N5] Dommel, H.W., Tinney, W.F. and Powell, W.L.; "Further Developments in Newton's Method for Power System Applications"; Predented at the IEEE Winter Power Meeting; New York, N.Y.; Jan. 25-30, 1970; Paper 70CP 161-PWR.

[N6] Gomez, F. and Semlyen, A.; "The Newton-Raphson Load Flow with Complex Variables"; Presented at the IEEE PES Summer Meeting; San Francisco, Calif.; July 9-14, 1972; Paper C72 483-6.

[N7] Good, E.J. and Hirsch, P.M.; "Power Flow Using Pairing"; Presented at the IEEE PES Summer Meeting & Energy Resources Conf.; Anaheim, Cal.; July 14-19, 1974; Paper C74 391-9.

[N8] Nabona, N. and Freris, L.L; "New Programming Approach to the Newton-Raphson Load-Flow"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan.28-Feb. 2, 1973; Paper C73 119-5.

[N9] Peterson, N.M. and Scott Meyer, W.; "Automatic Adjustment of Transformer and Phase-Shifter Taps in the Newton Power Flow"; IEEE Trans. on PAS; Vol. PAS-90, no 1; Jan./Feb., 1971; pp. 103-108.

[N10] Sasson, A.M., Trevino, C. and Aboytes, F.; "Improved Newton's Load Flow Through a Minimization Technique"; IEEE Trans. on PAS; Vol. PAS-90, no 5; Sept.-Oct. 1971; pp. 1974-1981.

[N11] Tinney, W.F. and Hart, C.E.; "Power Flow Solution by Newton's Method"; IEEE Trans. on PAS; Vol. PAS-86, no 11; Nov. 1967 pp. 1449-1460.

[N12] Tinney, W.F. and Powell, W.L.; Notes on Newton-Raphson Method for Solution of AC Power Flow Problem; Bonneville Power Administration; Portland, Oreg.; April 1971.

[N13] Van Ness, J.E.; "Iteration Methods for Digital Load Flow Studies"; AIEE Trans. (PAS); Vol. 78; Aug. 1959;pp.583-588.

[N14] Van Ness, J.E. and Griffin, J.H.; "Elimination Methods for Load-Flow Studies"; AIEE Trana. on PAS; Vol. 80; June 1961; pp. 299-304.

Decoupled Load-Flow Algorithms:

[D1] Bonaparte, J.E. and Maslin, W.W.; "Simplified Load Flow"; 1967 PICA Conference Record; May 1967; pp. 385-394.

[D2] Despotovic, S.T.; "A New Decoupled Load-Flow Method"; IEEE Trans. on PAS, Vol. PAS-93, May/June 1974, pp. 884-891.

[D3] Despotovic, S.T., Babic, B.S. and Mastilovic, V.P.; "A Rapid and Reliable Method for Solving Load Flow Problems"; IEEE Trans. on PAS; Vol. PAS-90, no 1; Jan./Feb., 1971; pp.123-130.

[D4] Jalali-Kushkij H. and Wvong, M.D.; "Fast Decoupled Line-Loss Load-Flow Algorithms"; Submitted to the IEEE Winter Power Meeting, New York, January 1978.

[D5] Masiello, R.D. and Wollenberg, B.F.; "Comments on 'Review of Load-Flow Calculation Methods'"; Proc. IEEE; Vol. 63, no 4; April 1975; pp. 713-715.

[D6] Medicherla, T.K.P. and Sachdev, M.S.; "A Second Order Decoupled Load Flow Technique"; Presented at the IEEE PES Summer Meeting; Portland, Oreg; July 18-23, 1976; Paper A76 491-1.

[D7] Peterson, N.M., Tinney, W.F. and Bree, D.W.; "Iterative Linear AC Power Flow Solution for Fast Approximate Outage Studies"; IEEE Trans. on PAS; Vol. PAS-91; Sept./Oct. 1972; pp. 2048-2056.

[D8] Sasson, A.M., Snyder, W. and Flam, M.; "Comments on 'Review of Load-Flow Calculation Methods'"; Proc. of IEEE; Vol. 63, no 4; April 1975; pp. 712-715.

[D9] Stott, B.; "Decoupled Newton Load Flow"; IEEE Trans. on PAS; Vol. PAS-91; Sept./Oct. 1972; pp. 1955-1959.

[D10] Stott, B.; "Effective Starting Process for Newton-Raphson Load Flows"; Proc. IEE; Vol. 118, no 8; Aug. 1971; pp. 983-987.

[D11] Stott, B. and Alsac, O.; "Fast Decoupled Load Flow"; IEEE Trans. on PAS; Vol. PAS-93; May/June 1974; pp. 859-869.

[D12] Uemura, K.; "Power Flow Solution by a Z-matrix Type Method and its Application to Contingency Evaluation"; PICA Conference Paper; 1971; pp. 386-390.

Diakoptics (Tearing):

[T1] Happ, H.H.; "Diakoptics - The Solution of System Problems by Tearing"; Proc. of the IEEE; Vol. 62, no 7; July 1974; pp. 930-940.

[T2] Happ, H.H.; "Multi-Level Tearing and Application"; IEEE Trans. on PAS, Vol. PAS-92, Mar/April 1973, pp. 725-733.

[T3] Jain, M.K. and Rao, N.D.; "A Piecewise Solution of Z-Bus Load Flow"; Presented at the IEEE PES Summer Meeting & Energy Resources Conf.; Anaheim, Calif.; July 14-19, 1974; Paper C74 348-9.

[T4] Kesavan, H.K, and Bhat, M.V.; "Piecewise Newton-Raphson Method: An Exact Model"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan. 27-Feb. 1, 1974; Paper C74 165-7.

[T5] Sasson, A.M.; "Decomposition Technique Applied to the Nonlinear Programming Load-Flow"; 1969 PICA Conference Papers; pp. 242-248.

[T6] Sasson, A.M. and Brown, H.E.; "Deductions and Clarifications on the Diakoptics Approach"; 1971 PICA Conference Papers; pp. 433-439.

Other Load Flow Algorithms:

[01] Birt, K.A., Graffy, J.J. and McDonald, J.D.; "Three Phase Load Flow Program"; IEEE Trans. on PAS, Vol. PAS-95, No. 1, Jan/Feb. 1976, pp. 59-65.

[02] Borkowska, B.; "Probablistic Load Flow"; IEEE Trans. on PAS, Vol. PAS-93, May/June 1974, pp. 752-759.

[03] Bosarge Jr., W.E., Jordan, J.A. and Murray, W.A.: "A Nonlinear Block SOR-Newton Load-Flow Algorithm"; Presented at the IEEE PES Summer Meeting & EHV/UHV Conference; Vancouver, Canada; July 15-20, 1973; Paper C73 464-5.

[04] Cheann, H.R. and Jalali-Kushki, H.; "A New Formulation of the Load-Flow Problem"; Presented at the IEEE PES Winter Meeting; New York, N.Y.; Jan. 27-Feb. 1, 1974; Paper C74 164-0.

[05] Dopazo, J.F., Klitin, O.A. and Sasson, A.M.; "Stochastic Load Flows"; IEEE Trans. on PAS; Vol. PAS-92, no 2; March-April, 1975; pp. 299-309.

[06] Dusonchet, Y.P., Talukdar, S.N., Sinnott, H.E. and El-Abiad, A.H. "Load Flows Using a Combination of Point Jacobi and Newton's Methods"; IEEE Trans. on PAS; Vol. PAS-90; May/June 1971; pp. 941-949.

[07] Galloway, R.H., Taylor, J., Hogg, W.D. and Scott, M.; "New Approach to Power-System Load-Flow Analysis in a Digital Computer"; Proc. IEE; Vol. 117; Jan. 1970; pp. 165-169.

[08] Jalali-Kushki, H.; A New Formulation of the Load-Flow Problem; MASc Thesis; University of B.C.; Vancouver, Canada; 1973.

[09] Jalali-Kushki, H. and Wvong, M.D.; "A Load-Flow Algorithm Using Approximate Line Losses"; Presented at the IEEE Winter Power Meeting; New York, N.Y.; Jan. 31-Feb. 4, 1977; Paper A77 142-3; IEEE Pub. 77 CH 1190-8-PWR.

[010] Okamura, M., et al; "A New Power Flow Model and Solution Method"; IEEE Trans. on PAS, Vol. PAS-94, May/June 1975, pp. 1042-1050.

[011] Roy, L. and Menon, K.B.; "A Generalized Piecewise Load Flow Method"; Presented at the IEEE PES Summer Meeting & Energy Resources Conf.; Anaheim, Calif.; July 14-19, 1974; Paper C 74 329-9.

[012] Sachdev, M.S. and Medicherla, T.K.P.; "A Second Order Load Flow Technique"; IEEE Trans. on PAS, Vol. PAS-96, No. 1, Jan/Feb. 1977, pp. 189-197.

[013] Semlyen, A. and Pervaiz, S.Z.; "Power Flow Calculation by Complex Waves"; Presented at the IEEE PES Summer Meeting & Energy Resources Conf.; Anaheim, Calif.; July 14-19, 1974; Paper C 74 332-3.

[014] Wvong, M.D. and Jalali-Kushki, H.; "Comparison of the Line-Loss Load-Flow Algorithm with the Newton-Raphson Load-Flow"; 1977 PICA Proceedings; pp. 321-324.

APPENDIX A

USER'S MANUAL FOR LFP

A.1 Introduction

LFP stands for the "Load Flow Package". It refers to a collection of computer programs written for the purpose of comparing the performance of various load-flow algorithms. At the present time, only the Newton-Raphson algorithm, the Fast Decoupled (Newton-Raphson) Load-Flow, and the different versions of the line-loss algorithm described in this thesis, are available. However, the structure of the package is such that any other algorithm can be added to it without difficutly. Also, LFP has fairly extensive monitoring features which make it particularly attractive for research work.

The package consists of three distinct modules, which operate in succession to one another in the following order:

(1) The input module; this module reads the input data, rearranges this data according to an internal format while saving the external bus numbers and other information not required by the load-flow algorithm, provides a listingoof input data if necessary, forms the compact nodal admittance matrix, initializes the proper vectors, sets proper pointers to the load-flow algorithm requested by the user, etc., etc. The output from this module is ready to be processed by the proper load-flow algorithm.

(2) The process module; this module performs the proper iterative process on the data provided by the input module. It also prints the values of variables at the end of various iteration steps if such information is required. The output from this

module is the final values of voltages along with some other information such as the number of iterations and the computation time taken by that particular algorithm.

(3) The output module; this module prints and/or plots the output information.

The same input and output modules are shared by all the algorithms while each algorithm has its own process module.

LFP was written for use on UBC's IBM 370/168 digital computer. Except for two small routines in IBM 370 assembly language, the programs are written in FORTRAN IV. They make use of several MTS features as well as some of the programs provided by UBC Computing Centre. All the programs use single precision arithmetic. Dynamic storage allocation and sparsity programming techniques are employed to ensure efficient use of storage. With these features, LFP is capable of handling very large power systems - systems of the order of few thousand buses and transmission lines.

The main purpose of this Appendix is to explain the data preparation and use of the LFP. When necessary, comments are also made about the programming details and the structure of the package.

A.2 How to Use

The object module is in the file LFP. The prefix & is used to specify the package (especially useful in terminal mode). The program may be used by issuing the following command:

```
$R   LFP   SCARDS = ...   SPRINT = ...   PAR = (Parameter list as)
                                                defined below
```

SCARDS may be assigned to the file or device from which the load-flow data is to be read and SPRINT to the unit on which the output should

appear. In terminal mode, the communication with the user is done via GUSER and SERCOM.

The PAR field contains the information about the load-flow algorithm to be used, the starting point and the data file. This information is provided using the following Keyword parameters. (Only the first letter of each Keyword would suffice):

(1) ALGORITHM = name of the load-flow algorithm to be used. Only one name must be specified and it must be one of those specified in section A.3, or names of other algorithms which may be added to the package at a later date. There is no default for this Keyword. Not specifying it either will terminate the job (batch mode) or will result in an error message followed by a request for that information (terminal mode).

(2) DATA = name of the file or device that contains the laod-flow data. The default for this Keyword is *SOURCE*? The format of the data must be as explained in section A.4.

(3) START = F or N. The default value is F, which indicates a flat-start or zero initial estimates for the losses, depending upon the load-flow algorithm being used. START = N indicates a non-flat starting point. The initial estimates of losses or voltages must be provided at a later point in the program. See section A.5 for more details.

Except for the name of the algorithm, these Keywords are optional. Their order is not important and they may be separated by commas and/or blanks. If a Keyword is specified several times the last value will be used. Note that using SCARDS = ... or DATA = ... Keywords, gives identical results.

If no PAR field is specified, either the job would be terminated (batch mode) or the user is requested to provide a PAR field (terminal mode). The information in this field is printed as part of the heading on every page of the output (unless other heading information is provided; see section A.4). If an error is detected in the PAR field, either the job is terminated (batch) or the user is requested to respecify the PAR field (terminal).

A.3 The Algorithms

At the present time, only the algorithms listed in this section may be specified. Other load-flow algorithms may be programmed and added to LFP, in which case the name of their respective entry points (SUBROUTINE or ENTRY) are used to specify the algorithms. The only restriction (apart from those imposed by the programming language) is that these names should not start with the letter "L". Names which start with L are considered as belonging to the line-loss algorithms. For these algorithms, START = N means that the initial estimates of line losses (not voltages) are specified.

The algorithms that may be specified are one of the following:

NEWTON: The Newton-Raphson algorithm.

STOTT: The Fast Decoupled (Newton-Raphson) Load-Flow.

LOSS1: The basic line-loss algorithm. The algorithms starts by setting the line losses either to zero or to their initial estimates, depending upon whether START = F or N.

FLOSS1: The basic line loss algorithm starting from a flat-start or from another point specified by the user, depending upon whether START = F or N. This is another entry point in LOSS1.

LSS1M: The incremental-change line-loss algorithm. Starts with one cycle of the basic line-loss algorithm in which the losses are initially set to zero or to their estimated values, depending upon whether START = F or N.

FLSS1M: The incremental-change line-loss algorithm, starting either from a flat-start or from another starting point specified by the user. This is another entry point in LSS1M.

DECPL1: The decoupled line-loss algorithm A.

DECPL2: The decoupled line-loss algorithm B. This is another entry point in DECPL1.

Apart from the above algorithms, the following names may also be specified:

TEST: Used only for testing the input module. Prints out the information processed by this module.

FLOWS: This is an entry point in LFP which may be specified if the power flows corresponding to a set of voltages are to be calculated. In that case, the START = N must be specified.

A.4 The Data Preparation

Each data file contains several different kinds of data records. Some records contain information about different elements in the system; some contain the general information controlling the number of iterations, the length and the type of the output, etc; and some contain comments or the heading information which is to be printed on the top of every output page. Different records are distinguished by means of the character in their first column. For example, a record starting with letter B contains the information about one of the system's buses while a record starting with letter H contains the heading information.

Because of this, the input records may be arranged in any arbitrary order. The following is a brief description and format of each data record.

A.4.1 Bus Information Records:

Letter in the first column is "B".

Each record contains the information about one particular bus in the system. This information (and its format, in parenthesis) is as follows:

BUS TYPE: A one digit integer number in column 2; (I1). 1 is used for P-Q buses; 2 is used for P-V buses; 3 is used for the slack buses.

BUS NUMBER: A five digit integer number in columns 3 to 7; (I5). This can be any arbitrary number. Used in communications with the user.

BUS NAME: An eight character alphanumerical string in columns 9 to 16; (A8). This can be any arbitrary combination of eight characters. Used as the bus name in the output printout.

BUS GENERATION: A complex number in columns 19 to 32; (2F7.2). Specifies the total active and reactive power generation of the bus in MW. Positive signs are used for generation.

BUS LOAD: A complex number in columns 33 to 48; (2F8.2). Specifies the total active and reactive bus load in MW. Positive signs are used for load.

GENERATOR Q LIMITS: Two real numbers in columns 49 to 63. These are the lower and upper limits for the reactive power generation of P-V buses in MW. At present, LFP does not use this information.

BASE VOLTAGE: A real number in columns 64 to 70; (F7.2). Specifies the base voltage of the bus in KV.

VOLTAGE MAGNITUDE: A real number in columns 71 to 77; (F7.2). Specifies the desired magnitude of voltage for P-V and slack buses in KV.

A.4.2 Transmission Line Information Records

Letter in the first column is "L".

Each record contains the information about one particular transmission line in the system. This information (and its format, in parenthesis) is as follows:

LINE NUMBER: A six digit integer number in columns 2 to 7; (I6). This is an arbitrary number assigned by the user and used in the printout.

LINE NAME: An eight character alphanumeric string in columns 9 to 16; (A8).

SENDING END BUS NO.: A five digit integer number in columns 17 to 21 indicating the bus number of one end of the transmission line; (I5). This number must also be specified on a bus information record.

RECEIVING END BUS NO.: A five digit integer number in columns 22 to 26 indicating the bus number of the other end of the transmission line; (I5). This number must also be specified on a bus information record.

LINE IMPEDANCE: A complex number in columns 29, 44; (2F8.5). Specifies the per unit value of the line impedance.

SENDING END SUSCEPTANCE: A real number in columns 49 to 55; (F7.5). Specifies the per unit value of shunt admittance at the sending end of the line.

RECEIVING END SUSCEPTANCE: A real number in columns 56 to 62; (F7.5). Specifies the per unit value of shunt admittance at the receiving end of the line.

TRANSFORMER INDICATOR: A logical value in column 64; (L1). A "T" in that column indicates that the line has a transformer as well. In this case, the data file must also contain a transformer record with the same number. Otherwise, an error occurs.

A.4.3 Transformer Records

Letter in the first column is "T".

Each record contains the information about one transformer in the system. Transformer taps must be fixed. If there is a transmission line which also has a transformer, the information about the transmission line must appear on a separate "L" record. (The order of the records is not important). The information on each record is as follows:

TRANSFORMER NUMBER: A six digit integer number in columns 2 to 7; (I6). This number is used to specify the transformer in the output printout. If a transformer is part of a transmission line, this number must be the same as the transmission line number on the "L" record.

TRANSFORMER NAME: An eight digit character string in columns 9 to 16; (A8).

SENDING END NUMBER: A five digit integer number in columns 17 to 21; (I5). Specifies the bus number of the sending end of

the transformer. This number must be specified on a bus information record. If the transformer is part of a transmission line which is specified on a "L" record, this number should be the same on the two records.

RECEIVING END NUMBER: A five digit integer number in columns 22 to 26; (I5). Specifies the bus number of the receiving end of the transformer. The comments made about the above item apply here as well.

TRANSFORMER IMPEDANCE: A complex number in columns 27 to 41; (2F8.5). Specifies the per unit value of transformer impedance.

SENDING END TAP: A real number in columns 43 to 50; (F8.4). Specifies the transformer tap (turn ratio) on the sending end side of the transformer.

RECEIVING END TAP: A real number in columns 51 to 58; (F8.4). Specifies the transformer tap (turn ratio) on the receiving end of the transformer.

A.4.4 The Heading Records

Letter in the first column is "H".

The information in the first 48 columns of this record, with the exception of column 1, are printed at the top of every page as part of the heading information. The remaining positions in the heading will contain date and time. If no such record is given the default heading will be used. If there are several "H" records in a data file, the last one will overwrite the previous ones.

A.4.5 The Comment Records

Letter in the first column is "C".

The information in columns 2 to 80 of these records are printed before the first output page is printed. Up to ten "C" records may be specified for each data file. This data is printed in the same order that it appears in the data file.

A.4.6 The General Control Records

Letter in the first column is "G".

This record may contain the information about the maximum number of iterations permitted, the length and type of output, the desired tolerance (on maximum power mismatches), the "ZERO" of the run (values with absolute values smaller than "ZERO" are assumed zero.), and the MVA base of the system. If no "G" record is specified, or if any of the above information is not specified on the "G" record, the default values would be used. If any of the values is assigned several times, the last assignment would be used.

The information on this record is specified by using several Keywords as follows. Only the first character of each Keyword would be sufficient:

ITERATIONS = maximum number of iterations permitted. This must be a positive integer number. The default value is 100.

LIST = an integer number indicating the length and the type of output to be produced. The following is an explanation of various possibilities:

LIST = 1 indicates that only the first page of the output should be printed. This page contains a summary of the results.

LIST = 2 is used if the fulloutput is to be printed. The output for this case is the output of LIST = 1 plus the complete printout of power flows through various branches. This is the default value.

LIST = 6 or 7 is identical to LIST = 1 or 2 except that, in this case, the input data is printed as well.

LIST = n1, n2, n6 or n7, where n is any positive integer, is exactly identical to LIST = 1, 2, 6 or 7 except that, in this case, the values of desired voltage magnitudes and phase angles and also the values of power mismatches are printed at the end of every n iteration. (See section A.5 for details).

LIST = -1, -2, -6 or -7 is the same as LIST = 1, 2, 6 or 7, respectively except that, in this case, the values of power mismatches and the voltage magnitudes and phase angles of the desired buses (see section A.5 for details), will be plotted as well.

LIST = -n1, -n2, -n6 or -n7 where n is any positive integer, is the same as LIST = -1, -2, -6 or -7, respectively except that, here, the values of power mismatches and the voltage magnitudes and phase angles of the desired buses are also printed at the end of every n iteration. In other words, the effect is that of the two above categories combined.

TOLERANCE = any real-number indicating the desired tolerance for the load-flow study in P.U. The default value is .005 P.U.

ZERO = any real number; specifying the minimum non-zero absolute

value of a number. Numbers with an absolute value less than "ZERO" are assumed zero. The default value is .00005.

BASE MVA: any real number; specifying the MVA base of the system. This value is used in converting the P.U. quantities back into their real values. The default value is 100 MVA.

A.4.7 Data Set Definition Records

Letter in the first column is "D".

This record is used to indicate to LFP that the rest of data records are to be read from the unit specified on this record. The format of this record is a "D" in the first column and, a two digit integer number, indicating the number of a logical unit, in columns 10 and 11. This record is usually used as the last record in a data file. In this way, after the data in the file is read, the control is transferred to GUSER, enabling the user to enter any particular commands (data records) he may have for that run.

A.4.8 The End Record

Letter in the first column "E".

The first letter is the only information used from this record. As soon as an "E" record is encountered, the input module stops reading any other input records. The effect of "E" records is identical to, and can be achieved by, using $END OF FILE.

If any record with an initial other than those mentioned above is encountered, either the job will be terminated (batch mode) or the user will be asked to intervene (terminal mode). A sample data file is listed below. This is the data for Test System 6:

```
G I=10 T=.C1 Z=.00C05 L=2
B3     1 ING230                                                          230.    235.
B2     3 JHT 13      127.                                                13.8    13.8
B1     4 LDR 13      47.C    5.0                                         13.8
B1     5 SCA 13      25.8    5.0                                         13.8
B2     6 PUN 13      24.5            17.8    8.15                        13.8    14.3
B2     7 ASH 13      26.7                                                13.8    13.8
B2     8 GGA 13      72.0                                                13.8    13.8
B1    10 JHT132                                                          132.0
B1    11 DBY132                     64.     6.4                          132.0
B1    12 LDR132     C.0     0.0     0.0     0.0                          132.0
B1    13 SCA132     C.C     0.0     0.0     0.0                          132.
B1    14 GLD132     0.0     0.0     5.0     2.5                          132.
B1    16 TAS132     0.0     0.0     18.0    5.4                          132.0
B1    17 CTI132     C.0     0.0     0.0     0.0                          132.0
B1    19 CBL132     0.0     0.0     10.6    5.3                          132.0
B1    20 PUN132     C.C     0.0     10.0    3.0                          132.0
B1    22 DMR132     C.C     0.0     0.0     0.0                          132.0
B1    23 PAL132     0.0     0.0     107.3   36.3                         132.0
B1    24 GCL132     C.C     0.0     6.9     2.6                          132.0
B1    25 ASH132     0.0     0.0     0.0     0.0                          132.0
B1    26 NOR132     0.0     0.0     9.6     4.8                          132.0
B1    27 HMC132     C.C     0.0     24.0    7.2                          132.0
B1    29 JPT132     C.0     0.0     0.0     0.0                          132.0
B1    30 VIT230     C.0     0.0     0.0     0.0                          230.0
B1    31 VIT 12     C.0     0.0     0.0     0.0                          12.6
B1    33 JPT 60     0.0     0.86    27.8    15.9                         60.0
B1    34 JPT 12     C.C     15.0    0.0     0.0                          12.0
B1    35 GGA132     0.0     0.0     0.0     0.0                          132.0
B1    36 CFT132     0.0     0.0     65.4    20.3                         132.0
B1    37 ARN230     C.0     0.0     0.0     0.0                          230.0
B1    38 ARN132     0.0     0.0     0.0     0.0                          132.0
B1    39 STD132     25.C    93.0    197.5   140.5                        132.0
B1    40 VIT132     C.C     0.0     0.0     0.0                          132.0
L     15 TAS132     14      16      .01575  .C393       .0088   .0088
L     17 DBY132     10      11      .0017   .0052       .C044   .0044
L     19 GLD132     13      14      .0516   .129        .0286   .0286   F
L     20 SCA132     12      13      .0168   .0448       .009    .009    F
L     21 LDR132     10      12      .0017   .CC86       .0084   .0084   F
L     22 CBL132     17      19      .00416  .01228      .0028   .0028   F
L     23 CTI132     10      17      .C0208  .C0614      .0014   .0014   F
L     24 PUN132     17      20      .04102  .12C86      .0266   .0266   F
L     25 DMR132     10      22      .04095  .12270      .1042   .1042   F
L     26 DMR132     20      22      .04C4   .1194       .0262   .0262   F
L     27 ASH132     24      25      .0161   .0427       .0084   .0084   F
L     28 GCL132     23      24      .0176   .0458       .0092   .0092   F
L     29 PAL132     22      23      .01C7   .C326       .0276   .0276   F
L     30 JPT132     22      29      .0254   .C75        .0656   .0656   F
L     31 JPT132     26      29      .0165   .0431       .0086   .0086   F
L     32 JPT132     27      29      .0165   .C431       .0086   .0086   F
L     33 STD132     29      39      .0440   .1299       .0284   .0284   F
L     34 GGA132     29      35      .0416   .1231       .0268   .0268   F
L     35 CFT132     35      36      .0128   .C38        .0082   .0082   F
L     36 STD132     36      39      .0094   .0279       .0062   .0062   F
L     37 VIT132     39      40      .00037  .C0313      .0030   .0030   F
L     39 VIT132     38      40      .02842  .C6428      .8190   .8190   F
L     40 ARN230      1      37      .0026   .01584      .0306   .0306   F
T      1 SCA132      5      13 .01632  .272    1.131   1.0
T      2 LDR132      4      12 .00949  .1583   1.125   1.0
T      3 JHT132      3      10 .C0375  .C6245  1.149   1.0
T      4 PUN132      6      20 .03534  .5893   1.045   1.0
T      5 ASH132      7      25 .02202  .3671   1.098   1.0
T      6 JPT 60     29      33 .01184  .2960   1.028   1.0
T      7 JPT 12     33      34 .00641  .1602   1.0     1.0
T      8 GGA132      8      35 .00393  .0982   1.099   1.0
T     10 VIT132     30      40 .003    .06     1.019   1.0
T     11 VIT 12     30      31 .00334  .0667   .975    1.0
T     12 ARN132     37      38 .00114  .0228   1.032   1.0
CATA SET 15
```

A.5 Comments and Restrictions

(1) If START = N is specified, LFP expects to read the initial estimates of line loss or voltages - depending upon the load-flow algorithm being used - from a file or device. The name of this file or device is read from GUSER. In terminal mode, the user is asked to provide this information while, in batch mode, the program expects to find this information immediately after the data records. (There is one exception to this. See the next comment). The name of this file or device must be found in the first 20 columns of the input record. This file or device must contain the initial values of voltages - for algorithms whose name does not start with "L" - or line losses - for algorithms whose names start with "L". If no name is given, the data is expected to come from GUSER.

The format of the data for entering the values of voltages is different from that for entering the initial values of line losses. In the former case, the initial values of bus voltages must be specified, in the same order as that of the input bus records but skipping over the slack buses, using 2F10.5 format. In other words, no bus numbers are necessary since the order is already fixed. The voltage magnitudes (in P.U.) and the voltage phase angles (in degrees) are specified using the above format.

In the latter case, the sending and receiving end bus numbers are specified, followed by the P.U. values of line losses. The order of the records is not important since the sending and receiving end numbers are specified. The format of each record is 2I5, 2F10.5. Note that the Per Unit values of line-losses have to be specified.

In each case the terminal user is given any necessary information with respect to the points mentioned here.

(2) When the LIST Keyword (on the "G" record) is specified such that printing and/or plotting of variables at the end of various iteration steps is desired, the program expects to read the bus numbers whose voltage magnitudes and phase angles are wanted. The terminal user is asked for this information at the proper place. The batch user is expected to enter this information (from *MSOURCE*) immediately after the "E" record. Note that, if START = N is specified at the same time, the information with respect to the initial estimates must follow the information mentioned here.

The format of the data is 20I4. Unspecified bus numbers are ignored. However, if a bus number is entered, and later the negative value of the same number is encountered, the information with respect to that bus is <u>not</u> printed/plotted. In other words, the negative bus numbers may be used to cancel the positive numbers previously entered. If the information about all the voltages (except the slack buses) is required, the word ALL can be entered on the first three columns of the record. Otherwise, the bus numbers are read, using 20I4 format, until a zero bus number is encountered at which point the information is considered to be complete. The power mismatches will always be printed, along with the information about the desired voltages, at the end of the respective iterations. Therefore, if only the power mismatches are wanted, a blank record <u>must</u> follow the "E" record, to indicate that there are no bus numbers specified.

(3) The following routines were used from UBC Computing Centre's library. They must be available at the time of the run:

Character Manipulation Routines: MOVEC, SETC, FINC, FINDST, IGC, DTB, EQUC.

Input/Output and Dynamic Storage Allocation Routines: SERCOM, GUSER, SETPFX, GUINFO, CUINFO, FTNCMD, EMPTYF, CREPLY, GSPACE, FSPACE, CALLER.

Bit Manipulation Routines: SHFTL, SHFTR, LOR, LXOR.

Plotting (Metric): PLCTRL, SYMBOL, NUMBER, PLOT, AXCTRL, SCALE, AXPLOT.

Other Routines: CDATE, PAR, LDINFO, TIME.

Apart from the above routines, many of the MTS features are also used. The program has to be updated according to the changes that may occur to these features. A listing of the program follows.

```
      SUBROUTINE PARR
      LCGICAL*1 IST(255),BATCH,HDNGS(80),MTSU(30),NAME(16),ALG(8)
      COMMCN /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMCN /ADS/ IA,ID,IS,ALG,BATCH
      INTECER*2 LEN
      INTEGER START,FINISH
      LNUM=1
      CALL MCVEC(14,'ASSIGN SCARCS=',MTSU)
      CALL SETC(80,HDNGS,' ')
      CALL MOVEC(5,'ALG.=',HDNGS)
      CALL MOVEC(5,'CATA=',HDNGS(14))
      CALL MOVEC(6,'START=',HDNGS(39))
      CALL CDATE(HDNGS(49))
10    CALL PAR(IST,NI,255,&100)
      GC TO 300
100   IF(BATCH) GC TC 150
      LEN=6
      CALL SERCOM(' PAR=?',LEN,0)
110   CALL SETPFX('?',1)
      CALL GUSER(IST,LEN,0,LNUM,&150)
      CALL SETPFX('&',1)
      NI=LEN
      GO TO 300
150   LEN=19
160   CALL SERCOM(' NO PAR FIELD GIVEN',LEN,0)
      CALL EXIT
300   START=1
      IA=0
      ID=0
      IS=0
350   CALL FINDC(IST,NI,'ADS',3,START,FINISH,ICF,&400,&400)
      GO TC 500
400   IF(IA .NE. 0) GC TC 5000
      LEN=27
      CALL SERCOM(' ALGORITHM DOES NOT DEFAULT',LEN,C)
450   IF(BATCH) CALL EXIT
      LEN=24
      CALL SERCOM(' RESPECIFY THE PAR FIELD',LEN,0)
      GC TC 110
500   START=FINISH+1
      CALL FINDC(IST,NI,'=',1,START,FINISH,ICE,&600,&600)
      GO TO 700
600   LEN=19
      CALL SERCOM(' ERROR IN PAR FIELD',LEN,0)
      GC TO 450
700   START=FINISH+1
      GC TC (1000,2000,3000),ICF
1000  CALL IGC(IST,NI,' ',1,START,FINISH,&400,&400)
      START=FINISH
      CALL FINDC(IST,NI,', ',2,START,FINISH,ICF,&1200)
      IF(START .EQ. FINISH) GO TC 350
      GC TC 1300
1200  FINISH=NI+1
1300  IA=FINISH-START
      IF(IA .GT. 6) IA=6
      CALL SETC(8,ALG,' ')
      CALL MOVEC(IA,IST(START),ALG)
      CALL MOVEC(8,ALG,HDNGS(6))
      START=FINISH+1
```

```
      GO TO 350
C     FIRST IGNORE ALL THE PRECEDING BLANKS:
2000  CALL IGC(IST,NI,' ',1,START,FINISH,&400,&400)
      START=FINISH
      CALL FINDC(IST,NI,', ',2,START,FINISH,ICF,&2100,&400)
C     CHECK TO SEE IF "DATA= , " HAS HAPPENED:
      IF(FINISH .EQ. START) GO TO 350
      GO TO 2200
2100  FINISH=NI+1
2200  NUMB=FINISH-START
      ID=NUMB
      CALL MOVEC(NUMB,IST(START),HDNGS(19))
      CALL MOVEC(NUMB,IST(START),MTSU(15))
      NOMTS=NUMB+14
      LL=19+NUMB
      NUMB=20-NUMB
      CALL SETC(NUMB,HDNGS(LL),' ')
      CALL FTNCMD(MTSU,NOMTS)
      START=FINISH+1
      GO TO 350
C     FIRST IGNORE ALL THE PRECEDING BLANKS:
3000  CALL IGC(IST,NI,' ',1,START,FINISH,&3100,&3100)
      START=FINISH
      CALL FINDC(IST,NI,',FN',3,START,FINISH,IS,&3100,&3100)
      START=FINISH+1
      IF (IS .EQ. 1) GO TO 3050
      CALL FINDC(IST,NI,' ,',2,START,FINISH,ICF,&3020,&3020)
      START=FINISH+1
      GO TO 3050
3020  START=NI+1
3050  IF(IS .LT. 3) GO TO 3200
      CALL MOVEC(4,'N   ',HDNGS(45))
      IS=2
      GO TO 350
3100  START=NI+1
3200  IS=1
      GO TO 350
5000  IF(ID .NE. 0) GO TO 5500
      CALL GTNAME('SCARDS  ',NAME,L)
      CALL MOVEC(L,NAME,HDNGS(19))
      LL=19+L
      L=20-L
      CALL SETC(L,HDNGS(LL),' ')
5500  IF(IS .EQ. 0) IS=1
      IF(IS .EQ. 1) CALL MOVEC(4,'F   ',HDNGS(45))
      RETURN
      END
      SUBROUTINE INPUT(COMNT,J2)
      INTEGER FINISH,DSET,START
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      INTEGER*2 TYPE,NO,FROM,TO,LEN,BUSES(20)
      LOGICAL*1 CODE,CODES(8)/'B','L','T','H','C','E','G','D'/
      LOGICAL*1 HDNGS(80),COMNT(1),ERROR(80),TX,BATCH,ALG(8)
      LOGICAL EQUC
      REAL*8 NAME
      EQUIVALENCE (BUSES(1),ERROR(1))
      REAL*4 BASE,SUSE,SURE,PTAP,QTAP,U
      COMPLEX GEN,LD,Z
```

```
      BMVA=100.
      ICD=-10
      LIST=2
      ITMAX=100
      TOLERN=0.005
      ZERO=.00005
      I1=0
      I2=0
      I3=0
      LINE=0
      J2=0
      DSET=5
      CALL FTNCMD('ASSIGN 1=-BUSINF',16)
      CALL EMPTYF(1)
      CALL FTNCMD('ASSIGN 2=-LININF',16)
      CALL EMPTYF(2)
      CALL FTNCMD('ASSIGN 3=-TXINF',15)
      CALL EMPTYF(3)
      CALL FTNCMD('ASSIGN 4=-SCRATCH',17)
      CALL EMPTYF(4)
50    CONTINUE
      IF(ICD .LT. 0) GO TO 60
      DSET=ICD
      ICD=-10
      CALL SETPFX('&',1)
60    READ(DSET,61,END=850)CODE
61    FORMAT(A1)
      BACKSPACE DSET
      CALL FINDST(CODES,8,CODE,1,1,FINISH,&1000)
      GO TO(100,200,300,400,500,800,600,700),FINISH
C     BUS DATA CARDS
100   I1=I1+1
      READ(DSET,101)TYPE,NO,NAME,GEN,LD,BASE,U
      WRITE(1)TYPE,NO,NAME,GEN,LD,BASE,U
101   FORMAT(1X,I1,I5,1X,A8,2X,2F7.2,2F8.2,15X,2F7.2)
      GO TO 50
C     TRANSMISSION LINE DATA CARD:
200   I2=I2+1
      READ(DSET,201)NO,NAME,FROM,TO,Z,SUSE,SURE,TX
      WRITE(2)       NO,NAME,FROM,TO,Z,SUSE,SURE,TX
201   FORMAT(1X,I6,1X,A8,2I5,2X,2F8.5,4X,2F7.5,1X,L1)
      IF(.NOT. TX)LINE=LINE+1
      GO TO 50
C     TRANSFORMER DATA CARD:
300   I3=I3+1
      READ(DSET,301)NO,NAME,FROM,TO,Z,PTAP,QTAP
      WRITE(3)       NO,NAME,FROM,TO,Z,PTAP,QTAP
301   FORMAT(1X,I6,1X,A8,2I5,2F8.5,2F8.4,2F7.5)
      GO TO 50
C     HEADING CARD, (WON'T WRITE OVER TIME AND DATE):
400   READ(DSET,401)(HDNGS(I),I=1,48)
401   FORMAT(1X,48A1)
      GO TO 50
C     COMMENT CARD:
500   IF(J2 .EQ. 79)GO TO 50
      J1=J2+1
      J2=J2+79
      READ(DSET,501)(COMNT(J),J=J1,J2)
501   FORMAT(1X,79A1)
      GO TO 50
```

```
C     GENERAL INFORMATION RECORD:
600   READ(DSET,601)ERROR
601   FORMAT(80A1)
      LEN2=80
      START=2
605   CALL FINDC(ERROR,LEN2,'ITZLB',5,START,FINISH,ICF,&50,&50)
      START=FINISH+1
      CALL FINDC(ERROR,LEN2,'=',1,START,FINISH,ICE,&605,&605)
      START=FINISH+1
      LIMIT=LEN2-START+1
      GO TO (610,620,620,610,620),ICF
610   CALL DTB(ERROR(START),INT,LIMIT,NSD,' ',&615)
      IF(ICF .EQ. 1) GO TO 612
      LIST=INT
      GO TO 615
612   ITMAX=INT
615   START=START+LIMIT+1
      GO TO 605
620   CALL DTB(ERROR(START),INT1,LIMIT,NSD,' ',&622)
      START=START+LIMIT
622   IF(EQUC(ERROR(START),'.'))GO TO 625
      START=START+1
      GO TO 605
625   LIMIT=LEN2-START+1
      CALL DTB(ERROR(START),INT2,LIMIT,NSD,'.',&630)
      START=START+LIMIT
      LIMIT=LIMIT-1
      IF(INT1 .EQ. 0 .AND. INT2 .EQ. 0) GO TO 605
      A1=INT1
      A2=INT2
      A=A1+A2*10.**(-LIMIT)
626   GO TO (627,627,628,628,629),ICF
627   TOLERN=A
      GO TO 605
628   ZERO=A
      GO TO 605
629   BMVA=A
      GO TO 605
630   START=START+LIMIT+1
      IF(INT1 .EQ. 0) GO TO 605
      A=INT1
      GO TO 626
700   READ(DSET,701)NEWD
701   FORMAT(9X,I2)
      IF(IOD .GT. 0)IOD=NEWD
      DSET=NEWD
      IF(DSET .EQ. 15) CALL FTNCMD('ASSIGN SCARDS=*SOURCE*',22)
      GO TO 50
800   READ(DSET,61)CODE
850   LINE=LINE+13
      IF(LIST .GT. 10 .OR. LIST .LT. 0) GO TO 900
      RETURN
900   IF(.NOT. BATCH)WRITE(16,901)
901   FORMAT(' ENTER THE DESIRED BUS NUMBERS IN 20I4 FORMAT  OR ''ALL'''
     */,' (A ZERO BUS NUMBER INDICATES END OF SERIES)')
      READ(15,61)CODE
      IF(EQUC(CODE,'A')) GO TO 950
      BACKSPACE 15
910   READ(15,911)BUSES
911   FORMAT(20I4)
```

```
      WRITE(1) BUSES
      IF(BUSES(20) .EQ. 0) RETURN
      GO TO 910
950   BUSES(1)=-999
      WRITE(1)BUSES
      RETURN
1000  READ(DSET,1001)ERROR
1001  FORMAT(80A1)
      LEN=41
      CALL SERCOM(' INVALID INITIAL ON THE FOLLOWING RECORD:',LEN,0)
      LEN=80
      CALL SERCOM(ERROR,LEN,0)
      IF(BATCH) GO TO 1500
      LEN=54
      CALL SERCOM(' ENTER Y TO REPLACE N TO TERMINATE OR RETURN TO IGNOR
     *E',LEN,0)
      CALL SETPFX('?',1)
      READ(16,61)CODE
      CALL SETPFX('&',1)
      IF(EQUC(CODE,' ')) GO TO 1100
      IF(EQUC(CODE,'Y')) GO TO 1200
      WRITE(16,1501)
      CALL EXIT
1100  LEN=15
      CALL SETPFX('&',1)
      CALL SERCOM(' RECORD IGNORED',LEN,0)
      GO TO 50
1200  IOD=DSET
      DSET=15
      GO TO 60
1500  CALL SETPFX('&',1)
      WRITE(6,1501)
1501  FORMAT(' JOB TERMINATED BY ERROR')
      STOP
      END
      SUBROUTINE HEAD
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 HDNGS(80)
      NPAGE=NPAGE+1
      WRITE(6,1)HDNGS,NPAGE
1     FORMAT(1H1,5X,80A1,15X,'PAGE',I3/110('*')/)
      NLINE=3
      RETURN
      END
      SUBROUTINE BLSHDR
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 HDNGS(80)
      CALL HEAD
      WRITE(6,101)
101   FORMAT(/' B U S  I N F O R M A T I O N :'/1X,30('+')//5X,'BUS #',3
     *X,'BUS NAME',3X,'TYPE',12X,'GENERATION  ',10X,'LOAD',12X,'BASE',6X
     *,'VOLTAGE'/40X,'MW',6X,'MVAR',8X,'MW',6X,'MVAR',2(7X,'KV',2X)/)
      NLINE=NLINE+6
      RETURN
      END
      SUBROUTINE LHDR
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 HDNGS(80)
      CALL HEAD
      WRITE(6,1)
```

```
1     FORMAT(/' L I N E  I N F O R M A T I O N :')
200   WRITE(6,201)
201   FORMAT(1X,32('+'),//,5X,'LINE #', 5X,'LINE NAME',9X,'S.E.',3X,'R.E
     *.',10X,'PER UNIT R',4X,'PER UNIT X',10X,'S.E. SUSEP.',3X,'R.E. SUS
     *EP.',6X,'TX?',/)
      NLINE=NLINE+5
      RETURN
      END
      SUBROUTINE TXHDR
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 HDNGS(80)
      CALL HEAD
      WRITE(6,1)
1     FORMAT(/' T R A N S F O R M E R  I N F O R M T I O N :')
200   WRITE(6,201)
201   FORMAT(1X,44('+'),//,5X,'TX #',8X,'TX NAME',12X,'S.E.',5X,'R.E.',
     *10X,'LEAKAGE IMPEDANCE',10X,'S.E. TAP',7X,'R.E. TAP'/)
      NLINE=NLINE+5
      RETURN
      END
      SUBROUTINE BUSINF(SCHED,V,U,ANG,NO,TYPE,INPRT)
      LOGICAL*1 BATCH,HDNGS(80),INPRT,MTSU(30),ALG(8)
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      LOGICAL EQUC
      REAL*8 NAME
      INTEGER*2 NO(1),BUSES(20),TYPE(1)
      REAL*4 U(1),ANG(1),BASE
      COMPLEX SCHED(1),V(1),GEN,LD
      EQUIVALENCE (NBUS,I1)
      EQUIVALENCE (MTSU(13),NRTRN)
      DATA NRTST/Z4C404040/
      REWIND 1
      IF(INPRT) CALL BUSHDR
      IEND=NBUS
      I=1
50    READ(1)TYPE(I),NO(I),NAME,GEN,LD,BASE,U(I)
      IF(.NOT. INPRT) GO TO 75
      IF(NLINE .GT. 50) CALL  BUSHDR
      WRITE(6,51)NO(I),NAME,TYPE(I),GEN,LD,BASE,U(I)
      NLINE=NLINE+1
51    FORMAT(3X,I5,6X,A8,4X,I1,8X,2F8.2,2X,2(2X,F8.2),4X,F6.2,5X,F6.2)
75    IF(TYPE(I) .EQ. 3)GO TO 100
      U(I)=U(I)/BASE
      SCHED(I)=(GEN-LD)/BMVA
      ANG(I)=0.
      IF(I .EQ. IEND) GO TO 200
      I=I+1
      GO TO 50
100   TYPE(IEND)=TYPE(I)
      NO(IEND)=NO(I)
      U(IEND)=U(I)/BASE
      V(IEND)=U(IEND)
      ANG(IEND)=0.
      IEND=IEND-1
      IF(IEND .LT. I) GO TO 200
      GO TO 50
200   DO 220 I=1,IEND
      IF(TYPE(I) .EQ. 1)U(I)=1.
```

```
      V(I)=U(I)
220   CONTINUE
225   READ(1,END=250)BUSES
      IF(BUSES(1) .LT. 0) GO TO 243
      DO 240 I=1,20
      IF(BUSES(I) .EQ. 0) GO TO 250
      DO 230 J=1,NBUS
      IF(NO(J) .EQ. BUSES(I)) GO TO 235
230   CONTINUE
      GO TO 240
235   NO(J)=-NO(J)
240   CONTINUE
      GO TO 225
243   DO 246 I=1,NBUS
246   NO(I)=-NO(I)
250   IF(IS .EQ. 1 .OR. EQUC(ALG(1),'L')) RETURN
      IF(BATCH) GO TO 260
      WRITE(16,251)
251   FORMAT(' ENTER NAME OF THE UNIT CONTAINING STARTING '/' VALUE DATA
     * (RETURN OR EOF IF *MSOURCE*)')
      CALL SETPFX('?',1)
260   READ(15,261,END=270)(MTSU(I),I=13,30)
261   FORMAT(20A1)
C     GO TO 270 IF 'RETURN' OR 'EOF'
      IF(NRTRN .EQ. NRTST) GO TO 270
C     ASSIGN LOGICAL UNIT 8 TO FD:
      CALL MOVEC(12,'ASSIGN    8=',MTSU)
      CALL FTNCMD(MTSU,30)
      GO TO 280
270   CALL FTNCMD('EQUATE 8=GUSER',14)
      IF(BATCH) GO TO 280
      WRITE(16,271)
271   FORMAT(' ENTER ''MAG'',''ANG'' IN 2F10.5 FORMAT'/
     *' ORDER THE SAME AS BUS DATA. SKIP OVER SLACK BUSES.')
C     VOLTAGES ARE READ IN P.U., ANGLES IN DEGREES:
C     THE ORDER IS THE SAME AS BUS DATA RECORDS EXCEPT THAT NO DATA
C     IS GIVEN FOR SLACK BUS(ES).
280   DO 290 I=1,IEND
      READ(8,281,END=300)U(I),ANG(I)
281   FORMAT(2F10.5)
      ANG(I)=ANG(I)*3.14156/180.
      A1=U(I)*COS(ANG(I))
      A2=U(I)*SIN(ANG(I))
      V(I)=CMPLX(A1,A2)
290   CONTINUE
300   CALL SETPFX('&',1)
      RETURN
      END
      SUBROUTINE LININF(INPRT,Y,CPONT,NXTR,RPONT,NXTC,NO,TXZ,PTAP,QTA
     *P,TXNO,PSIDE,QSIDE)
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 TX,HDNGS(80),INPRT
      INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1)
      INTEGER*2 LTNO,FROM,TO,NO(1),TXNO(1),PSIDE(1),QSIDE(1)
      COMPLEX Y(1),Z,TXZ(1),YP1,YP2,YB,ZA,ZC,Z1,Y2,Y3
      REAL*4 PTAP(1),FILLER(4),QTAP(1)
      REAL*8 NAME
      INTEGER GP,VP,PP
      EQUIVALENCE(I1,NBUS)
```

```
      REWIND 2
      REWIND 3
C     TO SET POINTERS FOR DIAGONAL ELEMENTS:
      K=1
      DO 20 I=1,NBUS
      Y(2*I)=0.
      CPONT(K)=I
      RPONT(K)=I
      NXTR(K)=K
      NXTC(K)=K
      K=K+8
20    CONTINUE
C     TO SET POINTERS FOR OFF-DIAGONAL ELEMENTS:
      GP=K
      LAST=GP+ LINE*8-16
      IF(GP .GT. LAST) GO TO 35
      DO 30 I=GP,LAST,8
      NXTR(I)=I+8
30    CONTINUE
35    NXTR(LAST+8)=0
      IF(I3 .EQ. 0)GO TO 60
      IF(INPRT) CALL TXHDR
      DO 50 I=1,I3
      READ(3)TXNO(I),NAME,PSIDE(I),QSIDE(I),TXZ(I),PTAP(I),QTAP(I)
      IF(.NOT. INPRT) GO TO 50
      IF(NLINE .LE. 50)GO TO 40
      CALL TXHDR
40    WRITE(6,41)TXNO(I),NAME,PSIDE(I),QSIDE(I),TXZ(I),PTAP(I),QTAP(I)
      NLINE=NLINE+1
41    FORMAT(3X,I5,10X,A8,4X,2(5X,I4),7X,2(3X,F8.4),1X,2(8X,F7.2))
50    CONTINUE
60    IF(INPRT) CALL LHDR
      DO 700 J=1,I2
      READ(2)LTNO,NAME,FROM,TO,Z,SESU,RESU,TX
      IF(.NOT. INPRT) GO TO 75
      IF(NLINE .LE. 50)GO TO 70
      CALL LHDR
70    WRITE(6,71)LTNO,NAME,FROM,TO,Z,SESU,RESU,TX
71    FORMAT(I9,7X,A8,6X,2(3X,I4),4X,2(6X,F9.4),7X,2(7X,F7.5),8X,L1)
      NLINE=NLINE+1
75    YP1=CMPLX(0.,SESU)
      YP2=CMPLX(0.,RESU)
      IF(.NOT. TX)GO TO 300
      DO 150 I=1,I3
      IF(TXNO(I) .EQ. LTNO)GO TO 160
150   CONTINUE
160   TXNO(I)=0
C     TRANSFORMERS ARE ALWAYS ASSUMED TO BE AT S.E.
      FROM=PSIDE(I)
      TO=QSIDE(I)
      RATIO=QTAP(I)/PTAP(I)
      IF(CABS(TXZ(I)) .NE. 0.)GO TO 200
      Z=1./Z
      ZA=YP1+Z
      YB=RATIO*RATIO*(YP2+Z)
      Z=-RATIO*Z
      YP2=ZA+Z
      YP1=YB+Z
      GO TO 350
200   ZA=Z
```

```
      ZC=TXZ(I)/RATIO
      YB=(1.-RATIO)/TXZ(I)+YP1
      Z1=ZA+ZC+ZA*ZC*YB
      Y3=YB*ZA/Z1
      Y2=YB*ZC/Z1
      YP2=YP2+Y2
      YP1=RATIO*(RATIO-1.)/TXZ(I)  +Y3
      Z=-1./Z1
      GO TO 350
300   Z=-1./Z
350   CONTINUE
      WRITE(4)LTNO,NAME,FROM,TO,YP1,YP2,Z
      CALL SET(FROM,TO,Z,YP1,YP2,NO,Y,CPCNT,RPONT,NXTR,NXTC,GP)
700   CONTINUE
      IF(I3 .EQ. 0)GO TO 1000
      REWIND 3
      DO 800 I=1,I3
      READ(3)LTNO,NAME,FROM,TO,FILLER
      IF(TXNO(I) .EQ. 0)GO TO 800
      RATIO=QTAP(I)/PTAP(I)
      IF(CABS(TXZ(I)) .EQ. 0.)TXZ(I)=CMPLX(0.,ZERO)
      YP2=1./TXZ(I)
      Z=-YP2*RATIO
      YP2=YP2+Z
      YP1=-RATIO*YP2
      WRITE(4)LTNO,NAME,FROM,TO,YP1,YP2,Z
      CALL SET(FROM,TO,Z,YP1,YP2 ,NO,Y,CPCNT,RPONT,NXTR,NXTC,GP)
800   CONTINUE
1000  RETURN
      END
      SUBROUTINE SET(FROM,TO,SY,YP1,YP2,NO,Y,CPCNT,RPONT,NXTR,NXTC,GP)
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      INTEGER GP,VP,PP
      INTEGER*2 FROM,TO,NO(1),CPCNT(1),RPCNT(1),NXTR(1),NXTC(1)
      COMPLEX SY,YP1,YP2,Y(1)
      EQUIVALENCE(I1,NBUS)
C     SY IS THE NEGATIVE OF SELF ADMITTANCE
C     YP1 AND YP2 ARE THE LOCAL ADMITTANCES AT SENDING AND RECEIVING ENDS.
C     TO AND FROM ARE THE EXTERNAL BUS NUMBERS OF SENDING AND RECEIVING ENDS.
C     FIRST FIND THE INTERNAL NUMBERS CORRESPONDING TO "TO" AND "FROM".:
      DO 100 I=1,NBUS
      NOI=NO(I)
      K=IABS(NOI)
      IF(K .EQ. TO) TO=-I
      IF(K .EQ. FROM) FROM=-I
100   CONTINUE
      TO=-TO
      FROM=-FROM
C     CHANGE THE SELF ADMITTANCES OF BOTH ENDS:
      KK=FROM*2
      Y(KK)=Y(KK)-SY+YP1
      JJ=TO*2
      Y(JJ)=Y(JJ)-SY+YP2
C     SET THE MUTUAL ADMITTANCE ELEMENT:
C     IF THE ELEMENT EXISTS, MODIFY IT:
      CALL SCAN(Y,CPCNT,NXTR,FROM,TO,VP,PP)
      IF(VP .EQ. 0)GO TO 600
      Y(VP)=Y(VP)+SY
      GO TO 700
C     IF THE ELEMENT IS NEW, INSERT IT:
```

```
600   CALL INSRT(Y,CPCNT,RPCNT,NXTR,NXTC,GP,FROM,TO,SY,&700)
700   RETURN
      END
      SUBROUTINE SCAN(Y,CPCNT,NXTR,FROM,TO,VP,PP)
      INTEGER*2 CPCNT(1),NXTR(1),FROM,TO,TEMP
      INTEGER FIRST,VP,PP
      COMPLEX Y(1)
C     ONLY UPPER HALF OF THE ADMITTANCE MATRIX IS FORMED.
C     THEREFORE, ELEMENT (ROW,COL) EXISTS IF (ROW .LE. COL).
      IF(FROM-TO)100,200,300
100   VP=0
      PP=0
      FIRST=FROM*8-7
      NEXT=NXTR(FIRST)
120   IF(NEXT.EQ.FIRST)GO TO 150
      IF(CPCNT(NEXT) .EQ. TO) GO TO 130
      NEXT=NXTR(NEXT)
      GO TO 120
130   PP=NEXT
      VP=(NEXT+7)/4
150   RETURN
200   VP=FROM*2
      PP=VP*4-7
      RETURN
300   TEMP=FROM
      FROM=TO
      TO=TEMP
      GO TO 100
      END
      SUBROUTINE INSRT(Y,CPCNT,RPCNT,NXTR,NXTC,GP,FROM,TO,VALUE,*,*)
      INTEGER VP,PP,GP
      COMPLEX VALUE ,Y(1)
      INTEGER*2 CPCNT(1),RPCNT(1),NXTR(1),NXTC(1),FROM,TO
C     ERROR RETURN 1 = NO INSERTION REQUIRED
C     ERROR RETURN 2 = THERE IS NO SPACE IN GARBAGE C
C     GARBAGE POINTER(GP) IS THE (PP) OF THE SPECIFIED LOCATION.
      IF(CABS(VALUE) .EQ. 0.)RETURN 1
      IF(GP .EQ. 0)RETURN 2
      LL=NXTR(GP)
      PP=FROM*8-7
      L=NXTR(PP)
      NXTR(PP)=GP
      NXTR(GP)=L
      RPCNT(GP)=FROM
      PP=TO*8-7
      L=NXTC(PP)
      NXTC(PP)=GP
      NXTC(GP)=L
      CPCNT(GP)=TO
      VP=(GP+7)/4
      Y(VP)=VALUE
      GP=LL
      RETURN
      END
      EXTERNAL BUSINF,LININF,PROSS,FLOWS,LNFST
      INTEGER ASCHEC,AV,AU,AANG,ANO,ATYPE,AY,AJQ,FT
      INTEGER ECHO1,ECHO2/1/,ATZ,OUT(20)
      LOGICAL*1 COMNT(790),HDNGS(80),BATCH,INPRT,ALG(8)
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
```

```
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      LOGICAL EQUC
      COMMON /LOSSES/ SL
      EQUIVALENCE(NBUS,I1)
C     STORE ECHO STATUS OF USER
      CALL GUINFO('ECHOOFF ',ECHO1)
C     THEN SET ECHO=OFF:
      CALL CUINFO('ECHOOFF ',ECHO2)
      NPAGE=0
      ISTNO=0
      BATCH=.TRUE.
      CALL CREPLY(&10)
      BATCH=.FALSE.
10    CALL SETPFX('&',1)
      CALL FTNCMD('EQUATE 5=SCARDS',15)
      CALL FTNCMD('EQUATE 6=SPRINT',15)
      CALL FTNCMD('EQUATE 15=GUSER',15)
      CALL FTNCMD('EQUATE 16=SERCOM',16)
      CALL PARR
      CALL INPUT(COMNT,J2)
      IF(J2 .EQ. 0)GO TO 100
      CALL HEAD
      WRITE(6,1)(COMNT(J),J=1,J2)
1     FORMAT(/' C O M M E N T S :'/1X,17('+')//(/5X,'*',79A1))
100   CALL GSPACE(ASCHED,NBUS*28)
      AV=ASCHED+NBUS*8
      AU=AV+NBUS*8
      AANG=AU+NBUS*4
      ANO=AANG+NBUS*4
      ATYPE=ANO+NBUS*2
      INPRT=IABS(MOD(LIST,10)) .GT. 5
      CALL CALLER(BUSINF,ASCHED,AV,AU,AANG,ANO,ATYPE,IPTR(INPRT))
C     GET SPACE FOR COMPACT ADMITTANCE MATRIX AND TRANSFORMER DATA:
      N1=(NBUS+ LINE)*16
      CALL GSPACE(AY,N1)
      IF(I3 .EQ. 0)GO TO 110
      CALL GSPACE(ATZ,I3*22)
C     READ LINE INFORMATION AND FORM Y MATRIX
      CALL CALLER(LININF,IPTR(INPRT),AY,AY,AY+2,AY+4,AY+6,ANO,ATZ,ATZ+I3
     **8,ATZ+I3*12,ATZ+I3*16,ATZ+I3*18,ATZ+I3*20)
      CALL  FSPACE(ATZ)
      GO TO 120
110   CALL CALLER(LININF,IPTR(INPRT),AY,AY,AY+2,AY+4,AY+6,ANO)
120   CONTINUE
      IF(.NOT. EQUC(ALG(1),'L')) GO TO 150
      CALL GSPACE(SL,NBUS*8)
      IF(IS .EQ. 1) GO TO 150
      CALL CALLER(LNFST,ANO,SL)
150   CALL LDINFO(1,ALG,BITS,OUT)
      CALL GSPACE(ASEQ,NBUS*2)
      CALL CALLER(PROSS,IPTR(OUT(3)),AY,AY,AY+2,AY+4,AY+6,ANO,ATYPE,
     *ASCHED,AV,AU,AANG,ASEQ,&1111)
      CALL FSPACE(ASEQ)
      LIST=IABS(LIST)
      IF(MOD(LIST,5) .EQ. 1) GO TO 200
      CALL CALLER(FLOWS,AV,AU,AANG,ANO)
200   CALL FSPACE(ASCHED)
C     RESTORE USER'S ECHO STATUS
      CALL CUINFO('ECHOOFF ',ECHO1)
      WRITE(6,201)
```

```
660     201   FORMAT(1H1)
661           STOP
662     1111  WRITE(6,1112)ISTNO
663     1112  FORMAT(//' ***ERROR RETURN '//5X,'ISTNO=',I7)
664           GO TO 200
665           END
666           SUBROUTINE PROSS(ALG,Y,CPONT,NXTR,RPONT,NXTC,NO,TYPE,SCHED,V,U,
667          *ANG,SEQ,*)
668           EXTERNAL NEWTON
669           COMPLEX Y(1),SCHED(1),V(1)
670           INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1),NO(1),TYPE(1)
671           INTEGER*2 SEQ(1)
672           REAL U(1),ANG(1)
673           CALL ALG(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,NXTC,&1)
674           RETURN
675     1     RETURN 1
676           END
677           SUBROUTINE TEST(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,NXTC)
678           INTEGER*2 SEQ(1)
679           COMPLEX Y(1),SCHED(1),V(1)
680           INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1),NO(1),TYPE(1)
681           REAL U(1),ANG(1)
682           COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
683           COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
684           COMMON /ADS/ IA,ID,IS,ALG,BATCH
685           LOGICAL*1 HDNGS(80),BATCH,ALG(8)
686           EQUIVALENCE(NBUS,I1)
687           N1=(NBUS+LINE)*2
688           WRITE(6,1)
689     1     FORMAT('1'//20X,'A D M I T T A N C E   M A T R I X:'//7X,'VALUE',
690          *7X,'CPONT',5X,'NXTR',5X,'RPONT',5X,'NXTC'//)
691           K=1
692           I=2
693     10    WRITE(6,2)Y(I),CPONT(K),NXTR(K),RPONT(K),NXTC(K)
694     2     FORMAT(2F10.4,4I10)
695           I=I+2
696           K=K+8
697           IF(I .GT. N1)GO TO 100
698           GO TO 10
699     100   WRITE(6,101)
700     101   FORMAT('1'//20X,'I N T E R N A L  B U S  A R R A G E M E N T :',
701          */5X,'NO',5X,'TYPE',10X,'SCHED',10X,'V',5X,'U',5X,'ANG'//)
702           WRITE(6,102)(NO(I),TYPE(I),SCHED(I),V(I),U(I),ANG(I),I=1,NBUS)
703     102   FORMAT(3X,2I5,6F10.4)
704           WRITE(6,103)
705     103   FORMAT('1'//20X,'C O M M O N  A R E A S:'//5X,'/ININF/:'/)
706           WRITE(6,104)I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
707     104   FORMAT(4I7,3F10.4,2I7)
708           WRITE(6,105)NPAGE,NLINE,HDNGS
709     105   FORMAT(//5X,'/OUTINF/:'/2I10//80A1)
710           WRITE(6,106)IA,ID,IS,ALG,BATCH
711     106   FORMAT(//5X,'/ADS/:'/3I10,5X,8A1,5X,L1)
712           CALL EXIT
713           RETURN
714           END
715           SUBROUTINE ORDER(CONN,N1,CPONT,NXTR,RPONT,NXTC,NO,TEMP,NPQ,NPV
716          *,TYPE)
717           COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
718           INTEGER SHFTR,FIRST,PP,COL,SHFTL
719           INTEGER CONN(N1,1),ONE/Z80C0C000/
```

```
      INTEGER*2 TEMP(1),NO(1),TYPE(1)
      INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1)
C     AT RETURN TIME "NO" IS CHANGED TO INDICATE THE OPTIMAL
C     ORDERING OF THE NODES. ALSO NO(IEND+1)=0
      NPQ=0
      NPV=0
      JJ=1
C     TO FORM CONNECTION MATRIX:
C     ROWS OF "Y" ARE IN COLUMNS OF "CONN"
      DO 100 I=1,IEND
      DO 10 J=1,N1
10    CONN(J,I)=0
C     D I A G O N A L  E L E M E N T :
      TEMP(I)=1
C     J IS WORD INDICATOR; K IS SHFTR INDICATOR:
      J=(I-1)/32+1
      K=I-J*32+31
      L=SHFTR(ONE,K)
      CONN(J,I)=L
      FIRST=I*8-7
      PP=NXTR(FIRST)
20    IF(PP .EQ. FIRST)GO TO 50
      COL=CPONT(PP)
      IF(COL .GT. IEND)GO TO 30
      TEMP(I)=TEMP(I)+1
      J=(COL-1)/32+1
      K=COL-J*32+31
      L=SHFTR(ONE,K)
      CONN(J,I)=LOR(CONN(J,I),L)
30    PP=NXTR(PP)
      GO TO 20
50    PP=NXTC(FIRST)
70    IF(PP .EQ. FIRST)GO TO 100
      COL=RPONT(PP)
      IF(COL .GT. IEND)GO TO 80
      TEMP(I)=TEMP(I)+1
      J=(COL-1)/32+1
      K=COL-J*32+31
      L=SHFTR(ONE,K)
      CONN(J,I)=LOR(CONN(J,I),L)
80    PP=NXTC(PP)
      GO TO 70
100   CONTINUE
C     TO FIND MINIMUM COUNT OF ROWS NOT YET PROCESSED
C     (AT PROCESSED ROWS TEMP=0):
150   MIN=IEND+1
      INDEX=0
      DO 200 I=1,IEND
      IF(TEMP(I) .EQ. 0)GO TO 200
      IF(TEMP(I) .GE. MIN)GO TO 200
      MIN=TEMP(I)
      INDEX=I
200   CONTINUE
      IF(INDEX .EQ. 0)GO TO 400
      NO(JJ)=INDEX
      IF(TYPE(INDEX) .EQ. 1) GO TO 210
      NPV=NPV+MIN
      GO TO 220
210   NPQ=NPQ+MIN
220   IF(JJ .EQ. IEND)GO TO 400
```

```
      JJ=JJ+1
      TEMP(INDEX)=0
C     TO CHANGE THE OTHER ROWS & FIND NEW COUNTS:
      DO 300 I=1,IEND
      IF(TEMP(I) .EQ. 0)GO TO 300
      J=1
      K=INDEX-1
230   IF(K .LE. 31)GO TO 240
      K=K-32
      J=J+1
      GO TO 230
C     CHECK TO SEE IF ROW "I" HAS AN LEMENT IN POSITION COL=INDEX
240   L=SHFTL(CONN(J,I),K)
      IF(L .GE. 0)GO TO 300
      DO 250 M=1,N1
250   CONN(M,I)=LOR(CONN(M,I),CONN(M,INDEX))
      L=SHFTR(ONE,K)
      CONN(J,I)=LXOR(CONN(J,I),L)
      TEMP(I)=NAMBER(CONN(1,I),IEND)
300   CONTINUE
      GO TO 150
800   RETURN
      END
      SUBROUTINE NEWTON(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,
     *NXTC,*)
      EXTERNAL ORDER,JOBN,BACK,CHANGE ,DRAW
      COMPLEX Y(1),V(1),SCHED(1)
      INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1),NO(1),TYPE(1)
      INTEGER*2 SEQ(1),NSTEP
      INTEGER ATEMP,ACONN,AWR,RES(2)
      REAL U(1),ANG(1)
      LOGICAL*1 BATCH,ALG(8),HDNGS(80),PLOT,PRINT
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /TYM/ RES
      EQUIVALENCE (NBUS,I1)
      CALL TIME(0)
      PLOT=LIST .LT. 0
      IF(PLOT) REWIND 3
      IF(PLOT) CALL EMPTYF(3)
      PRINT=IABS(LIST) .GT. 10
      NSTEP=IABS(LIST)/10
      N1=(IEND-1)/32+1
      NN=N1*IEND*4
      CALL GSPACE(ACONN,NN)
      CALL GSPACE(ATEMP,IEND*2)
      CALL CALLER(ORDER,ACONN,IPTR(N1),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT
     *),IPTR(NXTC),IPTR(SEQ),ATEMP,IPTR(NPQ),IPTR(NPV),IPTR(TYPE))
      CALL FSPACE(ATEMP)
      CALL FSPACE(ACONN)
      NWORDS=NPQ*5+NPV*3-IEND*2
      CALL GSPACE(AJO,NWORDS*4)
      CALL GSPACE(AWR,NBUS*16)
      CALL GSPACE(AWI,NBUS*2)
      CALL GSPACE(APNTR,NBUS*2)
      IT=0
      LIMIT=0
200   CALL CALLER(JOBN,AWR,AWI,AJO,AJO,IPTR(SEQ),IPTR(Y),IPTR(CPONT),IPT
     *R(NXTR),IPTR(RPONT),IPTR(NXTC),IPTR(V),IPTR(SCHED),IPTR(U),IPTR(TY
```

```
     *PF),APNTR,IPTR(ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3),&400)
      IERR2=NO(IERR2)
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(.NOT. PRINT .OR. IT .NE. LIMIT) GO TO 210
      LIMIT=LIMIT+NSTEP
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
210   IF(ERR2 .LE. TOLERN) GO TO 300
      IF(IT .GE. ITMAX) GO TO 250
      CALL CALLER(BACK,AJQ,AJQ,APNTR,IPTR(SEQ),AWR,AWR+NBUS*4,&400)
      CALL CALLER(CHANGE,AWR,AWR+NBUS*4,IPTR(U),IPTR(ANG),IPTR(V),IPTR(S
     *EQ),IPTR(TYPE))
      IT=IT+1
      GO TO 200
250   ISTNO=250
300   CALL FSPACE(AJQ)
      CALL FSPACE(AWR)
      CALL FSPACE(AWI)
      CALL FSPACE(APNTR)
      CALL TIME(3,C,RES)
      IF(.NOT. PLOT) GO TO 350
      IT=IT+1
      NN=(IEND*2+3)*IT*4
      CALL GSPACE(ARRAY,NN)
      CALL CALLER(DRAW,IPTR(IT),IPTR(NC),ARRAY)
      CALL FSPACE(ARRAY)
      IT=IT-1
350   CALL OUT1(IT,ERR1,ERR2,IERR2,ERR3)
      IF(ISTNO .NE. 0) RETURN 1
      RETURN
400   RETURN 1
      END
      SUBROUTINE PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      LOGICAL*1 ALG(8),BATCH
      LOGICAL EQUC
      REAL U(1),ANG(1)
      LOGICAL*1 HDNGS(80)
      COMPLEX V(1)
      INTEGER*2 NO(1)
      IF(IT .EQ. 0 .OR. NLINE .GE. 40) CALL HEAD
      IF(IT .EQ. 1 .AND. EQUC(ALG(1),'L')) CALL HEAD
      WRITE(6,1)IT
1     FORMAT(//2X,'ITERATION :',I4/2X,15('*')//9X,'BUSNO',15X,'V',15X,'U
     *',15X,'ANG')
      NLINE=NLINE+5
      DO 100 I=1,IEND
      IF(NO(I) .GT. 0) GO TO 100
      K=-NO(I)
      ANGI=ANG(I)*180./3.141592
      WRITE(6,2)K,V(I),U(I),ANGI
2     FORMAT(10X,I4,5X,F8.5,4X,F8.5,3X,F9.4,5X,F10.5)
      NLINE=NLINE+1
100   CONTINUE
      WRITE(6,101)ERR1,ERR2,IERR2,ERR3
101   FORMAT(/5X,'ERR1=',F10.6,5X,'ERR2=',F10.6,5X,'IERR2=',I4,5X,
     *'ERR3=',F10.6)
      NLINE=NLINE+2
```

```
      RETURN
      END
      SUBROUTINE BACK(QR,QI,POINTR,SEQ,DV,DANG,*)
      COMMON /ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 HDNGS(80)
      EQUIVALENCE (NBUS,I1)
      REAL*4 QR(1),DV(1),DANG(1)
      INTEGER QI(1),FROM,TO,COL
      INTEGER*2 POINTR(1),SEQ(1)
      DO 1000 I=1,IEND
      ROW1=0.
      ROW2=0.0
      J=IEND-I+1
      JP1=J+1
      FROM=POINTR(J)
      TO=POINTR(JP1)
      IF(TO .LT. 0)TO=-TO
      IF(FROM .LT. 0)GO TO 500
      INDEX=FROM+3
   50 IF(INDEX .EQ. TO) GO TO 200
      COL=QI(INDEX)
      IF(COL .LT. 0) COL=-COL
      DO 100 K=JP1,IEND
      IF(SEQ(K) .EQ. COL)GO TO 150
  100 CONTINUE
      ISTNO=100
      RETURN 1
  150 COL=QI(INDEX)
      INDEX=INDEX+1
      ROW1=ROW1+QR(INDEX)*DANG(K)
      INDEX=INDEX+1
      ROW2=ROW2+QR(INDEX)*DANG(K)
      INDEX=INDEX+1
      IF(COL .LT. 0)GO TO 50
      ROW1=ROW1+QR(INDEX)*DV(K)
      INDEX=INDEX+1
      ROW2=ROW2+QR(INDEX)*DV(K)
      INDEX=INDEX+1
      GO TO 50
  200 X2=QR(FROM+1)-ROW2
      DV(J)=X2
      ROW1=ROW1+QR(FROM+2)*X2
      DANG(J)=QR(FROM)-ROW1
      GO TO 1000
  500 FROM=-FROM
      INDEX=FROM+1
  550 IF(INDEX .EQ. TO)GO TO 700
      COL=QI(INDEX)
      IF(COL.LT. 0)COL=-COL
      DO 600 K=JP1,IEND
      IF(SEQ(K) .EQ. COL)GO TO 650
  600 CONTINUE
      ISTNO=600
      RETURN 1
  650 COL=QI(INDEX)
      INDEX=INDEX+1
      ROW1=ROW1+QR(INDEX)*DANG(K)
      INDEX=INDEX+1
      IF(COL .LT. 0)GO TO 550
```

```
      ROW1=ROW1+QR(INDEX)*DV(K)
      INDEX=INDEX+1
      GO TO 550
700   DANG(J)=QR(FROM)-ROW1
1000  CONTINUE
      RETURN
      END
      SUBROUTINE CHANGE(DV,DANG,U,ANG,V,SEQ,TYPE)
      COMMON /ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (NBUS,I1)
      REAL*4 DV(1),DANG(1),U(1),ANG(1)
      COMPLEX V(1)
      INTEGER*2 SEQ(1),TYPE(1)
      DO 100 I=1,IEND
      J=SEQ(I)
      ANG(J)=ANG(J)+DANG(I)
      IF(TYPE(J) .EQ. 1)  U(J)=U(J)+DV(I)*U(J)
      A1=U(J)*COS(ANG(J))
      A2=U(J)*SIN(ANG(J))
      V(J)=CMPLX(A1,A2)
100   CONTINUE
      RETURN
      END
      SUBROUTINE WRMKR(Y,V,TYPE,NXT,PONT,WRR,WRI)
      COMMON /PINF/IK,K,FIRST,ROW
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      INTEGER*2 TYPE(1),NXT(1),PONT(1),WRI(1)
      INTEGER FIRST,PP,VP,ROW
      COMPLEX Y(1),V(1),IJK,IK
      REAL WRR(4,1),NKM
      PP=NXT(FIRST)
50    IF(PP .EQ. FIRST)GO TO 300
      J=PONT(PP)
      WRI(K)=J
      IF(TYPE(J) .EQ. 2)WRI(K)=-J
      VP=(PP+7)/4
      IJK=V(J)*Y(VP)
      IK=IK+IJK
      IF(J .GT. IEND)GO TO 200
      IJK=V(ROW)*CONJG(IJK)
      HKM=AIMAG(IJK)
      NKM=REAL(IJK)
      WRR(1,K)=HKM
      WRR(2,K)=-NKM
      WRR(3,K)=NKM
      WRR(4,K)=HKM
      K=K+1
200   PP=NXT(PP)
      GO TO 50
300   RETURN
      END
      SUBROUTINE JGBN(WRR,WRI,QR,QI,SEQ,Y,CPONT,NXTR,RPONT,NXTC,V,
     *SCHED,U,TYPE,POINTR,ERR1,ERR2,IERR2,ERR3,*)
      REAL WRR(4,1),QR(1),U(1)
      INTEGER*2 WRI(1),CPONT(1),NXTR(1),RPONT(1),NXTC(1)
      INTEGER*2 SEQ(1),POINTR(1),TYPE(1)
      COMPLEX Y(1),V(1),IK,SK,DS,TEMP,SCHED(1)
      INTEGER ROW,FIRST,DIAG,QI(1),COL
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
```

```
      LOGICAL*1 TYPE2,TYPE1,HDNGS(80)
      COMMON /PINF/IK,K,FIRST,ROW
      EQUIVALENCE (I1,NBUS)
C     CAUTION: BE VERY CAREFUL WITH THIS PROGRAM.
C     IT WORKS! AND IT HAS TAKEN A LOT OF TIME TO DO SO.
C     SO, IF SOMETHING DOES NOT LOOK RIGHT IN THE FIRST GLANCE
C     DONOT CHANGE IT QUICKLY. READ THE WHOLE PROGRAM AND THINK AGAIN.
C
C     ROUTINE TO FORM TRIANGULARIZED JACOBIAN.
C     WRR IS REAL-PART WORKING ROW: CONTAINS H,J,N,L.
C     WRI IS INTEGER-PART WORKING ROW: CONTAINS COL.
C     NO SHOWS THE ORDER IN WHICH THE NODES ARE TO BE PROCESSED.
C     QI & QR ARE EQUIVALENT.( JACOBIAN MATRIX COMPACT FORM).
C
      DV1=0.
      DV2=0.
      ERR2=0.
      ERR3=0.
      POINTR(1)=1
      DO 1000 I=1,IEND
      ROW=SEQ(I)
      IK=0.0
C     TO FORM WORKING ROW:
      WRI(1)=ROW
      TYPE2=TYPE(ROW) .EQ. 2
      TYPE1=.NOT. TYPE2
      IF(TYPE2)WRI(1)=-ROW
      K=2
      FIRST=ROW*8-7
C    NOTE THE INFORMATION SHARED THROUGH /PINF/
      CALL WRMKR(Y,V,TYPE,NXTO,RPONT,WRR,WRI)
      CALL  WRMKR(Y,V,TYPE,NXTR,CPONT,WRR,WRI)
      TEMP=Y(ROW*2)
      IK=IK+V(ROW)*TEMP
      SK=V(ROW)*CONJG(IK)
      DS=SCHED(ROW)-SK
      DP=REAL(DS)
      DV1=DV1+DP
      ADD=ABS(DP)
      IF(ERR2 .GE. ADD) GO TO 50
      ERR2=ADD
      IERR2=ROW
50    ERR3=ERR3+ADD
      IF(TYPE2) GO TO 70
      DQ=AIMAG(DS)
      DV2=DV2+DQ
      ADD=ABS(DQ)
      IF(ERR2 .GE. ADD) GO TO 60
      ERR2=ADD
      IERR2=ROW
60    ERR3=ERR3+ADD
70    TEMP=TEMP*U(ROW)**2
      WRR(1,1)=-AIMAG(SK+TEMP)
      WRR(2,1)=REAL(SK-TEMP)
      WRR(3,1)=REAL(SK+TEMP)
      WRR(4,1)=AIMAG(SK-TEMP)
      K=K-1
      IF(I .EQ. 1)GO TO 300
C     TO PERFORM TRIANGULARIZATION PROCESS ON THE ROW BEFORE STORING IT:
      J=1
```

```
1080      90    IDONE=SEQ(J)
1081            IF(TYPE(IDONE) .EQ. 2) IDONE=-IDONE
1082            JJ=0
1083      100   JJ=JJ+1
1084            IF(IDONE .EQ. WRI(JJ))GO TO 500
1085      150   IF(JJ .LT. K) GO TO 100
1086            J=J+1
1087            IF(J .LT. I) GO TO 90
1088      C     ROW TRIANGULARIZED. DIVIDE THE ROW BY ITS DIAGONAL ELEMENT
1089      C     AND STORE THE RESULTS:
1090      300   RATIO=1./WRR(1,1)
1091            DO 350 J=1,K
1092            IF(WRI(J) .EQ. 0)GO TO 350
1093            WRR(1,J)=WRR(1,J)*RATIO
1094            IF(WRI(J) .GT. 0)WRR(3,J)=WRR(3,J)*RATIO
1095      350   CONTINUE
1096            DP=DP*RATIO
1097            IF(TYPE2)GO TO 400
1098            RATIO=-WRR(2,1)
1099            WRR(4,1)=WRR(4,1)+WRR(3,1)*RATIO
1100            RATIO2=1./WRR(4,1)
1101            DO 360 J=1,K
1102            IF(WRI(J) .EQ. 0)GO TO 360
1103            WRR(2,J)=(WRR(2,J)-WRR(1,J)*RATIO)*RATIO2
1104            IF(WRI(J) .GT. 0)WRR(4,J)=(WRR(4,J)+WRR(3,J)*RATIO)*RATIO2
1105      360   CONTINUE
1106            DQ=(DQ+DP*RATIO)*RATIO2
1107      C     ROW IS READY TO BE STORED
1108      400   INDEX=POINTR(I)
1109            IF(TYPE2)POINTR(I)=-INDEX
1110            J=2
1111            QR(INDEX)=DP
1112            INDEX=INDEX+1
1113            IF(TYPE2)GO TO 420
1114            QR(INDEX)=DQ
1115            INDEX=INDEX+1
1116            QR(INDEX)=WRR(3,1)
1117            INDEX=INDEX+1
1118      420   IF(J .GT. K) GO TO 460
1119            JJ=WRI(J)
1120            IF(JJ .EQ. 0)GO TO 450
1121            QI(INDEX)=JJ
1122            INDEX=INDEX+1
1123            QR(INDEX)=WRR(1,J)
1124            INDEX=INDEX+1
1125            IF(TYPE2)GO TO 430
1126            QR(INDEX)=WRR(2,J)
1127            INDEX=INDEX+1
1128      430   IF(JJ .LT. 0)GO TO 450
1129            QR(INDEX)=WRR(3,J)
1130            INDEX=INDEX+1
1131            IF(TYPE2)GO TO 450
1132            QR(INDEX)=WRR(4,J)
1133            INDEX=INDEX+1
1134      450   J=J+1
1135            GO TO 420
1136      460   POINTR(I+1)=INDEX
1137            GO TO 1000
1138      C     TO COMBINE A ROW WITH A PREVIOUSLY PROCESSED ROW:
1139      500   INDEX=POINTR(J)
```

```
      IF(INDEX .LT. 0) INDEX=-INDEX
      LAST=POINTR(J+1)
      IF(LAST .LT. 0) LAST=-LAST
      RATIO=-WRR(1,JJ)
C     FIRST COMBINE THE PRESENT ROW WITH ONLY THE FIRST ROW OF THE
C     PREVIOUSLY PROCESED ROW:
      IF(TYPE1)RATIO2=-WRR(2,JJ)
      DP=DP+QR(INDEX)*RATIO
      IF(TYPE1)DQ=DQ+QR(INDEX)*RATIO2
520   INDEX=INDEX+1
      IF(WRI(JJ) .LT. 0)GO TO 540
      INDEX=INDEX+1
      WRR(3,JJ)=WRR(3,JJ)+QR(INDEX)*RATIO
      IF(TYPE1)WRR(4,JJ)=WRR(4,JJ)+QR(INDEX)*RATIO2
      INDEX=INDEX+1
540   COL=QI(INDEX)
      INDEX=INDEX+1
C     SEE IF THE CORRSEPODING OFF DIAGONAL ELEMENT EXISTS:
      DO 550 NA=1,K
      IF(COL .EQ. WRI(NA))GO TO 600
550   CONTINUE
C     IF NOT, CREATE ONE WITH ZERO VALUES:
      K=K+1
      WRI(K)=COL
      DO 560 NB=1,4
560   WRR(NB,K)=0.
      NA=K
C     CHANGE THE VALUE OF THE OFF DIAGONAL ELEMENT (THAT CERTAINLY
C     EXISTS NOW):
600   WRR(1,NA)=WRR(1,NA)+QR(INDEX)*RATIO
      IF(TYPE1)WRR(2,NA)=WRR(2,NA)+QR(INDEX)*RATIO2
      INDEX=INDEX+1
      IF(WRI(JJ) .GT. 0)INDEX=INDEX+1
      IF(COL .LT. 0) GO TO 700
      WRR(3,NA)=WRR(3,NA)+QR(INDEX)*RATIO
      IF(TYPE1)WRR(4,NA)=WRR(4,NA)+QR(INDEX)*RATIO2
      INDEX=INDEX+1
      IF(WRI(JJ) .GT. 0) INDEX=INDEX+1
700   IF(INDEX .LT. LAST) GO TO 540
C     IF THE PREV PROC. ROW IS TYPE1, COMBINE THE SECONE ROW OF THAT:
      IF(WRI(JJ) .LT. 0)GO TO 800
      RATIO=-WRR(3,JJ)
      IF(TYPE1)RATIO2=-WRR(4,JJ)
      INDEX=POINTR(J)
      INDEX=IABS(INDEX)+1
      DP=DP+QR(INDEX)*RATIO
      IF(TYPE1)DQ=DQ+QR(INDEX)*RATIO2
      INDEX=INDEX+2
730   COL=QI(INDEX)
      INDEX=INDEX+2
      DO 740 NA=1,K
      IF(COL .EQ. WRI(NA))GO TO 760
740   CONTINUE
      ISTNO=740
      RETURN 1
760   WRR(1,NA)=WRR(1,NA)+QR(INDEX)*RATIO
      IF(TYPE1)WRR(2,NA)=WRR(2,NA)+QR(INDEX)*RATIO2
      INDEX=INDEX+1
      IF(COL .LT. 0)GO TO 780
      INDEX=INDEX+1
```

```
      WRR(3,NA)=WRR(3,NA)+QR(INDEX)*RATIO
      IF(TYPE(ROW) .EQ. 1)WRR(4,NA)=WRR(4,NA)+QR(INDEX)*RATIO2
      INDEX=INDEX+1
780   IF(INDEX .LT. LAST) GO TO 730
800   WRI(JJ)=0
      GO TO 150
1000  CONTINUE
      ERR1=SQRT(DV1*DV1+DV2*DV2)
      RETURN
      END
      SUBROUTINE OUT1(IT,ERR1,ERR2,IERR2,ERR3)
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /TYM/ RES
      INTEGER RES(2)
      LOGICAL*1 HDNGS(80)
      CALL HEAD
      WRITE(6,1)I1,I2,I3,LINE
1     FORMAT(/' T A B L E  O F  P A R A M E T E R S :'/1X,38('+'),4(/),5
     *X,'NO. OF BUSES=',I4,18X,'NO. OF TRANSMISSION LINES=',I5///5X,'NO.
     * OF TRANSFORMERS=',I5,10X,'TOTAL NUMBER OF BRANCHES=',I5//)
      WRITE(6,2)BMVA,LIST
2     FORMAT(5X,'BASE MVA SPECIFIED=',F7.2,10X,'LIST=',I4//)
      WRITE(6,3)IT,ITMAX,ERR1,TOLERN,ERR2,IERR2,ERR3
3     FORMAT(5X,'NO. OF ITERATIONS=',I6///5X,'MAX NO. OF ITERATIONS=',I6
     *///5X,'TOTAL POWER MISMATCHES=',F10.6///5X,'TOLERANCE SPECIFIED=',
     *F10.6///5X,'MAX POWER MISMATCHES=',F10.6,5X,'AT BUS ',I4///5X,'TOT
     *AL ABS POWER MISMATCHES=',F10.6)
      WRITE(6,4)RES
4     FORMAT(//5X,'CPU TIME USED IN PROSS=',I6///,5X,'ELAPSED TIME IN PR
     *OSS=',I6///)
      IF(LIST .LT.0) CALL PLCTNO
      RETURN
      END
      SUBROUTINE FLOWS(V,U,ANG,NO)
      EXTERNAL OUTPUT
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      EQUIVALENCE (I1,NBUS)
      LOGICAL*1 HDNGS(80)
      COMPLEX V(1)
      REAL U(1),ANG(1)
      INTEGER*2 NO(1)
      INTEGER FT
      CALL GSPACE(LTNO,LINE*2)
      CALL GSPACE(NAME,NBUS*18)
      CALL GSPACE(FT,LINE*4)
      CALL GSPACE(IMP,LINE*24)
      CALL GSPACE(BASE,NBUS*4)
      CALL CALLER(OUTPUT,IPTR(V),IPTR(U),IPTR(ANG),NAME,IPTR(NO),FT,FT+L
     *INE*2,IMP,IMP+LINE*8,IMP+LINE*16,BASE,LTNO)
      CALL FSPACE(NAME)
      CALL FSPACE(FT)
      CALL FSPACE(IMP)
      CALL FSPACE(BASE)
      CALL FSPACE(LTNO)
      RETURN
      END
      SUBROUTINE OUTPUT(V,U,ANG,NAME,NO,FROM,TO,YP1,YP2,Z,BASE,LTNO)
      COMMON/ININF/I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
```

```
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /ADS/IA,ID,IS,ALG,BATCH
      EQUIVALENCE (I1,NBUS)
      LOGICAL*1 HDNGS(80),ALG(8),BATCH
      LOGICAL*1 TEST(8)/'F','L','O','W','S',' ',' ',' '/
      COMPLEX V(1),Z(1),YP1(1),YP2(1),S,ST,YPI
      REAL U(1),ANG(1),FILLER(4),BASE(1)
      INTEGER*2 NO(1),TYPE,FROM(1),TO(1),LTNO(1)
      REAL*8 NAME(1),LTNAME
      INTEGER ITEST(2),IALG(2)
      EQUIVALENCE (ALG(1),IALG(1)),(TEST(1),ITEST(1))
      REWIND 1
      REWIND 4
      IEND=NBUS
      PI=3.14159
      I=1
  100 READ(1)TYPE,NO(I),NAME(I),FILLER,BASE(I),VOLT
      IF(TYPE .NE. 3) GO TO 200
      NO(IEND)=NO(I)
      NAME(IEND)=NAME(I)
      BASE(IEND)=BASE(I)
      IEND=IEND-1
      I=I-1
  200 I=I+1
      IF(I .LE. IEND) GO TO 100
      DO 300 I=1,LINE
      READ(4)LTNO(I),LTNAME,FROM(I),TO(I),YP1(I),YP2(I),Z(I)
  300 Z(I)=-Z(I)
      CALL HEAD
      DO 1000 I=1,NBUS
      K=NO(I)
      IC=NLINE+6
      DO 400 J=1,LINE
      IF(FROM(J) .EQ. K .OR. TO(J) .EQ. K) IC=IC+1
  400 CONTINUE
      IF(IC .GT. 60) CALL HEAD
      ST=0.
      VOLT=U(I)*BASE(I)
      DEG=ANG(I)*180./PI
      WRITE(6,401)K,NAME(I),U(I),VOLT,DEG
  401 FORMAT(//10X,'BUS NO',I5,2X,A8,4X,F5.3,' PU',4X,F8.3,' KV',4X,F8.3
     *,' DEG'/30X,'TO:')
      NLINE=NLINE+4
      DO 700 J=1,LINE
      IF(K .EQ. FROM(J))GO TO 500
      IF(K .NE. TO(J)) GO TO 700
      KK=FROM(J)
      YPI=YP2(J)
      GO TO 550
  500 KK=TO(J)
      YPI=YP1(J)
  550 DO 600 II=1,NBUS
      IF(KK .EQ. NO(II)) GO TO 650
  600 CONTINUE
  650 S=V(I)*(Z(J)+YPI)-V(II)*Z(J)
      S=V(I)*CONJG(S)*BMVA
      ST=ST+S
      WRITE(6,651)KK,NAME(II),S,LTNO(J)
  651 FORMAT(35X,I5,2X,A8,2(10X,F10.3),4X,'(#',I4,')')
      NLINE=NLINE+1
```

```
 7C0    CONTINUE
        WRITE(6,701)ST
 701    FORMAT(38X,52('-')/35X,' T O T A L S :',3X,2(10X,F10.3))
        NLINE=NLINE+2
 1C00   CONTINUE
        IF(IALG(1) .EQ. ITEST(1) .AND. IALG(2) .EQ. ITEST(2)) STOP
        RETURN
        END
        SUBROUTINE LOSS1(V,U,ANG,NC,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,NXTC
       *,*)
        EXTERNAL CRDER,MOCEF,FORMBP,FORBKW,LCHNGE,DRAW,ERSLC
        COMPLEX Y(1),V(1),SCHED(1)
        INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1),NO(1),TYPE(1)
        INTEGER*2 SEQ(1),NSTEP
        INTEGER RES(2),ACONN,ATEMP
        REAL U(1),ANG(1)
        LOGICAL*1 BATCH,ALG(8),HDNGS(80),PLOT,PRINT
        COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
        COMMON /ADS/ IA,IC,IS,ALG,BATCH
        COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
        COMMON /TYM/ RES
        COMMON /LOSSES/ SL
        EQUIVALENCE (I1,NBUS)
        IA=1
        GO TO 10
        ENTRY FLOSS1(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,NXTC,*)
        CALL GSPACE(SL,NBUS*8)
        IA=2
 1C     CALL TIME(0)
        PLOT=LIST .LT. 0
        IF(PLOT) REWIND 3
        IF(PLOT) CALL EMPTYF(3)
        PRINT=IABS(LIST) .GT. 10
        NSTEP=IABS(LIST)/10
        N1=(IEND-1)/32+1
        NN=N1*IEND*4
        CALL GSPACE(ACONN,NN)
        CALL GSPACE(ATEMP,IEND*2)
        CALL CALLER(CRDER,ACONN,IPTR(N1),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT
       *),IPTR(NXTC),IPTR(SEQ),ATEMP,IPTR(NPQ),IPTR(NPV),IPTR(TYPE))
        CALL FSPACE(ATEMP)
        CALL FSPACE(ACONN)
        NWORDS=NPQ*5+NPV*3-IEND*2
        CALL GSPACE(AM,NWORDS*4)
        CALL GSPACE(AWR,NBUS*16)
        CALL GSPACE(AWI,NBUS*2)
        CALL GSPACE(APNTR,NBUS*2)
        CALL GSPACE(BV,IEND*8)
        CALL GSPACE(NUMB,IEND*2)
        CALL CALLER(MOCEF,AWR,AWI,AM,AM,BV,IPTR(SEQ),IPTR(Y),IPTR(CPONT),I
       *PTR(NXTR),IPTR(RPONT),IPTR(NXTC),IPTR(SCHED),IPTR(U),IPTR(TYPE),
       *APNTR,NUMB,&400)
        CALL FSPACE(AWR)
        CALL FSPACE(AWI)
        CALL GSPACE(BP,IEND*8)
        IT=0
        IF(IA .EQ. 1) GO TO 50
        CALL CALLER(ERSLC, BP,NUMB,IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(U),IP
       *TR(ANG),IPTR(SCHED),IPTR(IT),SL,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPT
       *R(RPONT),IPTR(NXTC),IPTR(ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3))
```

```
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF(.NOT. PRINT) GO TO 20
      LIMIT=NSTEP
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
20    IF(ERR2 .LE. TOLERN) GO TO 300
      GO TO 100
50    LIMIT=1
100   CALL CALLER(FORMBP,BV,BP,IPTR(SEQ),SL,IPTR(IT),IPTR(V),IPTR(ANG),
     *IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPTR(RPCNT),IPTR(NXTC),IPTR(TYPE),
     *IPTR(U))
      CALL CALLER(FORBKW,AM,AM,BP,APNTR,NUMB,IPTR(SEQ))
      CALL CALLER(LCHNGE,BP,NUMB,IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(U),IP
     *TR(ANG),IPTR(SCHED),IPTR(IT),SL,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPT
     *R(RPCNT),IPTR(NXTC),IPTR(ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3))
      IT=IT+1
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF( .NOT. PRINT .OR. IT .NE. LIMIT) GO TO 200
      LIMIT=LIMIT+NSTEP
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
200   IF(ERR2 .LE. TOLERN) GO TO 300
      IF(IT .GE. ITMAX) GO TO 250
      GO TO 100
250   ISTNO=250
300   CALL FSPACE(AM)
      CALL FSPACE(APNTR)
      CALL FSPACE(BV)
      CALL FSPACE(BP)
      CALL FSPACE(NUMB)
      CALL TIME(3,C,RES)
      IF(.NOT. PLOT) GO TO 350
      IF(IA .EQ. 2) IT=IT+1
      NN=(IEND*2+3)*IT*4
      CALL GSPACE(ARRAY,NN)
      CALL CALLER(DRAW,IPTR(IT),IPTR(NO),ARRAY)
      CALL FSPACE(ARRAY)
      IF(IA .EQ. 2) IT=IT-1
350   CALL OUT1(IT,ERR1,ERR2,IERR2,ERR3)
      IF(ISTNO .NE. 0) RETURN 1
      RETURN
400   RETURN 1
      END
      SUBROUTINE LCHNGE(BP,NUMB,SEQ,TYPE,V,U,ANG,SCHED,IT,SL,Y,CPONT,NXT
     *R,RPONT,NXTC,ERR1,ERR2,IERR2,ERR3)
      REAL BP(1),U(1),ANG(1)
      INTEGER*2 NUMB(1),CPONT(1),NXTR(1),RPONT(1),NXTC(1),SEQ(1),TYPE(1)
      COMPLEX Y(1),V(1),SL(1),SCHED(1),IK,DV,SLK,DS
      INTEGER FIRST,PP,VP
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      DO 100 I=1,IEND
      K=SEQ(I)
      KB=NUMB(K)
      ANG(K)=-BP(KB)
      IF(TYPE(K) .EQ. 1)U(K)=SQRT(BP(KB+1)*2.)
      A1=U(K)*COS(ANG(K))
      A2=U(K)*SIN(ANG(K))
```

```
      V(K)=CMPLX(A1,A2)
100   CONTINUE
      ENTRY ERSLC(BP,NUMB,SEQ,TYPE,V,U,ANG,SCHED,IT,SL,Y,CPONT,NXTR,
     *RPONT,NXTC,ERR1,ERR2,IERR2,ERR3)
      DV1=0.
      DV2=0.
      ERR2=0.
      ERR3=0.
      DO 200 I=1,IEND
      IK=0.
      SLK=0.
      FIRST=I*8-7
      PP=NXTR(FIRST)
110   IF(PP .EQ. FIRST) GO TO 120
      VP=(PP+7)/4
      J=CPCNT(PP)
      IK=IK+Y(VP)*V(J)
      DV=V(I)-V(J)
      SLK=SLK-DV*CONJG(DV*Y(VP))
      PP=NXTR(PP)
      GO TO 110
120   PP=NXTC(FIRST)
130   IF(PP .EQ. FIRST) GO TO 150
      VP=(PP+7)/4
      J=RPCNT(PP)
      IK=IK+Y(VP)*V(J)
      DV=V(I)-V(J)
      SLK=SLK-DV*CONJG(DV*Y(VP))
      PP=NXTC(PP)
      GO TO 130
150   SL(I)=SLK/2.
      IK=IK+Y(I*2)*V(I)
      IK=V(I)*CONJG(IK)
      DS=SCHED(I)-IK
      DP=REAL(DS)
      DV1=DV1+DP
      ADD=ABS(DP)
      IF(ERR2 .GE. ADD) GO TO 160
      ERR2=ADD
      IERR2=I
160   ERR3=ERR3+ADD
      IF(TYPE(I) .EQ. 2) GO TO 200
      DQ=AIMAG(DS)
      DV2=DV2+DQ
      ADD=ABS(DQ)
      IF(ERR2 .GE. ADD) GO TO 170
      ERR2=ADD
      IERR2=I
170   ERR3=ERR3+ADD
200   CONTINUE
      ERR1=SQRT(DV1**2+DV2**2)
      RETURN
      END
      SUBROUTINE LWRMKR(WRR,WRI,Y,PONT,NXT,U,FIRST,K,TYPE,NB,B,SUM)
      COMMON /INIAF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      COMPLEX Y(1),SUM
      INTEGER*2 PONT(1),NXT(1),WRI(1),TYPE(1)
      REAL WRR(4,1),U(1),B(1)
      INTEGER VP,PP,FIRST
```

```
      PP=NXT(FIRST)
50    IF(PP .EQ. FIRST) GO TO 200
      J=PCNT(PP)
      WRI(K)=J
      IF(TYPE(J) .EQ. 2)WRI(K)=-J
      VP=(PP+7)/4
      SUM=SUM+Y(VP)
      GIK=REAL(Y(VP))
      BIK=AIMAG(Y(VP))
      IF(J .GT. IEND) GO TO 110
      WRR(1,K)=BIK
      WRR(2,K)=GIK
      IF(TYPE(J) .EQ. 2) GO TO 100
      WRR(3,K)=GIK
      WRR(4,K)=-BIK
      K=K+1
      GO TO 150
100   K=K+1
110   U1=U(J)**2/2.
      B(NB)=B(NB)-GIK*U1
      I=(FIRST+7)/8
      IF(TYPE(I) .EQ. 2) GO TO 150
      B(NB+1)=B(NB+1)+BIK*U1
150   PP=NXT(PP)
      GO TO 50
200   RETURN
      END
      SUBROUTINE LNFST(NO,SL)
      INTEGER*2 NO(1)
      COMPLEX SL(1),TEMP
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      LOGICAL*1 BATCH,ALG(8),MTSU(30)
      DATA NRTST/Z4C404040/
      EQUIVALENCE (MTSU(13),NRTRN)
      IF(BATCH) GO TO 100
      WRITE(16,1)
1     FORMAT(' ENTER NAME OF THE UNIT CONTAINING STARTING'/
     *' VALUES  (RETURN OR EOF IF *MSOURCE*)')
      CALL SETPFX('?',1)
100   READ(15,101,END=200)(MTSU(I),I=13,30)
101   FORMAT(20A1)
      IF(NRTRN .EQ. NRTST) GO TO 200
      CALL MOVEC(12,'ASSIGN    8=',MTSU)
      CALL FTNCMD(MTSU,30)
      GO TO 210
200   CALL FTNCMD('EQLATE 8=GUSER',14)
      IF(BATCH) GO TO 210
      WRITE(16,201)
201   FORMAT(' ENTER P.U. VALUES OF ''NO1'',''NO2'',''SL'''/
     *' IN (2I5,2F10.5) FORMAT')
210   DO 220 I=1,NBUS
220   SL(I)=0.
250   DO 290 I=1,LINE
      READ(8,251,END=300)NO1,NO2,TEMP
251   FORMAT(2I5,2F10.5)
      DO 280 J=1,NBUS
      K=NO(J)
      IF(K .LT. 0) K=-K
```

```
      IF(K .EQ. NO1) NO1=-J
      IF(K .EQ. NO2) NO2=-J
      IF(NC1 .LT. 0 .AND. NO2 .LT. 0) GO TO 285
280   CONTINUE
285   NO1=-NO1
      NO2=-NO2
      SL(NO1)=SL(NC1)+TEMP/2.
      SL(NC2)=SL(NC2)+TEMP/2.
290   CONTINUE
300   CALL SETPFX('&',1)
      RETURN
      END
      SUBROUTINE FORBKW(AM,MA,BP,POINTR,NUMB,SEQ)
      REAL AM(1),BP(1)
      INTEGER MA(1)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      INTEGER*2 POINTR(1),NUMB(1),SEQ(1)
      DO 500 I=1,IEND
      K=SEQ(I)
      NB=NUMB(K)
      J=POINTR(I)
      LAST=POINTR(I+1)
      IF(LAST .LT. 0) LAST=-LAST
      IF(J .LT. 0) J=-J
      IF(POINTR(I) .LT. 0) GO TO 200
      BP(NB+1)=BP(NB+1)-BP(NB)*AM(J+2)
100   J=J+3
      IF(J .EQ. LAST) GO TO 300
      K=IABS(MA(J))
      KB=NUMB(K)
      BP(KB)=BP(KB)-BP(NB)*AM(J+1)-BP(NB+1)*AM(J+2)
      IF(MA(J) .LT. 0) GO TO 100
      KB=KB+1
      J=J+2
      BP(KB)=BP(KB)-BP(NB)*AM(J+1)-BP(NB+1)*AM(J+2)
      GO TO 100
200   J=J+1
      IF(J .EQ. LAST) GO TO 300
      K=IABS(MA(J))
      KB=NUMB(K)
      J=J+1
      BP(KB)=BP(KB)-BP(NB)*AM(J)
      IF(MA(J-1) .LT. 0) GO TO 200
      KB=KB+1
      J=J+1
      BP(KB)=BP(KB)-BP(NB)*AM(J)
      GO TO 200
300   J=POINTR(I)
      IF(J .LT. 0) J=-J
      BP(NB)=BP(NB)*AM(J)
      IF(POINTR(I) .GT. 0) BP(NB+1)=BP(NB+1)*AM(J+1)
500   CONTINUE
C     BACK SUBSTITUTION STARTS:
      DO 1000 II=1,IEND
      I=IEND-II+1
      J=POINTR(I)
      IF(J .LT. 0) J=-J
      LAST=POINTR(I+1)
      IF(LAST .LT. 0) LAST=-LAST
```

```
      SUM1=0.
      IF(POINTR(I) .LT. 0) GO TO 700
      SUM2=0.
600   J=J+3
      IF(J .EQ. LAST) GO TO 800
      K=IABS(MA(J))
      KB=NUMB(K)
      SUM1=SUM1+AM(J+1)*BP(KB)
      SUM2=SUM2+AM(J+2)*BP(KB)
      IF(MA(J) .LT. 0) GO TO 600
      J=J+2
      KB=KB+1
      SUM1=SUM1+AM(J+1)*BP(KB)
      SUM2=SUM2+AM(J+2)*BP(KB)
      GO TO 600
700   J=J+1
      IF(J .EQ. LAST) GO TO 800
      K=IABS(MA(J))
      KB=NUMB(K)
      J=J+1
      SUM1=SUM1+AM(J)*BP(KB)
      IF(MA(J-1) .LT. 0) GO TO 700
      J=J+1
      KB=KB+1
      SUM1=SUM1+AM(J)*EP(KB)
      GO TO 700
800   J=POINTR(I)
      IF(J .LT. 0) J=-J
      K=SEQ(I)
      NB=NUMB(K)
      IF(POINTR(I) .LT. 0) GO TO 900
      BP(NB+1)=BP(NB+1)-SUM2
      SUM1=SUM1+BP(NB+1)*AM(J+2)
900   BP(NB)=BP(NB)-SUM1
1000  CONTINUE
      RETURN
      END
      SUBROUTINE FORMBP(B,BP,SEQ,SL,IT,V,ANG,Y,CPONT,NXTR,RPONT,NXTC
     *,TYPE,U)
      INTEGER FIRST,PP,VP
      REAL B(1),BP(1),ANG(1),U(1)
      INTEGER*2 SEQ(1),CPONT(1),NXTR(1),RPONT(1),NXTC(1),TYPE(1)
      COMPLEX SL(1),V(1),Y(1),SUM
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      LOGICAL*1 ALG(8),BATCH
      EQUIVALENCE (I1,NBUS)
      IF(IA .EQ. 2) GO TO 100
      IF(IS .EQ. 2 .OR. IT .NE. 0) GO TO 100
      DO 50 I=1,IEND
50    SL(I)=0.
100   NB=0
      DO 200 I=1,IEND
      K=SEQ(I)
      NB=NB+1
      BP(NB)=B(NB)-REAL(SL(K))
      IF(TYPE(K) .EQ. 2) GO TO 200
      NB=NB+1
      BP(NB)=B(NB)-AIMAG(SL(K))
200   CONTINUE
```

```
      IF(IT .EQ. 0 .AND. (IA+IS) .NE. 4) GO TO 450
C     NOTE THAT IT WILL NOT BRANCH IF IA=IS=2.
      NB=0
      DO 400 I=1,IEND
      NB=NB+1
      K=SEQ(I)
      SUM=0.
      FIRST=K*8-7
      PP=NXTR(FIRST)
C      NOTE THAT -EPS (ACCORDING TO THE DEFINITION) IS BEING FORMED
250   IF(PP .EQ. FIRST) GO TO 300
      J=CPCNT(PP)
      VP=(PP+7)/4
      DANG=ANG(K)-ANG(J)
      EPS=U(K)*U(J)*SIN(DANG)-DANG
      SUM=SUM+EPS*CONJG(Y(VP))
      PP=NXTR(PP)
      GO TO 250
300   PP=NXTC(FIRST)
320   IF(PP .EQ. FIRST) GO TO 350
      J=RPCNT(PP)
      VP=(PP+7)/4
      DANG=ANG(K)-ANG(J)
      EPS=U(K)*U(J)*SIN(DANG)-DANG
      SUM=SUM+EPS*CONJG(Y(VP))
      PP=NXTC(PP)
      GO TO 320
350   BP(NB)=BP(NB)+AIMAG(SUM)
      IF(TYPE(K) .EQ. 2) GO TO 400
      NB=NB+1
      BP(NB)=BP(NB)-REAL(SUM)
400   CONTINUE
450   CONTINUE
      RETURN
      END
      SUBROUTINE MOCEF(WRR,WRI,AM,MA,B,SEQ,Y,CPONT,NXTR,RPONT,NXTC,SCHED
     *,L,TYPE,POINTR,NUMB,*)
      REAL WRR(4,1),AM(1),U(1),B(1)
      INTEGER*2 WRI(1),SEQ(1),CPONT(1),RPONT(1),NXTR(1),NXTC(1)
      INTEGER*2 TYPE(1),POINTR(1),NUMB(1)
      INTEGER MA(1),ROW,PP,VP,FIRST,COL
      COMPLEX Y(1),SCHED(1),SUM,TEMP
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 HDNGS(80),PV,PQ
      KB=0
      DO 10 I=1,IEND
      KB=KB+1
      K=SEQ(I)
      B(KB)=REAL(SCHED(K))
      IF(TYPE(K) .EQ. 2) GO TO 10
      KB=KB+1
      B(KB)=AIMAG(SCHED(K))
10    CONTINUE
      POINTR(1)=1
      KB=0
      DO 1000 I=1,IEND
      KB=KB+1
      ROW=SEQ(I)
```

```
      NUMB(ROW)=KB
      WRI(1)=ROW
      PV=TYPE(ROW) .EQ. 2
      PC= .NOT. PV
      IF(PV) WRI(1)=-ROW
      K=2
      FIRST=ROW*8-7
      SUM=0.
      CALL LWRMKR(WRR,WRI,Y,CPONT,NXTR,U,FIRST,K,TYPE,KB,B,SUM)
      CALL LWRMKR(WRR,WRI,Y,RPONT,NXTC,U,FIRST,K,TYPE,KB,B,SUM)
      TEMP=Y(ROW*2)
      SUM=SUM+TEMP
      GII=REAL(TEMP)
      BII=AIMAG(TEMP)
      GI=REAL(SUM)
      BI=AIMAG(SUM)
      WRR(1,1)=BII-BI
      IF(PV) GO TO 20
      KB=KB+1
      WRR(2,1)=GII-GI
      WRR(3,1)=GII+GI
      WRR(4,1)=-BII-BI
      GO TO 30
20    U1=U(ROW)**2/2.
      B(KB)=B(KB)-(GII+GI)*U1
30    K=K-1
      IF(I .EQ. 1)GO TO 300
C     TO PERFORM TRIANGULARIZATION PROCESS ON THE ROW BEFORE STORING IT:
      J=1
90    IDONE=SEQ(J)
      IF(TYPE(IDONE) .EQ. 2) IDONE=-IDONE
      JJ=0
100   JJ=JJ+1
      IF(IDONE .EQ. WRI(JJ))GO TO 500
150   IF(JJ .LT. K) GO TO 100
      J=J+1
      IF(J .LT. I) GO TO 90
C     ROW TRIANGULARIZED. DIVIDE THE ROW BY ITS DIAGONAL ELEMENT
C     AND STORE THE RESULTS:
300   RATIO=1./WRR(1,1)
      DIAG1=RATIO
      DO 350 J=1,K
      IF(WRI(J) .EQ. 0)GO TO 350
      WRR(1,J)=WRR(1,J)*RATIO
      IF(WRI(J) .GT. 0)WRR(3,J)=WRR(3,J)*RATIO
350   CONTINUE
      IF(PV)GO TO 400
      RATIO=-WRR(2,1)
      WRR(4,1)=WRR(4,1)+WRR(3,1)*RATIO
      RATIO2=1./WRR(4,1)
      DIAG2=RATIO2
      DO 360 J=1,K
      IF(WRI(J) .EQ. 0)GO TO 360
      WRR(2,J)=(WRR(2,J)+WRR(1,J)*RATIO)*RATIO2
      IF(WRI(J) .GT. 0)WRR(4,J)=(WRR(4,J)+WRR(3,J)*RATIO)*RATIO2
360   CONTINUE
C     ROW IS READY TO BE STORED
400   INDEX=POINTR(I)
      IF(PV)POINTR(I)=-INDEX
      J=2
```

```
      AM(INDEX)=DIAG1
      INDEX=INDEX+1
      IF(PV)GO TO 420
      AM(INDEX)=DIAG2
      INDEX=INDEX+1
      AM(INDEX)=WRR(3,1)
      INDEX=INDEX+1
420   IF(J .GT. K) GO TO 460
      JJ=WRI(J)
      IF(JJ .EQ. 0)GO TO 450
      MA(INDEX)=JJ
      INDEX=INDEX+1
      AM(INDEX)=WRR(1,J)
      INDEX=INDEX+1
      IF(PV)GO TO 430
      AM(INDEX)=WRR(2,J)
      INDEX=INDEX+1
430   IF(JJ .LT. 0)GO TO 450
      AM(INDEX)=WRR(3,J)
      INDEX=INDEX+1
      IF(PV)GO TO 450
      AM(INDEX)=WRR(4,J)
      INDEX=INDEX+1
450   J=J+1
      GO TO 420
460   POINTR(I+1)=INDEX
      GO TO 1000
C     TO COMBINE A ROW WITH A PREVIOUSLY PROCESSED ROW:
500   INDEX=POINTR(J)
      IF(INDEX .LT. 0) INDEX=-INDEX
      LAST=POINTR(J+1)
      IF(LAST .LT. 0) LAST=-LAST
      RATIO=-WRR(1,JJ)
C     FIRST COMBINE THE PRESENT ROW WITH ONLY THE FIRST ROW OF THE
C     PREVIOUSLY PROCESED ROW:
      IF(PC)RATIO2=-WRR(2,JJ)
520   INDEX=INDEX+1
      IF(WRI(JJ) .LT. 0)GO TO 540
      INDEX=INDEX+1
      WRR(3,JJ)=WRR(3,JJ)+AM(INDEX)*RATIO
      IF(PC)WRR(4,JJ)=WRR(4,JJ)+AM(INDEX)*RATIO2
      INDEX=INDEX+1
540   COL=MA(INDEX)
      INDEX=INDEX+1
C     SEE IF THE CORRSEPODING OFF DIAGONAL ELEMENT EXISTS:
      DO 550 NA=1,K
      IF(COL .EQ. WRI(NA))GO TO 600
550   CONTINUE
C     IF NOT, CREATE ONE WITH ZERO VALUES:
      K=K+1
      WRI(K)=COL
      DO 560 NB=1,4
560   WRR(NB,K)=0.
      NA=K
C     CHANGE THE VALUE OF THE OFF DIAGONAL ELEMENT (THAT CERTAINLY
C     EXISTS NOW):
600   WRR(1,NA)=WRR(1,NA)+AM(INDEX)*RATIO
      IF(PC)WRR(2,NA)=WRR(2,NA)+AM(INDEX)*RATIO2
      INDEX=INDEX+1
      IF(WRI(JJ) .GT. 0)INDEX=INDEX+1
```

```
      IF(COL .LT. 0) GO TO 700
      WRR(3,NA)=WRR(3,NA)+AM(INDEX)*RATIO
      IF(PQ)WRR(4,NA)=WRR(4,NA)+AM(INDEX)*RATIO2
      INDEX=INDEX+1
      IF(WRI(JJ) .GT. 0) INDEX=INDEX+1
700   IF(INDEX .LT. LAST) GO TO 540
C     IF THE PREV PROC. ROW IS PQ, COMBINE THE SECONE ROW OF THAT:
      IF(WRI(JJ) .LT. 0)GO TO 800
      RATIO=-WRR(3,JJ)
      IF(PQ)RATIO2=-WRR(4,JJ)
      INDEX=PQINTR(J)
      INDEX=IABS(INDEX)+1
      INDEX=INDEX+2
730   COL=MA(INDEX)
      INDEX=INDEX+2
      DO 740 NA=1,K
      IF(COL .EQ. WRI(NA))GO TO 760
740   CONTINUE
      ISTNO=740
      RETURN 1
760   WRR(1,NA)=WRR(1,NA)+AM(INDEX)*RATIO
      IF(PQ)WRR(2,NA)=WRR(2,NA)+AM(INDEX)*RATIO2
      INDEX=INDEX+1
      IF(COL .LT. 0)GO TO 780
      INDEX=INDEX+1
      WRR(3,NA)=WRR(3,NA)+AM(INDEX)*RATIO
      IF(TYPE(ROW) .EQ. 1)WRR(4,NA)=WRR(4,NA)+AM(INDEX)*RATIO2
      INDEX=INDEX+1
780   IF(INDEX .LT. LAST) GO TO 730
800   WRI(JJ)=0
      GO TO 150
1000  CONTINUE
      RETURN
      END
      SUBROUTINE LSSIM(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,NXTC
     *,*)
      EXTERNAL CRDER,MODEF,FORMBP,FORBKW,MODCHG,DRAW,MODER
      COMPLEX Y(1),V(1),SCHED(1)
      INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1),NO(1),TYPE(1)
      INTEGER*2 SEQ(1),NSTEP
      INTEGER RES(2),ACCNN,ATEMP
      REAL U(1),ANG(1)
      LOGICAL*1 BATCH,ALG(8),HDNGS(80),PLOT,PRINT
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /ADS/ IA,IC,IS,ALG,BATCH
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /TYM/ RES
      COMMON /LOSSES/ SL
      EQUIVALENCE (I1,NBUS)
      IA=1
      GO TO 10
      ENTRY FLSSIM(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,NXTC,*)
      IA=2
10    CALL TIME(0)
      PLOT=LIST .LT. 0
      IF(PLOT) REWIND 3
      IF(PLOT) CALL EMPTYF(3)
      PRINT=IABS(LIST) .GT. 10
      NSTEP=IABS(LIST)/10
      N1=(IEND-1)/32+1
```

```
      NN=N1*IEND*4
      CALL GSPACE(ACCNN,NN)
      CALL GSPACE(ATEMP,IEND*2)
      CALL CALLER(CRDER,ACONN,IPTR(N1),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT
     *),IPTR(NXTC),IPTR(SEQ),ATEMP,IPTR(NPQ),IPTR(NPV),IPTR(TYPE))
      CALL FSPACE(ATEMP)
      CALL FSPACE(ACCNN)
      NWORDS=NPQ*5+NPV*3-IEND*2
      CALL GSPACE(AM,NWORDS*4)
      CALL GSPACE(AWR,NBUS*16)
      CALL GSPACE(AWI,NBUS*2)
      CALL GSPACE(APNTR,NBUS*2)
      CALL GSPACE(BV,IEND*8)
      CALL GSPACE(NUMB,IEND*2)
      CALL CALLER(MODEF,AWR,AWI,AM,AM,BV,IPTR(SEQ),IPTR(Y),IPTR(CPONT),I
     *PTR(NXTR),IPTR(RPONT),IPTR(NXTC),IPTR(SCHED),IPTR(U),IPTR(TYPE),
     *APNTR,NUMB,6400)
      CALL FSPACE(AWR)
      CALL FSPACE(AWI)
      CALL GSPACE(BP,IEND*8)
      IT=0
      IF(IA .EQ. 1) GO TO 50
      CALL CALLER(MODER ,BP,NUMB,IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(U),IP
     *TR(ANG),IPTR(SCHED),IPTR(IT),SL,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPT
     *R(RPONT),IPTR(NXTC),IPTR(ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3))
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF(.NOT. PRINT) GO TO 20
      LIMIT=NSTEP
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
20    IF(ERR2 .LE. TOLERN) GO TO 300
      GO TO 100
50    LIMIT=1
      CALL CALLER(FORMBP,BV,BP,IPTR(SEQ),SL,IPTR(IT),IPTR(V),IPTR(ANG),
     *IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT),IPTR(NXTC),IPTR(TYPE),
     *IPTR(U))
100   CALL CALLER(FORBKW,AM,AM,BP,APNTR,NUMB,IPTR(SEQ))
      CALL CALLER(MODCHG,BP,NUMB,IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(U),IP
     *TR(ANG),IPTR(SCHED),IPTR(IT),SL,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPT
     *R(RPONT),IPTR(NXTC),IPTR(ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3))
      IT=IT+1
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF( .NOT. PRINT .OR. IT .NE. LIMIT) GO TO 200
      LIMIT=LIMIT+NSTEP
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
200   IF(ERR2 .LE. TOLERN) GO TO 300
      IF(IT .GE. ITMAX) GO TO 250
      GO TO 100
250   ISTNO=250
300   CALL FSPACE(AM)
      CALL FSPACE(APNTR)
      CALL FSPACE(BV)
      CALL FSPACE(BP)
      CALL FSPACE(NUMB)
      CALL TIME(3,C,RES)
      IF(.NOT. PLOT) GO TO 350
      IF(IA .EQ. 2) IT=IT+1
```

```
      NN=(IEND*2+3)*IT*4
      CALL GSPACE(ARRAY,NN)
      CALL CALLER(DRAW,IPTR(IT),IPTR(NC),ARRAY)
      CALL FSPACE(ARRAY)
      IF(IA .EQ. 2) IT=IT-1
  350 CALL OUT1(IT,ERR1,ERR2,IERR2,ERR3)
      IF(ISTNO .NE. 0) RETURN 1
      RETURN
  400 RETURN 1
      END
      SUBROUTINE MODCHG(BP,NUMB,SEQ,TYPE,V,U,ANG,SCHED,IT,SL,Y,CPONT,NXT
     *R,RPCNT,NXTC,ERR1,ERR2,IERR2,ERR3)
      REAL BP(1),U(1),ANG(1)
      INTEGER*2 NUMB(1),CPONT(1),NXTR(1),RPONT(1),NXTC(1),SEQ(1),TYPE(1)
      COMPLEX Y(1),V(1),SL(1),SCHED(1),IK,DS
      INTEGER FIRST,PP,VP
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      LOGICAL*1 ALG(8),BATCH
      EQUIVALENCE (I1,NBUS)
      DV1=0.
      DV2=0.
      ERR2=0.
      ERR3=0.
      IF(IA .EQ. 2) GO TO 150
      IF(IT .NE. 0) GO TO 150
      DO 100 I=1,IEND
      K=SEQ(I)
      KB=NUMB(K)
      ANG(K)=-BP(KB)
      IF(TYPE(K) .EQ. 1)U(K)=SQRT(BP(KB+1)*2.)
      A1=U(K)*COS(ANG(K))
      A2=U(K)*SIN(ANG(K))
      V(K)=CMPLX(A1,A2)
  100 CONTINUE
      GO TO 200
  150 DO 170 I=1,IEND
      K=SEQ(I)
      KB=NUMB(K)
      ANG(K)=ANG(K)-BP(KB)
      IF(TYPE(K) .EQ. 1)U(K)=SQRT(U(K)**2+BP(KB+1)*2.)
      A1=U(K)*COS(ANG(K))
      A2=U(K)*SIN(ANG(K))
      V(K)=CMPLX(A1,A2)
  170 CONTINUE
      ENTRY MODER(BP,NUMB,SEQ,TYPE,V,U,ANG,SCHED,IT,SL,Y,CPONT,NXTR,
     *RPONT,NXTC,ERR1,ERR2,IERR2,ERR3)
  200 DO 400 I=1,IEND
      IK=0.
      K=SEQ(I)
      KB=NUMB(K)
      FIRST=K*8-7
      PP=NXTR(FIRST)
  310 IF(PP .EQ. FIRST) GO TO 320
      VP=(PP+7)/4
      J=CPONT(PP)
      IK=IK+Y(VP)*V(J)
      PP=NXTR(PP)
      GO TO 310
  320 PP=NXTC(FIRST)
```

```
330   IF(PP .EQ. FIRST) GO TO 350
      VP=(PP+7)/4
      J=RPCNT(PP)
      IK=IK+Y(VP)*V(J)
      PP=NXTC(PP)
      GO TO 330
350   CONTINUE
      IK=IK+Y(K*2)*V(K)
      IK=V(K)*CONJG(IK)
      DS=SCHED(K)-IK
      DP=REAL(DS)
      BP(KB)=DP
      DV1=DV1+DP
      ADD=ABS(DP)
      IF(ERR2 .GE. ADD) GO TO 360
      ERR2=ADD
      IERR2=K
360   ERR3=ERR3+ADD
      IF(TYPE(K) .EQ. 2) GO TO 400
      DQ=AIMAG(DS)
      BP(KB+1)=DQ
      DV2=DV2+DQ
      ADD=ABS(DQ)
      IF(ERR2 .GE. ADD) GO TO 370
      ERR2=ADD
      IERR2=K
370   ERR3=ERR3+ADD
400   CONTINUE
      ERR1=SQRT(DV1**2+DV2**2)
      RETURN
      END
      SUBROUTINE STOTT(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPONT,NXTR,RPONT,NXTC
     *,*)
      EXTERNAL ORDER,BPBPP,DPDQ,DVDD ,CRAW
      COMPLEX Y(1),V(1),SCHED(1)
      INTEGER*2 CPONT(1),NXTR(1),RPONT(1),NXTC(1),NO(1),TYPE(1)
      INTEGER*2 SEQ(1),NSTEP
      INTEGER AWI,RES(2)
      REAL U(1),ANG(1)
      LOGICAL*1 BATCH,ALG(8),HDNGS(80),PLOT,PRINT
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /ADS/ IA,ID,IS,ALG,BATCH
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      COMMON /TYM/ RES
      EQUIVALENCE (I1,NBUS)
      CALL TIME(0)
      PLOT=LIST .LT. 0
      IF(PLOT) REWIND 3
      IF(PLOT) CALL EMPTYF(3)
      PRINT=IABS(LIST) .GT. 10
      NSTEP=IABS(LIST)/10
      N1=(IEND-1)/32+1
      NN=N1*IEND*4
      CALL GSPACE(ACONN,NN)
      CALL GSPACE(ATEMP,IEND*2)
      CALL CALLER(ORDER,ACONN,IPTR(N1),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT
     *),IPTR(NXTC),IPTR(SEQ),ATEMP,IPTR(NPQ),IPTR(NPV),IPTR(TYPE))
      CALL FSPACE(ATEMP)
      CALL FSPACE(ACONN)
C
```

```
C       THE STORAGE IS SLIGHTLY OVER - ALLOCATED.
C
      NW1=(NPV+NPQ)*2
      NW2=NPQ*2
      CALL GSPACE(B1,NW1*4)
      CALL GSPACE(B2,NW2*4)
      CALL GSPACE(AWR,NBUS*4)
      CALL GSPACE(AWI,NBUS*2)
      CALL GSPACE(NUMB,NBUS*2)
      CALL GSPACE(B1ST,NBUS*2)
      CALL GSPACE(B2ST,NBUS*2)
      IND=1
      CALL CALLER(BPBPP,B1,B1,AWR,AWI,B1ST,IPTR(Y),IPTR(CPONT),IPTR(NXTR
     *),IPTR(RPONT),IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),NUMB,IPTR(IND))
      IND=2
      CALL CALLER(BPBPP,B2,B2,AWR,AWI,B2ST,IPTR(Y),IPTR(CPONT),IPTR(NXTR
     *),IPTR(RPONT),IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),NUMB,IPTR(IND))
      CALL FSPACE(AWR)
      CALL FSPACE(AWI)
      CALL GSPACE(DEL1,IEND*4)
      CALL GSPACE(DEL2,IEND*4)
      IT=0
      LIMIT=0
      ITLAST=ITMAX*2
      KP=1
      KQ=1
100   IND=1
      DO 110 I=1,IEND
      A1=U(I)*COS(ANG(I))
      A2=U(I)*SIN(ANG(I))
110   V(I)=CMPLX(A1,A2)
      CALL CALLER(DPDQ,DEL1,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT),
     *IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(SCHED),IPTR(U),IPTR(
     *ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3),IPTR(KP),IPTR(IND))
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF(.NOT. PRINT .OR. IT .NE. LIMIT) GO TO 120
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
      LIMIT=LIMIT+NSTEP
120   IF(KP .EQ. 0) GO TO 200
      IF(IT .GE. ITLAST) GO TO 350
      CALL CALLER(DVDQ,B1,B1,B1ST,DEL1,IPTR(ANG),IPTR(TYPE),IPTR(SEQ),
     *NUMB,IPTR(IND))
      IT=IT+1
130   IND=2
      DO 140 I=1,IEND
      A1=U(I)*COS(ANG(I))
      A2=U(I)*SIN(ANG(I))
140   V(I)=CMPLX(A1,A2)
      CALL CALLER(DPDQ,DEL2,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT),
     *IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(SCHED),IPTR(U),IPTR(
     *ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3),IPTR(KQ),IPTR(IND))
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF(.NOT. PRINT .OR. IT .NE. LIMIT) GO TO 150
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
      LIMIT=LIMIT+NSTEP
150   IF(KQ .EQ. 0) GO TO 300
```

```
      IF(IT .GE. ITLAST) GO TO 350
      CALL CALLER(DVDD,B2,B2,B2ST,DEL2,IPTR(U)  ,IPTR(TYPE),IPTR(SEQ),
     *NUMB,IPTR(IND))
      IT=IT+1
      GO TO 100
200   IF(KQ .EQ. 0) GO TO 400
      GO TO 130
300   IF(KP .EQ. 0) GO TO 400
      GO TO 100
350   IST=350
400   CALL FSPACE(B1)
      CALL FSPACE(B2)
      CALL FSPACE(B1ST)
      CALL FSPACE(B2ST)
      CALL FSPACE(NUMB)
      CALL TIME(3,C,RES)
      IF(.NOT. PLOT) GO TO 450
      IT=IT+1
      ITMAX=-ITMAX
      NN=(IEND*2+3)*IT*4
      CALL GSPACE(ARRAY,NN)
      CALL CALLER(DRAW,IPTR(IT),IPTR(NC),ARRAY)
      ITMAX=-ITMAX
      CALL FSPACE(ARRAY)
      IT=IT-1
450   CALL OUT1(IT,ERR1,ERR2,IERR2,ERR3)
      IF(ISTNO .NE. 0) RETURN 1
      RETURN
      END
      SUBROUTINE BPBPP(BR,BI,WRR,WRI,BST,Y,CPCNT,NXTR,RPCNT,NXTC,SEQ,TYP
     *E,NUMB,IND)
      REAL BR(1),WRR(1)
      INTEGER BI(1),FIRST,VP,PP
      COMPLEX Y(1)
      INTEGER*2 CPCNT(1),NXTR(1),RPCNT(1),NXTC(1),SEQ(1),TYPE(1)
      INTEGER*2 WRI(1),BST(1),NUMB(1)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE(I1,NBUS)
      LOGICAL*1 DPRIME
      DPRIME= IND .NE. 1
      IF(DPRIME) GO TO 15
      DO 10 I=1,IEND
      K=SEQ(I)
      NUMB(K)=I
10    CONTINUE
15    BST(1)=1
      DO 1000 II=1,IEND
      I=SEQ(II)
      IF(DPRIME .AND. TYPE(I) .NE. 1) GO TO 700
      K=2
      FIRST=I*8-7
      SUM=0.
      PP=NXTR(FIRST)
20    IF(PP .EQ. FIRST) GO TO 50
      VP=(PP+7)/4
      J=CPCNT(PP)
      IF(DPRIME .AND. TYPE(J) .NE. 1) GO TO 30
      BIJ=AIMAG(Y(VP))
      IF(DPRIME) GO TO 25
      GIJ=REAL(Y(VP))
```

```
      RATIC=GIJ/BIJ
      BIJ=BIJ*(1.+RATIO**2)
      SUM=SUM+BIJ
      IF(J .GT. IEND) GO TO 30
 25   WRR(K)=-BIJ
      WRI(K)=J
      K=K+1
 30   PP=NXTR(PP)
      GO TO 20
 50   PP=NXTC(FIRST)
 70   IF(PP .EQ. FIRST) GO TO 100
      VP=(PP+7)/4
      J=RPCNT(PP)
      IF(DPRIME .AND. TYPE(J) .NE. 1) GO TO 80
      BIJ=AIMAG(Y(VP))
      IF(DPRIME) GO TO 75
      GIJ=REAL(Y(VP))
      RATIC=GIJ/BIJ
      BIJ=BIJ*(1.+RATIO**2)
      SUM=SUM+BIJ
      IF(J .GT. IEND) GO TO 80
 75   WRR(K)=-BIJ
      WRI(K)=J
      K=K+1
 80   PP=NXTC(PP)
      GO TO 70
 100  WRI(1)=I
      IF(.NOT. DPRIME) WRR(1)=SUM
      IF(DPRIME) WRR(1)=-AIMAG(Y(I*2))
      K=K-1
C
C      WORKING ROW FORMATION IS COMPLETED
C
      IF(II .EQ. 1) GO TO 500
      IM1=II-1
      DO 200 JJ=1,IM1
      J=SEQ(JJ)
      IF(DPRIME .AND. TYPE(J) .NE. 1) GO TO 200
      INDEX=BST(JJ)+1
      JJ1=JJ
 105  JJ1=JJ1+1
      LAST=BST(JJ1)
      IF(LAST .EQ. 0) GO TO 105
      IF(INDEX .GE. LAST) GO TO 200
      DO 110  KK=1,K
      IF(WRI(KK) .EQ. J) GO TO 115
 110  CONTINUE
      GO TO 200
 115  RATIC=-WRR(KK)
      WRI(KK)=0
 120  JK=BI(INDEX)
      DO 130 JKK=1,K
      IF(WRI(JKK) .EQ. JK) GO TO 140
 130  CONTINUE
      K=K+1
      WRR(K)=0.
      WRI(K)=JK
      JKK=K
 140  WRR(JKK)=WRR(JKK)+BR(INDEX+1)*RATIO
      INDEX=INDEX+2
```

```
      IF(INDEX .LT. LAST) GO TO 120
200   CONTINUE
C
C      WORKING ROW COMBINATION PROCESS COMPLETED
C      NOW DIVIDE THE ROW BY ITS DIAGONAL AND STORE THE RESULTS
C
500   RATIO=1./WRR(1)
      INDEX=BST(II)
      BR(INDEX)=RATIO
      INDEX=INDEX+1
      IF(K .EQ. 1) GO TO 650
      DO 600 KK=2,K
      IF(WRI(KK) .EQ. 0) GO TO 600
      BI(INDEX)=WRI(KK)
      BR(INDEX+1)=WRR(KK)*RATIO
      INDEX=INDEX+2
600   CONTINUE
650   BST(II+1)=INDEX
      GO TO 1000
700   BST(II+1)=BST(II)
      BST(II)=0
1000  CONTINUE
      RETURN
      END
      SUBROUTINE CPCQ(DEL,Y,CPONT,NXTR,RPONT,NXTC,SEQ,TYPE,V,SCHED,U,ERR
     *1,ERR2,IERR2,ERR3,KPQ,IND)
      REAL DEL(1),U(1)
      COMPLEX Y(1),V(1),SCHED(1),IK,SK,DS
      INTEGER*2 CPCNT(1),NXTR(1),RPONT(1),NXTC(1),SEQ(1),TYPE(1)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      LOGICAL*1 DPRIME
      INTEGER FIRST,VP,PP
      KPQ=0
      ERR2=0.
      ERR3=0.
      DV1=0.
      DV2=0.
      DPRIME=IND .EQ. 2
      DO 200 II=1,IEND
      I=SEQ(II)
      FIRST=I*8-7
      IK=Y(I*2)*V(I)
      PP=NXTR(FIRST)
30    IF(PP .EQ. FIRST) GO TO 50
      VP=(PP+7)/4
      J=CPCNT(PP)
      IK=IK+Y(VP)*V(J)
      PP=NXTR(PP)
      GO TO 30
50    PP=NXTC(FIRST)
80    IF(PP .EQ. FIRST) GO TO 100
      VP=(PP+7)/4
      J=RPCNT(PP)
      IK=IK+Y(VP)*V(J)
      PP=NXTC(PP)
      GO TO 80
100   SK=V(I)*CONJG(IK)
      DS=SCHED(I)-SK
      DP=REAL(DS)
```

```
      ADD=ABS(DP)
      IF(DPRIME) GO TO 120
      DEL(II)=DP/U(I)
      IF(ADD .GT. TOLERN) KPQ=1
  120 DV1=DV1+DP
      ERR3=ERR3+ADD
      IF(ERR2 .GE. ADD) GO TO 130
      ERR2=ADD
      IERR2=I
  130 IF(TYPE(I) .EQ. 2) GO TO 200
      DQ=AIMAG(DS)
      ADD=ABS(DQ)
      IF( .NOT. DPRIME) GO TO 140
      DEL(II)=DQ/U(I)
      IF(ADD .GT. TOLERN) KPQ=1
  140 DV2=DV2+DQ
      ERR3=ERR3+ADD
      IF(ERR2 .GE. ADD) GO TO 200
      ERR2=ADD
      IERR2=I
  200 CONTINUE
      ERR1=SQRT(DV1**2+DV2**2)
      RETURN
      END
      SUBROUTINE DVDD(BR,BI,BST,DEL,UANG,TYPE,SEQ,NUMB,IND)
      REAL BR(1),DEL(1),UANG(1)
      INTEGER BI(1)
      INTEGER*2 BST(1),TYPE(1),SEQ(1),NUMB(1)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      LOGICAL*1 DPRIME
      DPRIME= IND .EQ. 2
      DO 200 II=1,IEND
      I=SEQ(II)
      IF(DPRIME .AND. TYPE(I) .EQ. 2) GO TO 200
      INDEX=BST(II)+1
      III=II
   40 III=III+1
      LAST=BST(III)
      IF(LAST .EQ. 0) GO TO 40
   50 IF(INDEX .GE. LAST) GO TO 100
      JJ=BI(INDEX)
      J=NUMB(JJ)
      DEL(J)=DEL(J)-DEL(II)*BR(INDEX+1)
      INDEX=INDEX+2
      GO TO 50
  100 INDEX=BST(II)
      DEL(II)=DEL(II)*BR(INDEX)
  200 CONTINUE
C
C      FORWARD PROCESS COMPLETED
C
C      START BACK SUBSTITUTION.
C
      DO 400 II=1,IEND
      JJ=IEND+1-II
      I=SEQ(JJ)
      IF(DPRIME .AND. TYPE(I) .EQ. 2) GO TO 400
      SUM=0.
      INDEX=BST(JJ)+1
```

```
      JJ1=JJ
240   JJ1=JJ1+1
      LAST=BST(JJ1)
      IF(LAST .EQ. 0) GO TO 240
250   IF(INDEX .GE. LAST) GO TO 300
      J=BI(INDEX)
      J=NUMB(J)
      SUM=SUM+BR(INDEX+1)*DEL(J)
      INDEX=INDEX+2
      GO TO 250
300   DEL(JJ)=DEL(JJ)-SUM
400   CONTINUE
C
C      BACK SUBSTITUTION ENDED.
C
      DO 500 I=1,IEND
      J=SEQ(I)
      IF(DPRIME) GO TO 450
      UANG(J)=UANG(J)+DEL(I)
      GO TO 500
450   IF(TYPE(J) .EQ. 2) GO TO 500
      UANG(J)=UANG(J)+DEL(I)*UANG(J)
500   CONTINUE
      RETURN
      END
      SUBROUTINE DRAW(IT,NO,ARRAY)
      REAL ARRAY(IT,1)
      INTEGER*2 NO(1)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
      LOGICAL*1 HDNGS(80)
      EQUIVALENCE (I1,NBUS)
      IF(IT .LE. 1) RETURN
      CALL FTNCMD('ASSIGN 9=PLOTFILE',17)
      CALL EMPTYF(9)
      REWIND 3
      J=1
50    K=1
      DO 100 I=1,IEND
      IF(NO(I) .GT. 0) GO TO 100
      READ(3)ARRAY(J,K),ARRAY(J,K+1)
      K=K+2
100   CONTINUE
      READ(3)ARRAY(J,K),ARRAY(J,K+1),ARRAY(J,K+2)
      J=J+1
      IF(J .LE. IT) GO TO 50
      CALL PLOTRL('METR',1)
      K=1
      DO 200 I=1,IEND
      IF(NO(I) .GT. 0) GO TO 200
      ANO=-NO(I)
      CALL DRAW2(IT,ITMAX,ARRAY,K)
      CALL SYMBOL(10.5,2.,.35,'BUS #',90.,5)
      CALL NUMBER(10.5,4.,.35,ANO,90.,-1)
      CALL SYMBOL(10.5,5.2,.35,'(U)',90.,3)
      CALL SYMBOL(10.5,14.5,.35,'BUS #',90.,5)
      CALL NUMBER(10.5,16.5,.35,ANO,90.,-1)
      CALL SYMBOL(10.5,17.7,.35,'(ANG)',90.,5)
      CALL PLOT(15.,0.,-3)
      K=K+2
```

```
  200   CONTINUE
        CALL DRAW2(IT,ITMAX,ARRAY,K)
        CALL SYMBOL(10.5,2.,.35,'ERR 1',90.,5)
        CALL SYMBOL(10.5,14.5,.35,'ERR 2',90.,5)
        CALL PLOT(15.,0.,-3)
        K=K+2
        K=-K
        CALL DRAW2(IT,ITMAX,ARRAY,K)
C       PLOT AN INVISIBLE BOUNDARY:
        CALL PLOT(10.5,24.5,3)
        CALL SYMBOL(10.5,2.,.35,'ERR 3',90.,5)
        CALL SYMBOL(0.,18.,.25,HDNGS,0.,48)
        CALL SYMBOL(3.,16.,.25,HDNGS(49),0.,32)
        RETURN
        END
        SUBROUTINE PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
        REAL U(1),ANG(1)
        INTEGER*2 NO(1)
        COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
        EQUIVALENCE (I1,NBUS)
        DO 100 I=1,IEND
        IF(NO(I) .GT. 0) GO TO 100
        ANGI=ANG(I)*180./3.14159
        WRITE(3)U(I),ANGI
  100   CONTINUE
        WRITE(3)ERR1,ERR2,ERR3
        RETURN
        END
        SUBROUTINE DRAW2(IT,ITMAX,ARRAY,KK)
        REAL ARRAY(IT,1)
        COMMON /ADS/ IA,ID,IS,ALG,BATCH
        LOGICAL*1 ALG(8),BATCH
        LOGICAL EQUC
        INDEX=0
        IF(KK .GT. 0) GO TO 20
        KK=-KK
        INDEX=1
   20   Y=1.
        K=KK
        FACTOR=1.
        IF(ITMAX .LT. 0) FACTOR=.5
        DX=IABS(ITMAX)/10.
        XINC=FACTOR/DX
   50   CALL AXCTRL('YORIG',Y)
        CALL AXCTRL('DIGITS',-1)
        CALL AXPLOT('ITERATION:',0.,10.,0.,DX)
        DIFF=ARRAY(1,K)-ARRAY(IT,K)
        IF(ABS(DIFF) .GT. .000001) GO TO 70
        IF(MOD(K,2) .EQ. 0) GO TO 65
        YMIN=.8
        DY=.05
        DO 60 I=1,IT
   60   ARRAY(I,K)=(ARRAY(I,K)-.8)*20.
        GO TO 80
   65   YMIN=-4.
        DY=1.
        DO 67 I=1,IT
   67   ARRAY(I,K)=ARRAY(I,K)+4.
        GO TO 80
   70   CALL SCALE(ARRAY(1,K),IT,10.,YMIN,DY,1)
```

```
 80     CALL AXCTRL('DIGITS',-2)
        CALL AXPLOT(':',90.,10.,YMIN,DY)
        X=0.
        IF(EQUC(ALG(1),'L')) X=XINC
        YSCLD=ARRAY(1,K)+Y
        CALL PLOT(X,YSCLD,3)
        CALL PLOT(X,YSCLD,2)
        DO 100 I=2,IT
        X=X+XINC
        YSCLD=ARRAY(I,K)+Y
        CALL PLOT(X,YSCLD,1)
 100    CONTINUE
        IF(INDEX .NE. 0) RETURN
        K=K+1
        INDEX=INDEX+1
        Y=13.5
        GO TO 50
        END
        SUBROUTINE DECPL1(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPCNT,NXTR,RPONT,
       *NXTC,*)
        EXTERNAL CRDER,MC1MC2,DECS,DU2DD,DRAW
        COMPLEX Y(1),V(1),SCHED(1)
        INTEGER*2 CPCNT(1),NXTR(1),RPONT(1),NXTC(1),NO(1),TYPE(1)
        INTEGER*2 SEQ(1),NSTEP
        INTEGER AWI,RES(2)
        REAL U(1),ANG(1)
        LOGICAL*1 BATCH,ALG(8),HDNGS(80),PLOT,PRINT
        COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
        COMMON /ADS/ IA,ID,IS,ALG,BATCH
        COMMON /OUTINF/ NPAGE,NLINE,HDNGS,ISTNO
        COMMON /TYM/ RES
        EQUIVALENCE (I1,NBUS)
        IA=1
        GO TO 10
        ENTRY DECPL2(V,U,ANG,NO,SCHED,TYPE,SEQ,Y,CPCNT,NXTR,RPCNT,NXTC,*)
        IA=2
 10     CONTINUE
        CALL TIME(0)
        PLOT=LIST .LT. 0
        IF(PLOT) REWIND 3
        IF(PLOT) CALL EMPTYF(3)
        PRINT=IABS(LIST) .GT. 10
        NSTEP=IABS(LIST)/10
        N1=(IEND-1)/32+1
        NN=N1*IEND*4
        CALL GSPACE(ACCNN,NN)
        CALL GSPACE(ATEMP,IEND*2)
        CALL CALLER(CRDER,ACCNN,IPTR(N1),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT
       *),IPTR(NXTC),IPTR(SEQ),ATEMP,IPTR(NPC),IPTR(NPV),IPTR(TYPE))
        CALL FSPACE(ATEMP)
        CALL FSPACE(ACCNN)
C
C        THE STORAGE IS SLIGHTLY OVER - ALLOCATED.
C
        NW1=(NPV+NPQ)*2
        NW2=NPC*2
        CALL GSPACE(B1,NW1*4)
        CALL GSPACE(B2,NW2*4)
        CALL GSPACE(AWR,NBUS*4)
        CALL GSPACE(AWI,NBUS*2)
```

```
      CALL GSPACE(NUMB,NBUS*2)
      CALL GSPACE(B1ST,NBUS*2)
      CALL GSPACE(B2ST,NBUS*2)
      IND=1
      CALL CALLER(MC1MC2,B1,B1,AWR,AWI,B1ST,IPTR(Y),IPTR(CPONT),IPTR(NXT
     *R),IPTR(RPONT),IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),NUMB,IPTR(IND))
      IND=2
      CALL CALLER(MC1MC2,B2,B2,AWR,AWI,B2ST,IPTR(Y),IPTR(CPONT),IPTR(NXT
     *R),IPTR(RPONT),IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),NUMB,IPTR(IND))
      CALL FSPACE(AWR)
      CALL FSPACE(AWI)
      CALL GSPACE(DEL1,IEND*4)
      CALL GSPACE(DEL2,IEND*4)
      IT=0
      LIMIT=0
      ITLAST=ITMAX*2
      KP=1
      KQ=1
100   IND=1
      DO 110 I=1,IEND
      A1=U(I)*COS(ANG(I))
      A2=U(I)*SIN(ANG(I))
110   V(I)=CMPLX(A1,A2)
      CALL CALLER(DECS,DEL1,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT),
     *IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(SCHED),IPTR(U),IPTR(
     *ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3),IPTR(KP),IPTR(IND))
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF(.NOT. PRINT .OR. IT .NE. LIMIT) GO TO 120
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
      LIMIT=LIMIT+NSTEP
120   IF(KP .EQ. 0) GO TO 200
      IF(IT .GE. ITLAST) GO TO 350
      CALL CALLER(DU2DD,B1,B1,B1ST,DEL1,IPTR(ANG),IPTR(TYPE),IPTR(SEQ),
     *NUMB,IPTR(IND))
      IT=IT+1
130   IND=2
      DO 140 I=1,IEND
      A1=U(I)*COS(ANG(I))
      A2=U(I)*SIN(ANG(I))
140   V(I)=CMPLX(A1,A2)
      CALL CALLER(DECS,DEL2,IPTR(Y),IPTR(CPONT),IPTR(NXTR),IPTR(RPONT),
     *IPTR(NXTC),IPTR(SEQ),IPTR(TYPE),IPTR(V),IPTR(SCHED),IPTR(U),IPTR(
     *ERR1),IPTR(ERR2),IPTR(IERR2),IPTR(ERR3),IPTR(KQ),IPTR(IND))
      IERR2=NO(IERR2)
      IF(IERR2 .LT. 0) IERR2=-IERR2
      IF(PLOT) CALL PLOTR(U,ANG,ERR1,ERR2,ERR3,NO)
      IF(.NOT. PRINT .OR. IT .NE. LIMIT) GO TO 150
      CALL PRINTR(IT,V,U,ANG,ERR1,ERR2,IERR2,ERR3,NO)
      LIMIT=LIMIT+NSTEP
150   IF(KQ .EQ. 0) GO TO 300
      IF(IT .GE. ITLAST) GO TO 350
      CALL CALLER(DU2DD,B2,B2,B2ST,DEL2,IPTR(U)  ,IPTR(TYPE),IPTR(SEQ),
     *NUMB,IPTR(IND))
      IT=IT+1
      GO TO 100
200   IF(KQ .EQ. 0) GO TO 400
      GO TO 130
300   IF(KP .EQ. 0) GO TO 400
```

```
      GC TC 100
350   IST=350
400   CALL FSPACE(B1)
      CALL FSPACE(B2)
      CALL FSPACE(B1ST)
      CALL FSPACE(B2ST)
      CALL FSPACE(NUMB)
      CALL TIME(3,C,RES)
      IF(.NCT. PLOT) GO TO 450
      IT=IT+1
      ITMAX=-ITMAX
      NN=(IEND*2+3)*IT*4
      CALL GSPACE(ARRAY,NN)
      CALL CALLER(DRAW,IPTR(IT),IPTR(NC),ARRAY)
      ITMAX=-ITMAX
      CALL FSPACE(ARRAY)
      IT=IT-1
450   CALL OUT1(IT,ERR1,ERR2,IERR2,ERR3)
      IF(ISTNO .NE. 0) RETURN 1
      RETURN
      END
      SUBRCUTINE MC1MC2(BR,BI,WRR,WRI,BST,Y,CPONT,NXTR,RPONT,NXTC,SEQ,
     *TYPE,NUMB,INC)
      REAL BR(1),WRR(1)
      INTEGER BI(1),FIRST,VP,PP
      COMPLEX Y(1)
      INTEGER*2 CPCNT(1),NXTR(1),RPONT(1),NXTC(1),SEC(1),TYPE(1)
      INTEGER*2 WRI(1),BST(1),NUMB(1)
      COMMCN /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      COMMCN /ADS/ IA,ID,IS,ALG,BATCH
      LCGICAL*1 ALC(8),BATCH
      EQUIVALENCE(I1,NBUS)
      LCGICAL*1 DPRIME
      DPRIME= IND .NE. 1
      IF(DPRIME) GC TO 15
      DO 10 I=1,IENC
      K=SEC(I)
      NUMB(K)=I
10    CONTINUE
15    BST(1)=1
      DC 1000 II=1,IEND
      I=SEQ(II)
      IF(DPRIME .AND. TYPE(I) .NE. 1) GC TO 700
      K=2
      FIRST=I*8-7
      SUM=C.
      PP=NXTR(FIRST)
20    IF(PP .EQ. FIRST) GO TO 50
      VP=(PP+7)/4
      J=CPCNT(PP)
      BIJ=AIMAG(Y(VP))
      IF(IA .EQ. 1 .CR. DPRIME) GO TO 23
      GIJ=REAL(Y(VP))
      RATIC=GIJ/BIJ
      BIJ=BIJ*(1.+RATIC**2)
23    SUM=SUM+BIJ
      IF((DPRIME .AND. TYPE(J) .NE. 1) .OR. J .GT. IEND) GO TO 30
25    WRR(K)=-BIJ
      WRI(K)=J
      K=K+1
```

```
30    PP=NXTR(PP)
      GO TO 20
50    PP=NXTC(FIRST)
70    IF(PP .EQ. FIRST) GO TO 100
      VP=(PP+7)/4
      J=RPCNT(PP)
      BIJ=AIMAG(Y(VP))
      IF(IA .EQ. 1 .OR. DPRIME) GO TO 73
      GIJ=REAL(Y(VP))
      RATIO=GIJ/BIJ
      BIJ=BIJ*(1.+RATIO**2)
73    SUM=SUM+BIJ
      IF((DPRIME .AND. TYPE(J) .NE. 1) .OR. J .GT. IEND) GO TO 80
75    WRR(K)=-BIJ
      WRI(K)=J
      K=K+1
80    PP=NXTC(PP)
      GO TO 70
100   WRI(1)=I
      IF(.NOT. DPRIME) WRR(1)=+SUM
      IF(DPRIME) WRR(1)=-2.*AIMAG(Y(I*2))-SUM
      K=K-1
C
C      WORKING ROW FORMATION IS COMPLETED
C
      IF(II .EQ. 1) GO TO 500
      IM1=II-1
      DO 200 JJ=1,IM1
      J=SEQ(JJ)
      IF(DPRIME .AND. TYPE(J) .NE. 1) GO TO 200
      INDEX=BST(JJ)+1
      JJ1=JJ
105   JJ1=JJ1+1
      LAST=BST(JJ1)
      IF(LAST .EQ. 0) GO TO 105
      IF(INDEX .GE. LAST) GO TO 200
      DO 110  KK=1,K
      IF(WRI(KK) .EQ. J) GO TO 115
110   CONTINUE
      GO TO 200
115   RATIO=-WRR(KK)
      WRI(KK)=0
120   JK=BI(INDEX)
      DO 130 JKK=1,K
      IF(WRI(JKK) .EQ. JK) GO TO 140
130   CONTINUE
      K=K+1
      WRR(K)=0.
      WRI(K)=JK
      JKK=K
140   WRR(JKK)=WRR(JKK)+BR(INDEX+1)*RATIO
      INDEX=INDEX+2
      IF(INDEX .LT. LAST) GO TO 120
200   CONTINUE
C
C      WORKING ROW COMBINATION PROCESS COMPLETED
C      NOW DIVIDE THE ROW BY ITS DIAGONAL AND STORE THE RESULTS
C
500   RATIO=1./WRR(1)
      INDEX=BST(II)
```

```
      BR(INDEX)=RATIO
      INDEX=INDEX+1
      IF(K .EQ. 1) GO TO 650
      DO 600 KK=2,K
      IF(WRI(KK) .EQ. 0) GO TO 600
      BI(INDEX)=WRI(KK)
      BR(INDEX+1)=WRR(KK)*RATIO
      INDEX=INDEX+2
600   CONTINUE
650   BST(II+1)=INDEX
      GO TO 1000
700   BST(II+1)=BST(II)
      BST(II)=0
1000  CONTINUE
      RETURN
      END
      SUBROUTINE DECS(DEL,Y,CPCNT,NXTR,RPCNT,NXTC,SEQ,TYPE,V,SCHED,U,ERR
     *1,ERR2,IERR2,ERR3,KPQ,IND)
      REAL DEL(1),U(1)
      COMPLEX Y(1),V(1),SCHED(1),IK,SK,DS
      INTEGER*2 CPCNT(1),NXTR(1),RPCNT(1),NXTC(1),SEQ(1),TYPE(1)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      LOGICAL*1 DPRIME
      INTEGER FIRST,VP,PP
      KPQ=0
      ERR2=0.
      ERR3=0.
      DV1=0.
      DV2=0.
      DPRIME=IND .EQ. 2
      DO 200 II=1,IEND
      I=SEQ(II)
      FIRST=I*8-7
      IK=Y(I*2)*V(I)
      PP=NXTR(FIRST)
30    IF(PP .EQ. FIRST) GO TO 50
      VP=(PP+7)/4
      J=CPCNT(PP)
      IK=IK+Y(VP)*V(J)
      PP=NXTR(PP)
      GO TO 30
50    PP=NXTC(FIRST)
80    IF(PP .EQ. FIRST) GO TO 100
      VP=(PP+7)/4
      J=RPCNT(PP)
      IK=IK+Y(VP)*V(J)
      PP=NXTC(PP)
      GO TO 80
100   SK=V(I)*CONJG(IK)
      DS=SCHED(I)-SK
      DP=REAL(DS)
      ADD=ABS(DP)
      IF(DPRIME) GO TO 120
      DEL(II)=DP
      IF(ADD .GT. TOLERN) KPQ=1
120   DV1=DV1+DP
      ERR3=ERR3+ADD
      IF(ERR2 .GE. ADD) GO TO 130
      ERR2=ADD
```

```
      IERR2=I
  130 IF(TYPE(I) .EQ. 2) GO TO 200
      DQ=AIMAG(DS)
      ADD=ABS(DQ)
      IF( .NOT. DPRIME) GO TO 140
      DEL(II)=DQ
      IF(ADD .GT. TOLERN) KPQ=1
  140 DV2=DV2+DQ
      ERR3=ERR3+ADD
      IF(ERR2 .GE. ADD) GO TO 200
      ERR2=ADD
      IERR2=I
  200 CONTINUE
      ERR1=SQRT(DV1**2+DV2**2)
      RETURN
      END
      SUBROUTINE DU2DD(BR,BI,BST,DEL,UANG,TYPE,SEQ,NUMB,IND)
      REAL BR(1),DEL(1),UANG(1)
      INTEGER BI(1)
      INTEGER*2 BST(1),TYPE(1),SEQ(1),NUMB(1)
      COMMON /ININF/ I1,I2,I3,ITMAX,TOLERN,ZERO,BMVA,LINE,LIST,IEND
      EQUIVALENCE (I1,NBUS)
      LOGICAL*1 DPRIME
      DPRIME= IND .EQ. 2
      DO 200 II=1,IEND
      I=SEQ(II)
      IF(DPRIME .AND. TYPE(I) .EQ. 2) GO TO 200
      INDEX=BST(II)+1
      II1=II
   40 II1=II1+1
      LAST=BST(II1)
      IF(LAST .EQ. 0) GO TO 40
   50 IF(INDEX .GE. LAST) GO TO 100
      JJ=BI(INDEX)
      J=NUMB(JJ)
      DEL(J)=DEL(J)-DEL(II)*BR(INDEX+1)
      INDEX=INDEX+2
      GO TO 50
  100 INDEX=BST(II)
      DEL(II)=DEL(II)*BR(INDEX)
  200 CONTINUE
C
C      FORWARD PROCESS COMPLETED
C
C      START BACK SUBSTITUTION.
C
      DO 400 II=1,IEND
      JJ=IEND+1-II
      I=SEQ(JJ)
      IF(DPRIME .AND. TYPE(I) .EQ. 2) GO TO 400
      SUM=0.
      INDEX=BST(JJ)+1
      JJ1=JJ
  240 JJ1=JJ1+1
      LAST=BST(JJ1)
      IF(LAST .EQ. 0) GO TO 240
  250 IF(INDEX .GE. LAST) GO TO 300
      J=BI(INDEX)
      J=NUMB(J)
      SUM=SUM+BR(INDEX+1)*DEL(J)
```

```
      INDEX=INDEX+2
      GO TO 250
300   DEL(JJ)=DEL(JJ)-SUM
400   CONTINUE
C
C      BACK SUBSTITUTION ENDED.
C
      DO 500 I=1,IEND
      J=SEQ(I)
      IF(DPRIME) GO TO 450
      UANG(J)=UANG(J)+DEL(I)
      GO TO 500
450   IF(TYPE(J) .EQ. 2) GO TO 500
      UANG(J)=SQRT(UANG(J)**2+2.*DEL(I))
500   CONTINUE
      RETURN
      END
END OF FILE
```

```
GTNAME   CSECT
*****************************************************************
*        CALLING SEQUENCE FROM FORTRAN:                         *
*        CALL GTNAME(MTU,NAME,LEN)                              *
*        WHERE                                                  *
*              MTU IS THE LOCATION OF AN ARRAY THAT CONTAINS THE NAME *
*        OF TH E MTS UNIT. NAME MUST BE 8 CHARACTERS. IF IT IS LESS   *
*        THAN  8, IT MUST BE LEFT JUSTIFIED WITH TRAILING BLANKS.     *
*        EG.   'SCARDS  '.
*              NAME(1) IS A LOGICAL*1 ARRAY THAT IN RETURN WILL CONTAIN
*        THE   NAME OF THE FILE ASSIGNED TO THAT UNIT.               *
*              LEN IS AN INTEGER VARIABLE THAT IN RETURN WILL CONTAIN *
*        THE   NUMBER OF CHARACTERS IN NAME(1).                      *
*****************************$*****************************************
         USING *,9
         STM   14,12,12(13)        STORE ALL THE REGISTERS BUT 13.
         LR    9,15                SET BASE REGISTER
         LA    10,SAVE
         ST    10,8(0,13)          SET THE FORWARD LINK
         ST    13,4(0,10)          SET THE BACKWARD LINK
         LR    13,10               ESTABLISH THE SAVE AREA
         L     5,0(C,1)            GR5=A(MTU)
         L     6,4(C,1)            GR6=A(NAME)
         L     7,8(C,1)            GR7=A(LEN)
         LM    0,1,C(5)            GET THE NAME OF THE UNIT.
         L     15,=V(GDINFO)                THEN
         BALR  14,15               CALL THE SUBROUTINE GDINFO
         LTR   15,15               CHECK RETURN CODE
         BNZ   ERRCR               AND BRANCH IF NON-ZERO
         L     1,36(0,1)           GR1=A(LENGTH AND NAME)
         LH    2,0(0,1)            GR2=LENGTH OF NAME
         LA    1,2(0,1)            GR1=FIRST LOCATION OF NAME
         ST    2,0(0,7)            SET LEN = LENGTH
         BCTR  2,C                 SUBTRACT ONE FROM LENGTH
         EX    2,MOVE              EXECUTE MOVE
         L     13,4(0,13)          RESTORE SAVE AREA ADDRESS
         LM    14,12,12(13)        RESTORE GENERAL REGISTERS
         SR    15,15               INDICATE ZERO RETURN CODE
         BCR   15,14               RETURN
ERROR    L     13,4(0,13)          RESTORE SAVE AREA ADDRESS
         L     14,12(0,13)         RESTORE RETURN ADDRESS
         LM    0,12,20(13)         RESTORE GR0 TO GR12
         BC    15,14               RETURN
MOVE     MVC   0(0,6),0(1)         MOVE 'FILE NAME' TO 'NAME'
NAME     DS    4F
SAVE     DS    18F                 SAVE AREA
         END
NAMBER   CSECT
         USING *,9
         STM   14,12,12(13)
         LR    9,15
         LA    11,SAVE
         ST    11,8(0,13)      SET BACKWARD LINK.
         ST    13,4(0,11)      SET FORWARD LINK.
         LR    13,11           ESTABLISH SAVE AREA.
         L     4,0(C,1)        LOAD A(ARRAY)
         L     5,4(C,1)        LOAD A(IEND)
         L     5,0(C,5)        LOAD IEND
*
```

```
*      TO INITIALIZE
*
            LA     7,0(0,0)        WORD INDICATOR=0
            LA     3,33(0,0)       BIT INDICATOR=33
            LR     6,7             COUNT=0
            L      2,0(7,4)        LOAD PROPER WORD.
GOON        BCT    3,NEXT          DECREMENT BIT INDI. :GO TO NEXT IF NON-ZE
            LA     3,32(0,0)       RE-INIT. BIT INDICATOR.
            LA     7,4(0,7)        MODIFY WORD INDICATOR.
            L      2,0(7,4)        LOAD PROPER WORD.
NEXT        LTR    2,2             TEST REGISTER 2
            BC     11,ZERO         BRANCH IF NON-NEGATIVE.
            LA     6,1(C,6)        MODIFY COUNT
ZERO        SLL    2,1(0)          SHIFT LEFT.
            BCT    5,GOON          DECREMENT IEND: BRANCH IF NOT ZERO.
FINISHED    LR     0,6             STORE COUNT.
            L      13,4(0,13)      RESTORE SAVE AREA ADDRESS.
            LM     14,15,12(13)    RESTORE GENERAL REGISTERS.
            LM     1,12,24(13)     EXCEPT GR 0.
            SR     15,15           INDICATE ZERO RETURN CODE.
            BCR    15,14           RETURN.
SAVE        DS     18F
            END
END OF FILE
```

APPENDIX B

DATA FOR TEST SYSTEMS 6, 9 AND 10

The data for Test Systems 6, 9 and 10 are given below. This data was obtained from B.C. Hydro and is published here with their permission. The data for all the other test systems, are published in the respective references, mentioned in Table 2.2.

For the sake of clarity, the listing of input data, produced by LFP, is used. The format of the input data (for LFP) is as explained in Appendix A, section A.4. In the listing that follows, the names SAMPLE (4001), SAMPLE (5001) and BCH.138 in the heading refer, respectively, to Test Systems 6, 9 and 10.

ALG.=TEST CATA=SAMPLE(4001) START=F 09:27 P.M. JULY 24, 1977 PAGE 1

****

B U S I N F O R M A T I O N :

++++++++++++++++++++++++++++++

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
1	ING230	3	0.0	0.0	0.0	0.0	230.00	235.00
3	JHT 13	2	127.00	0.0	0.0	0.0	13.80	13.80
4	LDR 13	1	47.00	5.00	0.0	0.0	13.80	0.0
5	SCA 13	1	25.80	5.00	0.0	0.0	13.80	0.0
6	PUN 13	2	24.50	0.0	17.80	8.15	13.80	14.30
7	ASH 13	2	26.70	0.0	0.0	0.0	13.80	13.80
8	GGA 13	2	72.00	0.0	0.0	0.0	13.80	13.80
10	JHT132	1	0.0	0.0	0.0	0.0	132.00	0.0
11	CBY132	1	0.0	0.0	64.00	6.40	132.00	0.0
12	LDR132	1	0.0	0.0	0.0	0.0	132.00	0.0
13	SCA132	1	0.0	0.0	0.0	0.0	132.00	0.0
14	GLD132	1	0.0	0.0	5.00	2.50	132.00	0.0
16	TAS132	1	0.0	0.0	18.00	5.40	132.00	0.0
17	CTI132	1	0.0	0.0	0.0	0.0	132.00	0.0
19	CBL132	1	0.0	0.0	10.60	5.30	132.00	0.0
20	PUN132	1	0.0	0.0	10.00	3.00	132.00	0.0
22	CMR132	1	0.0	0.0	0.0	0.0	132.00	0.0
23	PAL132	1	0.0	0.0	107.30	36.30	132.00	0.0
24	GCL132	1	0.0	0.0	6.90	2.60	132.00	0.0
25	ASH132	1	0.0	0.0	0.0	0.0	132.00	0.0
26	NOR132	1	0.0	0.0	9.60	4.80	132.00	0.0
27	HMC132	1	0.0	0.0	24.00	7.20	132.00	0.0
29	JPT132	1	0.0	0.0	0.0	0.0	132.00	0.0
30	VIT230	1	0.0	0.0	0.0	0.0	230.00	0.0
31	VIT 12	1	0.0	0.0	0.0	0.0	12.60	0.0
33	JPT 60	1	0.0	0.86	27.80	15.90	60.00	0.0
34	JPT 12	1	0.0	15.00	0.0	0.0	12.00	0.0
35	GGA132	1	0.0	0.0	0.0	0.0	132.00	0.0
36	CFT132	1	0.0	0.0	65.40	20.30	132.00	0.0
37	ARN230	1	0.0	0.0	0.0	0.0	230.00	0.0
38	ARN132	1	0.0	0.0	0.0	0.0	132.00	0.0
39	STD132	1	25.00	93.00	197.50	140.50	132.00	0.0
40	VIT132	1	0.0	0.0	0.0	0.0	132.00	0.0

T R A N S F O R M E R I N F O R M T I O N :

TX #	TX NAME	S.E.	R.E.	LEAKAGE IMPEDANCE		S.E. TAP	R.E. TAP
1	SCA132	5	13	0.0163	0.2720	1.13	1.00
2	LDR132	4	12	0.0095	0.1583	1.13	1.00
3	JHT132	3	10	0.0037	0.0624	1.15	1.00
4	PUN132	6	20	0.0353	0.5893	1.05	1.00
5	ASH132	7	25	0.0220	0.3671	1.10	1.00
6	JPT 60	29	33	0.0118	0.2960	1.03	1.00
7	JPT 12	33	34	0.0064	0.1602	1.00	1.00
8	GGA132	8	35	0.0039	0.0982	1.10	1.00
10	VIT132	30	40	0.0030	0.0600	1.02	1.00
11	VIT 12	30	31	0.0033	0.0667	0.98	1.00
12	ARN132	37	38	0.0011	0.0228	1.03	1.00

LINE INFORMATION:

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
15	TAS132	14	16	0.0157	0.0393	0.00880	0.00880	F
17	DBY132	10	11	0.0017	0.0052	0.00440	0.00440	F
19	GLD132	13	14	0.0516	0.1290	0.02860	0.02860	F
20	SCA132	12	13	0.0168	0.0448	0.00900	0.00900	F
21	LDR132	10	12	0.0017	0.0086	0.00840	0.00840	F
22	CBL132	17	19	0.0042	0.0123	0.00280	0.00280	F
23	CTI132	10	17	0.0021	0.0061	0.00140	0.00140	F
24	PUN132	17	20	0.0410	0.1209	0.02660	0.02660	F
25	DMR132	10	22	0.0410	0.1227	0.10420	0.10420	F
26	DMR132	20	22	0.0404	0.1194	0.02620	0.02620	F
27	ASH132	24	25	0.0161	0.0427	0.00840	0.00840	F
28	GCL132	23	24	0.0176	0.0458	0.00920	0.00920	F
29	PAL132	22	23	0.0107	0.0326	0.02760	0.02760	F
30	JPT132	22	29	0.0254	0.0750	0.06560	0.06560	F
31	JPT132	26	29	0.0165	0.0431	0.00860	0.00860	F
32	JPT132	27	29	0.0165	0.0431	0.00860	0.00860	F
33	STD132	29	39	0.0440	0.1299	0.02840	0.02840	F
34	GGA132	29	35	0.0416	0.1231	0.02680	0.02680	F
35	CFT132	35	36	0.0128	0.0380	0.00820	0.00820	F
36	STD132	36	39	0.0094	0.0279	0.00620	0.00620	F
37	VIT132	39	40	0.0004	0.0031	0.00300	0.00300	F
39	VIT132	38	40	0.0284	0.0643	0.81900	0.81900	F
40	ARN230	1	37	0.0026	0.0158	0.03060	0.03060	F

ALG.=TEST DATA=SAMPLE(5001) START=F 09:26 P.M. JULY 24, 1977 PAGE 1

****

B U S I N F O R M A T I O N :

++++++++++++++++++++++++++++

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
21	BUS 21	1	0.0	0.0	0.0	0.0	500.00	0.0
22	BUS 22	1	0.0	0.0	0.0	0.0	500.00	0.0
23	BUS 23	1	0.0	0.0	0.0	0.0	500.00	0.0
24	BUS 24	1	0.0	0.0	0.0	0.0	500.00	0.0
25	BUS 25	1	0.0	0.0	0.0	0.0	500.00	0.0
27	BUS 27	1	0.0	0.0	44.40	21.50	138.00	0.0
28	BUS 28	2	333.00	0.0	0.0	0.0	13.80	14.15
29	BUS 29	1	0.0	0.0	0.0	0.0	230.00	0.0
30	BUS 30	1	0.0	0.0	259.00	0.0	230.00	0.0
31	BUS 31	1	0.0	0.0	0.0	0.0	230.00	0.0
32	BUS 32	1	0.0	0.0	0.0	0.0	360.00	0.0
33	BUS 33	1	0.0	0.0	0.0	0.0	360.00	0.0
34	BUS 34	2	208.00	0.0	0.0	0.0	13.80	14.15
35	BUS 35	1	0.0	0.0	0.0	0.0	230.00	0.0
36	BUS 36	1	0.0	0.0	0.0	0.0	360.00	0.0
37	BUS 37	1	0.0	0.0	124.00	29.00	230.00	0.0
38	BUS 38	1	0.0	0.0	0.0	0.0	230.00	0.0
39	BUS 39	1	0.0	0.0	0.0	0.0	230.00	0.0
40	BUS 40	1	0.0	0.0	0.0	0.0	230.00	0.0
41	BUS 41	1	0.0	0.0	66.40	32.10	230.00	0.0
42	BUS 42	1	0.0	0.0	0.0	0.0	230.00	0.0
43	BUS 43	1	0.0	0.0	0.0	0.0	230.00	0.0
44	BUS 44	1	0.0	0.0	83.50	40.40	230.00	0.0
45	BUS 45	1	0.0	0.0	111.70	54.10	230.00	0.0
46	BUS 46	1	0.0	0.0	0.0	0.0	230.00	0.0
47	BUS 47	1	0.0	0.0	0.0	0.0	230.00	0.0
48	BUS 48	1	0.0	0.0	682.00	45.00	230.00	0.0
49	BUS 49	1	0.0	0.0	23.60	22.20	230.00	0.0
50	BUS 50	2	972.00	0.0	0.0	0.0	13.80	14.15
51	BUS 51	2	306.00	0.0	0.0	0.0	13.80	14.15
52	BUS 52	2	42.00	0.0	0.0	0.0	13.80	13.80
53	BUS 53	2	22.00	0.0	0.0	0.0	13.80	13.80
54	BUS 54	2	60.00	0.0	12.00	5.80	13.80	13.80
55	BUS 55	2	140.00	0.0	0.0	0.0	13.80	14.15
56	BUS 56	2	50.00	0.0	0.0	0.0	13.80	13.80
57	BUS 57	2	52.50	0.0	0.0	0.0	4.40	4.40
58	BUS 58	2	105.60	0.0	0.0	0.0	13.80	13.80
59	BUS 59	2	3241.00	756.00	1955.00	70.00	230.00	241.30
60	BUS 60	3	11337.00	-886.50	9092.00	-3645.00	100.00	113.00
61	BUS 61	2	826.00	292.00	411.00	-24.00	100.00	110.00
62	BUS 62	2	936.00	711.00	389.00	-290.00	100.00	115.00
63	BUS 63	2	0.0	0.0	0.0	0.0	12.30	12.60

B U S I N F O R M A T I O N :

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
64	BUS 64	2	0.0	0.0	0.0	0.0	12.50	12.50
65	BUS 65	2	0.0	0.0	0.0	0.0	12.50	12.50
66	BUS 66	2	0.0	0.0	0.0	0.0	13.80	14.15
67	BUS 67	2	290.00	200.00	0.0	0.0	16.50	16.50
68	BUS 68	1	0.0	0.0	70.20	17.50	230.00	0.0
69	BUS 69	1	0.0	0.0	150.00	61.00	66.00	0.0
70	BUS 70	1	0.0	0.0	0.0	76.87	12.00	0.0
71	BUS 71	1	0.0	0.0	3.70	1.80	66.00	0.0
72	BUS 72	1	0.0	0.0	9.30	4.50	66.00	0.0
73	BUS 73	1	0.0	0.0	0.40	0.20	66.00	0.0
74	BUS 74	1	0.0	0.0	0.0	0.0	360.00	0.0
75	BUS 75	1	0.0	0.0	0.0	0.0	230.00	0.0
76	BUS 76	1	0.0	0.0	140.60	43.50	60.00	0.0
77	BUS 77	1	0.0	0.0	47.20	16.40	60.00	0.0
78	BUS 78	1	0.0	0.0	31.60	14.50	60.00	0.0
79	BUS 79	1	0.0	0.0	106.80	49.90	60.00	0.0
80	BUS 80	1	0.0	0.0	3.80	1.30	60.00	0.0
81	BUS 81	1	0.0	0.0	5.70	1.90	60.00	0.0
82	BUS 82	1	0.0	0.0	81.60	39.50	60.00	0.0
83	BUS 83	1	0.0	0.0	95.20	46.10	60.00	0.0
84	BUS 84	1	0.0	0.0	110.10	53.30	60.00	0.0
85	BUS 85	1	0.0	0.0	155.70	69.80	60.00	0.0
86	BUS 86	1	0.0	0.0	28.50	12.30	60.00	0.0
87	BUS 87	1	0.0	0.0	29.00	12.40	60.00	0.0
88	BUS 88	1	0.0	0.0	711.00	-343.00	230.00	0.0
89	BUS 89	2	0.0	0.0	3152.00	-756.00	500.00	506.80
90	BUS 90	1	0.0	0.0	312.00	243.09	230.00	0.0
91	BUS 91	1	0.0	0.0	200.00	48.11	132.00	0.0
92	BUS 92	1	0.0	0.0	33.90	11.20	60.00	0.0
95	BUS 95	1	0.0	0.0	28.90	11.30	230.00	0.0
96	BUS 96	1	0.0	0.0	36.70	17.80	230.00	0.0
97	BUS 97	1	0.0	0.0	0.0	0.0	230.00	0.0
98	BUS 98	1	0.0	0.0	0.0	-69.02	12.00	0.0
99	BUS 99	1	0.0	0.0	18.90	8.10	60.00	0.0
100	BUS100	1	0.0	0.0	7.90	3.80	60.00	0.0
101	BUS101	1	0.0	0.0	0.0	0.0	60.00	0.0
102	BUS102	1	0.0	0.0	85.80	18.90	60.00	0.0
103	BUS103	1	0.0	0.0	7.30	3.50	60.00	0.0
104	BUS104	1	0.0	0.0	9.30	4.50	60.00	0.0
105	BUS105	1	0.0	0.0	31.30	15.10	60.00	0.0
106	BUS106	1	0.0	0.0	8.60	4.20	60.00	0.0
107	BUS107	1	0.0	0.0	16.40	28.42	60.00	0.0

B U S I N F C R M A T I C N :

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
108	BUS108	1	0.0	0.0	36.90	27.86	60.00	0.0
109	BUS109	1	0.0	0.0	38.60	13.20	360.00	0.0
110	BUS110	1	0.0	0.0	0.0	-76.87	12.00	0.0
111	BUS111	2	275.00	0.0	0.0	0.0	13.80	14.15
112	BUS112	1	0.0	0.0	21.40	8.50	230.00	0.0
113	BUS113	1	0.0	0.0	30.30	10.70	60.00	0.0
114	BUS114	1	0.0	0.0	0.0	0.0	230.00	0.0
115	BUS115	1	0.0	0.0	360.00	360.68	13.80	0.0
270	BUS270	2	0.0	0.0	0.0	0.0	100.00	113.01

T R A N S F C R M E R I N F C R M T I O N :

TX #	TX NAME	S.E.	R.E.	LEAKAGE IMPEDANCE		S.E. TAP	R.E. TAP
96		21	28	0.0	0.0558	1.00	0.98
97		21	50	0.0	0.0143	1.00	0.98
98		69	63	0.0	0.0959	1.00	0.93
99		90	91	0.0	0.0968	1.00	1.03
100		90	92	0.0	0.2253	1.00	1.02
301		46	85	0.0	0.0714	1.00	1.02
302		46	85	0.0	0.0768	1.00	1.06
303		31	76	0.0	0.1108	1.00	1.03
304		24	31	0.0	0.0193	1.00	1.03
305		39	82	0.0	0.2280	1.00	1.07
306		39	82	0.0	0.2280	1.00	1.07
307		22	29	0.0	0.0305	1.00	1.00
308		29	69	0.0	0.1112	1.00	1.05
309		29	69	0.0	0.1132	1.00	1.05
310		21	27	0.0	0.0648	1.00	1.00
311		90	91	0.0	0.0500	1.00	1.03
312		90	91	0.0	0.0500	1.00	1.03
313		90	91	0.0	0.9550	1.00	1.03
314		23	30	0.0	0.0309	1.00	1.00
315		30	98	0.0	0.0720	1.00	0.97
316		37	99	0.0	0.4285	1.00	0.97
317		37	99	0.0	0.4110	1.00	0.97
318		101	97	0.0	0.1147	1.00	0.91
319		38	102	0.0	0.0404	1.00	0.97
321		101	97	0.0	0.1147	1.00	0.91
323		22	29	0.0	0.0300	1.00	1.00
324		21	111	0.0	0.0558	1.00	0.98
326		29	110	0.0	0.0715	1.00	1.06
68		44	83	0.0	0.1335	1.00	1.07
69		25	47	0.0	0.0117	1.00	1.00
70		75	55	0.0	0.0835	1.00	1.00
71		40	67	0.0	0.0154	1.00	0.95
72		42	66	0.0	0.1137	1.00	0.96
73		44	83	0.0	0.1333	1.00	1.07
74		46	85	0.0	0.0768	1.00	1.06
75		31	76	0.0	0.1108	1.00	1.03
76		32	31	0.0	0.0158	1.00	1.08
77		24	31	0.0	0.0193	1.00	1.03
78		39	82	0.0	0.2385	1.00	1.07
79		56	80	0.0	0.1180	1.00	1.09
80		57	78	0.0	0.3206	1.00	1.16
81		58	77	0.0	0.1734	1.00	1.14
82		74	54	0.0	0.1948	1.00	0.97

T R A N S F C R M E R I N F O R M T I O N :

TX #	TX NAME	S.E.	R.E.	LEAKAGE IMPEDANCE		S.E. TAP	R.E. TAP
83		33	34	0.0	0.0800	1.00	1.02
84		35	73	0.0	0.2993	1.00	0.97
85		33	51	0.0	0.0525	1.00	1.02
86		33	35	0.0	0.0188	1.00	1.03
87		71	53	0.0	0.4437	1.00	0.99
88		72	52	0.0	0.3191	1.00	0.99
89		22	29	0.0	0.0304	1.00	1.00
90		29	69	0.0	0.1112	1.00	1.05
91		23	30	0.0	0.0309	1.00	1.00
92		21	27	0.0	0.0648	1.00	1.00
93		29	70	0.0	0.0715	1.00	1.06
94		29	64	0.0	0.0717	1.00	0.99
95		30	65	0.0	0.0680	1.00	0.93
327		114	115	0.0	0.0216	1.00	1.00

****

L I N E I N F O R M A T I O N :
++++++++++++++++++++++++++++++

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
1		21	22	0.0028	0.0239	0.51290	0.51290	F
2		22	23	0.0031	0.0281	0.84650	0.84650	F
3		23	24	0.0030	0.0410	0.78110	2.00560	F
4		24	25	0.0004	0.0056	0.19340	0.19340	F
5		32	109	0.0023	0.0218	0.20660	0.20660	F
6		33	36	0.0067	0.0653	0.63040	0.63040	F
7		36	74	0.0003	0.0031	0.02710	0.02710	F
8		35	37	0.0221	0.0768	0.11510	0.11510	F
9		38	39	0.0008	0.0048	0.00440	0.00440	F
10		31	46	0.0032	0.0195	0.01890	0.01890	F
11		31	90	0.0026	0.0158	0.01540	0.01540	F
12		31	47	0.0055	0.0330	0.03190	0.03190	F
13		37	39	0.0109	0.0651	0.05750	0.05750	F
14		38	40	0.0021	0.0131	0.01290	0.01290	F
15		37	75	0.0030	0.0190	0.01700	0.01700	F
16		37	97	0.0083	0.0573	0.05560	0.05560	F
17		40	43	0.0035	0.0240	0.02320	0.02320	F
18		41	43	0.0014	0.0087	0.08560	0.08560	F
19		42	43	0.0002	0.0017	0.00140	0.00140	F
21		46	96	0.0018	0.0052	0.36700	0.36700	F
22		31	42	0.0013	0.0079	0.00770	0.00770	F
23		40	41	0.0015	0.0096	0.08770	0.08770	F
24		39	44	0.0004	0.0028	0.19730	0.19730	F
25		39	40	0.0024	0.0155	0.01520	0.01520	F
26		44	45	0.0005	0.0028	0.20620	0.20620	F
27		30	35	0.0102	0.0704	0.06800	0.06800	F
28		30	49	0.0203	0.1401	0.13580	0.13580	F
29		29	113	0.0122	0.0849	0.08220	0.08220	F
30		29	68	0.0077	0.1046	0.20650	0.20650	F
31		82	83	0.0111	0.0417	0.02380	0.02380	F
32		83	84	0.0107	0.0394	0.02280	0.02280	F
33		84	85	0.0113	0.0417	0.00140	0.00140	F
34		76	85	0.0967	0.2352	0.00140	0.00140	F
35		79	82	0.0124	0.0481	0.00120	0.00120	F
36		85	87	0.0452	0.1679	0.00110	0.00110	F
37		79	86	0.0356	0.1255	0.00080	0.00080	F
38		86	87	0.0089	0.0347	0.00020	0.00020	F
39		80	81	0.1233	0.2051	0.00100	0.00100	F
40		81	82	0.0543	0.0980	0.00050	0.00050	F
41		79	80	0.0632	0.1151	0.00200	0.00200	F
42		76	79	0.0869	0.2080	0.00130	0.00130	F
43		78	79	0.0535	0.1697	0.00850	0.00850	F
44		77	78	0.0226	0.0682	0.00040	0.00040	F

L I N E I N F O R M A T I O N :

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
45		76	77	0.1539	0.4232	0.00280	0.00280	F
46		72	73	0.1457	0.2555	0.00220	0.00220	F
47		71	73	0.2248	0.6099	0.00480	0.00480	F
48		61	88	-15.3670	36.6340	0.0	0.0	F
49		47	48	0.0029	0.0178	0.00570	0.00570	F
50		48	60	-0.2625	0.4722	0.0	0.0	F
51		62	89	-21.4680	16.0660	0.0	0.0	F
52		60	88	-0.0732	0.1962	0.0	0.0	F
53		88	89	-0.8718	2.4920	0.0	0.0	F
54		48	59	-0.0174	0.2817	0.0	0.0	F
55		59	88	0.0038	0.0348	0.0	0.0	F
56		48	89	-0.0131	0.1389	0.0	0.0	F
57		62	88	-0.0207	0.1314	0.0	0.0	F
58		59	60	-0.0096	0.0388	0.0	0.0	F
59		60	62	-1.0163	0.7056	0.0	0.0	F
60		59	89	-0.0086	0.0509	0.0	0.0	F
61		48	61	-0.0181	0.2965	0.0	0.0	F
62		61	89	-0.0397	0.1729	0.0	0.0	F
227		25	89	0.0017	0.0363	0.88600	0.88600	F
65		59	61	-0.0695	0.3615	0.0	0.0	F
66		60	89	-0.0144	0.0303	0.0	0.0	F
67		60	61	-0.3306	0.4649	0.0	0.0	F
201		21	22	0.0028	0.0239	0.51290	0.51290	F
202		22	23	0.0031	0.0281	0.84650	0.84650	F
203		35	37	0.0104	0.0592	0.15280	0.15280	F
204		31	90	0.0027	0.0137	0.01950	0.01950	F
205		31	90	0.0026	0.0158	0.01520	0.01520	F
206		30	35	0.0100	0.0690	0.06670	0.06670	F
207		82	83	0.0113	0.0425	0.02420	0.02420	F
208		76	77	0.1537	0.4226	0.00280	0.00280	F
209		29	95	0.0038	0.0267	0.02590	0.02590	F
210		41	96	0.0012	0.0080	0.68700	0.68700	F
211		31	45	0.0033	0.0206	0.07320	0.07320	F
212		38	97	0.0030	0.0194	0.01890	0.01890	F
213		23	24	0.0027	0.0363	0.48220	1.70670	F
214		99	100	0.5305	0.7527	0.00380	0.00380	F
215		100	101	0.0789	0.1183	0.00060	0.00060	F
216		101	103	0.0042	0.0164	0.00010	0.00010	F
217		103	104	0.0164	0.0642	0.00050	0.00050	F
218		101	106	0.0236	0.0786	0.00060	0.00060	F
219		104	105	0.0128	0.0498	0.00030	0.00030	F
220		105	102	0.0250	0.0922	0.00060	0.00060	F
221		106	107	0.0095	0.0350	0.00020	0.00020	F

LINE INFORMATION:

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
222		107	108	0.0143	0.0529	0.00030	0.00030	F
223		108	102	0.0082	0.0322	0.00020	0.00020	F
224		108	102	0.0088	0.0343	0.00020	0.00020	F
225		48	88	-9.7693	35.6993	0.0	0.0	F
226		59	62	-2.4809	8.9636	0.0	0.0	F
63		25	89	0.0017	0.0363	0.88600	0.88600	F
228		49	113	0.0114	0.0794	0.07700	0.07700	F
229		30	112	0.0115	0.0800	0.07760	0.07760	F
230		46	90	0.0023	0.0146	0.01440	0.01440	F
231		36	109	0.0005	0.0052	0.04900	0.04900	F
232		47	114	0.0012	0.0103	0.01050	0.01050	F
233		47	114	0.0011	0.0097	0.00980	0.00980	F
234		60	270	0.0	0.0100	0.0	0.0	F

ALG.=TEST DATA=BCH.138 START=F 09:25 P.M. JULY 24, 1977 PAGE 1

****

B U S I N F O R M A T I O N :

++++++++++++++++++++++++++++

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
21	GMS500	1	0.0	0.0	0.0	0.0	500.00	0.0
22	WSN500	1	0.0	0.0	0.0	0.0	500.00	0.0
23	KLY500	1	0.0	0.0	0.0	0.0	500.00	0.0
24	ING500	1	0.0	0.0	0.0	0.0	500.00	0.0
25	CTR500	1	0.0	0.0	0.0	0.0	500.00	0.0
27	GMS138	1	0.0	0.0	96.80	41.70	138.00	0.0
28	GMS 1	2	261.00	0.0	0.0	0.0	13.80	14.15
29	WSN230	1	0.0	0.0	0.0	0.0	230.00	0.0
30	KLY230	1	0.0	0.0	0.0	0.0	230.00	0.0
31	ING230	2	0.0	0.0	0.0	0.0	230.00	241.00
32	ING360	1	0.0	0.0	0.0	0.0	360.00	0.0
33	BRT360	1	0.0	0.0	0.0	0.0	360.00	0.0
34	BR1 13	2	208.00	0.0	0.0	0.0	13.80	14.15
35	BRT230	1	0.0	0.0	0.0	0.0	230.00	0.0
36	ROS360	1	0.0	0.0	0.0	0.0	360.00	0.0
37	CKY230	1	0.0	0.0	165.20	47.60	230.00	0.0
38	WLT230	1	0.0	0.0	0.0	0.0	230.00	0.0
39	HPN230	1	0.0	0.0	0.0	0.0	230.00	0.0
40	BUT230	1	0.0	0.0	0.0	0.0	230.00	0.0
41	NEL230	1	0.0	0.0	102.50	49.60	230.00	0.0
42	PTM230	1	0.0	0.0	0.0	0.0	230.00	0.0
43	WHY230	1	0.0	0.0	66.10	21.70	230.00	0.0
44	MUR230	1	0.0	0.0	141.90	68.70	230.00	0.0
45	MAN230	1	0.0	0.0	136.20	65.90	230.00	0.0
46	KI2230	1	0.0	0.0	0.0	0.0	230.00	0.0
47	MER500	1	0.0	0.0	0.0	0.0	500.00	0.0
48	MER230	1	0.0	0.0	0.0	0.0	230.00	0.0
49	SCK230	1	0.0	0.0	62.40	20.70	60.00	0.0
50	GMS 2	2	1044.00	0.0	0.0	0.0	13.80	14.15
51	BR2 13	2	306.00	0.0	0.0	0.0	13.80	14.15
52	SCN 13	2	42.00	0.0	0.0	0.0	13.80	13.80
53	LAJ 13	2	22.00	0.0	0.0	0.0	13.80	13.80
54	WAH 13	2	60.00	0.0	19.70	9.50	13.80	13.80
55	CMS 13	2	140.00	0.0	0.0	0.0	13.80	14.15
56	BUN 13	2	50.00	0.0	0.0	0.0	13.80	13.80
57	SFL 4	2	52.50	0.0	0.0	0.0	4.40	4.40
58	RUS 13	2	105.60	0.0	0.0	0.0	13.80	13.80
63	WSNSYN	2	0.0	0.0	0.0	0.0	12.30	12.60
64	WSNSC4	2	0.0	0.0	0.0	0.0	12.50	12.50
65	KLYSC2	2	0.0	0.0	0.0	0.0	12.50	12.50
66	PTM 13	2	0.0	0.0	0.0	0.0	13.80	14.15
67	BUT 16	2	178.00	0.0	0.0	0.0	16.50	16.50

B U S I N F C R M A T I O N :

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
68	GLN138	1	0.0	0.0	161.90	50.90	13.80	0.0
69	WSN 66	1	0.0	0.0	249.50	89.20	66.00	0.0
70	WSNRX2	1	0.0	0.0	0.0	0.0	12.00	0.0
71	LAJ 66	1	0.0	0.0	3.50	1.70	66.00	0.0
72	SCN 66	1	0.0	0.0	15.60	7.50	66.00	0.0
73	BRT 66	1	0.0	0.0	0.40	0.20	66.00	0.0
74	WAH360	1	0.0	0.0	0.0	0.0	360.00	0.0
75	CMS230	1	0.0	0.0	0.0	0.0	230.00	0.0
76	ING 60	1	0.0	0.0	164.50	57.40	60.00	0.0
77	RUS 60	1	0.0	0.0	33.10	12.30	60.00	0.0
78	SFL 60	1	0.0	0.0	39.30	19.10	60.00	0.0
79	BND 60	1	0.0	0.0	188.00	87.30	60.00	0.0
80	LBN 60	1	0.0	0.0	5.20	2.50	60.00	0.0
81	SHL 60	1	0.0	0.0	5.70	1.90	60.00	0.0
82	HPN 60	1	0.0	0.0	107.50	52.10	60.00	0.0
83	MUR 60	1	0.0	0.0	152.40	73.80	60.00	0.0
84	SPG 60	1	0.0	0.0	119.40	57.80	60.00	0.0
85	KI1 60	1	0.0	0.0	215.10	94.50	60.00	0.0
86	NWR 60	1	0.0	0.0	25.50	10.80	60.00	0.0
87	RYL 60	1	0.0	0.0	25.60	10.90	60.00	0.0
90	ARN230	1	0.0	0.0	312.00	170.00	230.00	0.0
91	ARN132	1	0.0	0.0	200.00	0.0	132.00	0.0
92	ARN 60	1	0.0	0.0	55.00	18.10	60.00	0.0
95	SVY230	1	0.0	0.0	63.10	22.50	230.00	0.0
96	CSN230	1	0.0	0.0	57.50	27.90	230.00	0.0
97	CYP230	1	0.0	0.0	0.0	0.0	230.00	0.0
98	KLY RY	1	0.0	0.0	0.0	0.0	12.00	0.0
99	CKY 60	1	0.0	0.0	59.80	21.40	60.00	0.0
100	HSB 60	1	0.0	0.0	14.50	7.00	60.00	0.0
101	CYP 60	1	0.0	0.0	0.0	0.0	60.00	0.0
102	WLT 60	1	0.0	0.0	135.50	33.30	60.00	0.0
103	HCT 60	1	0.0	0.0	7.30	3.60	60.00	0.0
104	GLR 60	1	0.0	0.0	12.40	6.00	60.00	0.0
105	CAP 60	1	0.0	0.0	32.60	15.80	60.00	0.0
106	JLN 60	1	0.0	0.0	18.60	9.00	60.00	0.0
107	NOR 60	1	0.0	0.0	31.90	15.40	60.00	0.0
108	NVR 60	1	0.0	0.0	66.10	30.50	60.00	0.0
109	ATZ360	1	0.0	0.0	104.20	34.20	360.00	0.0
110	WSNØ73	1	0.0	0.0	0.0	0.0	12.00	0.0
111	GMS 3	2	825.00	0.0	0.0	0.0	13.80	14.15
112	HMH230	1	0.0	0.0	22.70	10.80	230.00	0.0
113	BLW230	1	0.0	0.0	42.20	20.40	230.00	0.0

B U S I N F O R M A T I O N :

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
116	SVA230	1	0.0	0.0	0.0	0.0	230.00	0.0
117	SVA138	1	0.0	0.0	7.60	2.50	138.00	0.0
118	KWD138	1	0.0	0.0	0.0	0.0	138.00	0.0
119	DUG138	1	0.0	0.0	29.00	9.50	138.00	0.0
120	VVW138	1	0.0	0.0	111.10	36.50	138.00	0.0
121	SAM138	1	0.0	0.0	51.40	16.90	138.00	0.0
122	VNT138	1	0.0	0.0	91.30	30.00	138.00	0.0
123	WSK138	1	0.0	0.0	11.40	3.80	138.00	0.0
124	WSH138	1	0.0	0.0	0.0	0.0	138.00	0.0
125	WSH 13	2	57.60	0.0	0.0	0.0	13.80	14.50
126	SLN138	1	0.0	0.0	0.0	0.0	138.00	0.0
127	SLN 7	2	198.00	0.0	0.0	0.0	7.20	7.20
128	BEL138	1	0.0	0.0	0.0	0.0	138.00	0.0
132	HLD138	1	0.0	0.0	206.90	45.90	138.00	0.0
133	NIC138	1	0.0	0.0	74.10	19.60	138.00	0.0
134	NIC230	1	0.0	0.0	0.0	0.0	230.00	0.0
135	NIC500	1	0.0	0.0	0.0	0.0	500.00	0.0
136	MCA500	1	0.0	0.0	0.0	0.0	500.00	0.0
137	MCA 1	2	300.00	0.0	0.0	0.0	13.80	13.80
138	VNT230	1	0.0	0.0	0.0	0.0	230.00	0.0
139	KCN230	1	0.0	0.0	0.0	0.0	230.00	0.0
140	KCN 13	2	375.00	0.0	0.0	0.0	13.80	13.80
141	SMI230	1	0.0	0.0	0.0	0.0	230.00	0.0
142	BEL161	1	0.0	0.0	0.0	0.0	161.00	0.0
143	BLT 13	2	120.00	0.0	0.0	0.0	13.80	13.80
144	BLT 69	1	0.0	0.0	35.00	15.00	69.00	0.0
145	CLV161	1	0.0	0.0	0.0	0.0	161.00	0.0
147	SSL 69	1	0.0	0.0	172.00	31.00	69.00	0.0
148	TRL161	1	0.0	0.0	0.0	0.0	161.00	0.0
149	TRL 69	1	0.0	0.0	500.00	210.00	69.00	0.0
150	WAN 14	2	360.00	0.0	0.0	0.0	14.40	14.40
151	WAN230	1	0.0	0.0	0.0	0.0	230.00	0.0
152	WAN 69	1	0.0	0.0	0.0	0.0	69.00	0.0
153	GLN500	1	0.0	0.0	0.0	0.0	500.00	0.0
154	CBK230	2	0.0	0.0	234.70	84.70	230.00	207.00
155	GMS 9	2	300.00	0.0	0.0	0.0	13.80	14.15
156	MCA 2	2	300.00	0.0	0.0	0.0	13.80	13.80
251	CST230	2	12.70	6.20	1507.14	130.63	230.00	237.68
252	CST500	2	0.0	0.0	5.56	-42.19	500.00	512.00
253	MNR500	2	192.50	184.50	3460.70	-790.68	500.00	500.00
254	JGC230	2	3497.10	507.50	2266.02	30.26	230.00	240.67
255	BEL230	2	104.60	193.20	2144.87	-250.98	230.00	236.00

B U S I N F O R M A T I O N :

BUS #	BUS NAME	TYPE	GENERATION MW	GENERATION MVAR	LOAD MW	LOAD MVAR	BASE KV	VOLTAGE KV
256	BDY230	2	350.00	69.60	0.0	28.97	230.00	241.96
257	DIABLO	2	836.60	398.40	504.36	-87.82	100.00	114.53
258	NCXON	2	1194.10	792.70	836.65	-606.42	100.00	111.10
259	TACOMA	2	206.90	382.40	481.93	-272.00	100.00	140.13
260	UCOLBA	3	3168.60	1660.70	1840.98	-854.71	100.00	107.80
261	HANFRD	2	720.00	434.80	633.40	-208.40	100.00	111.85
262	LNGVEW	2	923.40	545.20	1154.21	-519.86	100.00	116.04
263	LCOBLA	2	2694.20	1703.40	2667.74	-1396.00	100.00	110.46
264	J DAY	2	1752.90	1338.80	1340.19	-1001.40	100.00	107.91
265	LSNAKE	2	1156.10	593.30	751.06	-322.54	100.00	107.61
266	USNAKE	2	1169.30	980.80	1077.48	-655.87	100.00	116.81
267	WOREGN	2	1013.80	1037.70	1515.54	-1074.70	100.00	120.98

T R A N S F O R M E R I N F C R M T I O N :

TX #	TX NAME	S.E.	R.E.	LEAKAGE IMPEDANCE		S.E. TAP	R.E. TAP
68		44	83	0.0	0.1335	1.00	1.08
69		255	251	0.0	0.0117	1.00	1.00
70		75	55	0.0	0.0835	1.00	1.00
71		40	67	0.0	0.0382	1.05	1.00
72		42	66	0.0	0.1137	1.04	1.00
73		44	83	0.0	0.1333	1.00	1.08
74		46	85	0.0	0.0768	1.00	1.07
75		31	76	0.0	0.1108	1.00	1.05
76		32	31	0.0	0.0158	0.96	1.03
77		24	31	0.0	0.0193	1.02	1.05
78		39	82	0.0	0.2385	1.00	1.10
79		56	80	0.0	0.1180	0.96	1.05
80		57	78	0.0	0.3206	0.94	1.10
81		58	77	0.0	0.1734	0.96	1.09
82		74	54	0.0	0.1948	0.98	0.96
83		33	34	0.0	0.0800	0.98	1.00
84		35	73	0.0	0.2993	0.99	0.95
85		33	51	0.0	0.0525	0.98	1.00
86		33	35	0.0	0.0188	1.01	1.03
87		71	53	0.0	0.4437	0.97	0.96
88		72	52	0.0	0.3191	0.97	0.96
89		22	29	0.0	0.0304	1.05	1.05
90		29	69	0.0	0.1112	1.00	1.05
91		23	30	0.0	0.0309	1.05	1.05
92		21	27	0.0	0.0648	1.00	1.00
93		29	70	0.0	0.0715	1.00	1.06
94		29	64	0.0	0.0717	1.00	0.99
95		30	65	0.0	0.0680	1.00	0.93
96		21	28	0.0	0.0558	1.00	0.98
97		21	50	0.0	0.0143	1.00	0.98
98		69	63	0.0	0.0959	1.00	0.93
99		90	91	0.0	0.0	1.00	1.05
100		90	92	0.0	0.2253	1.00	1.02
301		46	85	0.0	0.0714	1.00	1.03
302		46	85	0.0	0.0768	1.00	1.07
303		31	76	0.0	0.1108	1.00	1.05
304		24	31	0.0	0.0193	1.00	1.03
305		39	82	0.0	0.2280	1.00	1.10
306		39	82	0.0	0.2280	1.00	1.10
307		22	29	0.0	0.0305	1.00	1.00
308		29	69	0.0	0.1112	1.00	1.05
309		29	69	0.0	0.1132	1.00	1.05
310		21	27	0.0	0.0648	1.00	1.00

T R A N S F O R M E R I N F O R M T I O N :

TX #	TX NAME	S.E.	R.E.	LEAKAGE IMPEDANCE		S.E. TAP	R.E. TAP
311		90	91	0.0	0.0500	1.00	1.05
312		90	91	0.0	0.0500	1.00	1.05
313		90	91	0.0	0.9550	1.00	1.05
314		23	30	0.0	0.0309	1.00	1.00
315		30	98	0.0	0.0720	1.00	0.97
316		37	99	0.0	0.4285	1.00	0.97
317		37	99	0.0	0.4110	1.00	0.97
318		101	97	0.0	0.1147	1.00	0.91
319		38	102	0.0	0.0404	1.00	0.97
321		101	97	0.0	0.1147	1.00	0.91
323		22	29	0.0	0.0300	1.00	1.00
324		21	111	0.0	0.0186	1.00	0.98
326		29	110	0.0	0.0715	1.00	1.06
344		153	68	0.0	0.0300	1.00	1.03
328		116	117	0.0	0.0233	1.00	1.04
329		124	125	0.0	0.2343	1.00	1.00
330		147	126	0.0	0.1010	1.00	1.09
343		156	136	0.0	0.0073	1.00	1.02
332		24	31	0.0	0.0193	1.00	1.03
348		47	48	0.0	0.0129	1.00	1.03
349		47	48	0.0	0.0129	1.00	1.03
353		21	155	0.0	0.0465	1.00	0.98
333		135	134	0.0	0.0151	1.00	1.03
334		134	133	0.0	0.0450	1.00	1.00
335		139	140	0.0	0.0310	1.00	0.97
336		138	122	0.0	0.0450	1.00	1.00
337		137	136	0.0	0.0073	1.00	1.02
338		142	128	0.0	0.1480	1.00	0.96
339		143	144	0.0	0.0439	1.00	1.05
340		127	147	0.0	0.0167	1.00	1.04
341		148	149	0.0	0.1540	1.00	0.93
342		150	152	0.0	0.0181	1.00	1.09

LINE INFORMATION:

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
1		21	22	0.0028	0.0179	0.51290	0.51290	F
2		22	23	0.0031	0.0211	0.84650	0.84650	F
3		23	24	0.0030	0.0205	0.78110	2.00560	F
4		24	25	0.0004	0.0056	0.19340	0.19340	F
5		32	109	0.0023	0.0218	0.20660	0.20660	F
6		33	36	0.0067	0.0653	0.63040	0.63040	F
7		36	74	0.0003	0.0031	0.02710	0.02710	F
8		35	37	0.0221	0.0768	0.11510	0.11510	F
9		38	39	0.0008	0.0048	0.00440	0.00440	F
10		31	46	0.0032	0.0195	0.01890	0.01890	F
11		31	90	0.0026	0.0158	0.01540	0.01540	F
12		24	25	0.0004	0.0047	0.22700	0.22700	F
13		37	38	0.0095	0.0573	0.05610	0.05610	F
14		38	48	0.0037	0.0236	0.02320	0.02320	F
15		37	75	0.0030	0.0190	0.01700	0.01700	F
16		37	97	0.0083	0.0573	0.05560	0.05560	F
17		42	43	0.0004	0.0022	0.00220	0.00220	F
18		42	48	0.0022	0.0153	0.01470	0.01470	F
19		41	48	0.0035	0.0218	0.09870	0.09870	F
21		46	96	0.0018	0.0052	0.36700	0.36700	F
22		31	43	0.0009	0.0056	0.00550	0.00550	F
23		40	41	0.0015	0.0096	0.08770	0.24970	F
24		44	48	0.0048	0.0303	0.25880	0.25880	F
25		39	40	0.0024	0.0155	0.01520	0.01520	F
26		44	45	0.0005	0.0028	0.20620	0.36820	F
27		30	35	0.0102	0.0704	0.06800	0.06800	F
28		30	49	0.0203	0.1401	0.13580	0.14300	F
29		29	113	0.0122	0.0849	0.08220	0.08220	F
30		22	153	0.0016	0.0217	-.20710	1.01740	F
31		82	83	0.0111	0.0417	0.21580	0.02380	F
32		83	84	0.0107	0.0394	0.30780	0.02280	F
33		84	85	0.0113	0.0417	0.10340	0.00140	F
34		76	85	0.0967	0.2352	0.00140	0.00140	F
35		79	82	0.0124	0.0481	0.00120	0.00120	F
36		85	87	0.0452	0.1679	0.00110	0.00110	F
37		79	86	0.0356	0.1255	0.00080	0.00080	F
38		86	87	0.0089	0.0347	0.00020	0.00020	F
39		80	81	0.1233	0.2051	0.00100	0.00100	F
40		81	82	0.0543	0.0980	0.00050	0.00050	F
41		79	80	0.0632	0.1151	0.00200	0.00200	F
42		76	79	0.0869	0.2080	0.00130	0.00130	F
43		78	79	0.0535	0.1697	0.00850	0.00850	F
44		77	78	0.0226	0.0682	0.00040	0.00040	F

L I N E I N F O R M A T I O N :

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
45		76	77	0.1539	0.4232	0.00280	0.00280	F
46		72	73	0.1457	0.2555	0.00220	0.00220	F
47		71	73	0.2248	0.6099	0.00480	0.00480	F
909		266	267	-2.9023	2.8004	0.0	0.0	F
908		265	267	-0.5588	0.8411	0.0	0.0	F
50		24	47	0.0002	0.0030	0.14260	0.14260	F
51		40	48	0.0016	0.0105	0.01030	0.01030	F
52		40	48	0.0016	0.0105	0.01030	0.01030	F
907		265	266	-0.6550	1.5140	0.0	0.0	F
906		264	267	-0.1943	0.3147	0.0	0.0	F
905		264	266	-2.2094	2.3464	0.0	0.0	F
904		264	265	-0.1123	0.3960	0.0	0.0	F
903		263	267	-0.1863	0.2326	0.0	0.0	F
902		263	266	-0.1795	0.4398	0.0	0.0	F
901		263	265	-0.0890	0.2390	0.0	0.0	F
900		263	264	-0.0941	0.1697	0.0	0.0	F
899		262	267	-0.8873	0.8362	0.0	0.0	F
898		262	266	-4.6897	4.7822	0.0	0.0	F
897		262	265	-1.1300	1.5223	0.0	0.0	F
896		262	264	-0.5653	0.7038	0.0	0.0	F
895		262	263	-0.2321	0.3430	0.0	0.0	F
894		261	267	-0.5967	0.9315	0.0	0.0	F
201		21	22	0.0028	0.0179	0.51290	0.51290	F
202		22	23	0.0031	0.0211	0.84650	0.84650	F
203		35	37	0.0104	0.0592	0.15280	0.15280	F
204		31	90	0.0027	0.0137	0.01950	0.68950	F
205		31	90	0.0026	0.0158	0.01520	0.01520	F
206		30	35	0.0100	0.0690	0.06670	0.06670	F
207		82	83	0.0113	0.0425	0.02420	0.02420	F
208		76	77	0.1537	0.4226	0.00280	0.00280	F
209		29	95	0.0038	0.0267	0.02590	0.02590	F
210		41	96	0.0012	0.0080	0.68700	0.68700	F
211		31	45	0.0033	0.0206	0.07320	0.07320	F
212		38	97	0.0030	0.0194	0.01890	0.01890	F
213		23	47	0.0026	0.0344	0.38940	1.61390	F
214		99	100	0.5305	0.7527	0.00380	0.00380	F
215		100	101	0.0789	0.1183	0.00060	0.00060	F
216		101	103	0.0042	0.0164	0.00010	0.00010	F
217		103	104	0.0164	0.0642	0.00050	0.00050	F
218		101	106	0.0236	0.0786	0.00060	0.00060	F
219		104	105	0.0128	0.0498	0.00030	0.00030	F
220		105	102	0.0250	0.0922	0.00060	0.00060	F
221		106	107	0.0095	0.0350	0.00020	0.00020	F

L I N E I N F O R M A T I O N :

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
222		107	108	0.0143	0.0529	0.19230	0.00030	F
223		108	109	0.0082	0.0322	0.10220	0.00020	F
224		108	102	0.0088	0.0343	0.00020	0.00020	F
893		261	266	-2.0281	4.0087	0.0	0.0	F
892		261	265	-0.0501	0.4020	0.0	0.0	F
891		261	264	-0.0991	0.3906	0.0	0.0	F
228		49	113	0.0114	0.0794	0.07700	0.07700	F
229		30	112	0.0115	0.0800	0.07760	0.07760	F
230		46	90	0.0023	0.0146	0.01440	0.01440	F
231		36	109	0.0005	0.0052	0.04900	0.04900	F
890		261	263	-0.2277	0.4451	0.0	0.0	F
889		261	262	-1.1985	1.6585	0.0	0.0	F
888		260	267	-0.5780	0.6528	0.0	0.0	F
234		141	154	0.0248	0.1623	0.15000	0.15000	F
235		30	116	0.0119	0.0822	0.08000	0.03000	F
236		30	116	0.0121	0.0834	0.08120	0.08120	F
237		117	118	0.0246	0.0886	0.01130	0.01130	F
238		117	119	0.0276	0.0992	0.01270	0.01270	F
239		118	120	0.0139	0.0498	0.00630	0.00630	F
240		119	120	0.0036	0.0131	0.15170	0.00170	F
241		120	121	0.1118	0.2918	0.03520	0.03520	F
242		121	122	0.0537	0.1397	0.01670	0.01670	F
243		120	122	0.1136	0.2964	0.03560	0.03560	F
244		122	123	0.1020	0.2663	0.08290	0.03190	F
245		122	123	0.1020	0.2663	0.03190	0.03190	F
246		123	126	0.0674	0.2048	0.02500	0.02500	F
247		123	124	0.0055	0.0166	0.00200	0.00200	F
248		122	128	0.0496	0.1511	0.01860	0.01860	F
887		260	266	-0.4103	1.1718	0.0	0.0	F
886		260	265	-0.1065	0.4018	0.0	0.0	F
885		260	264	-0.1749	0.3764	0.0	0.0	F
251		117	132	0.0295	0.1062	0.01350	0.01350	F
252		117	132	0.0526	0.1043	0.01370	0.01370	F
253		117	132	0.0157	0.1091	0.01370	0.01370	F
254		132	133	0.0374	0.1348	0.01710	0.01710	F
255		134	138	0.0156	0.1025	0.09420	0.09420	F
256		138	139	0.0276	0.1813	0.16680	0.16680	F
257		139	141	0.0079	0.0518	0.04770	0.04770	F
258		141	256	0.0019	0.0125	0.01220	0.01220	F
259		135	136	0.0026	0.0355	1.78620	0.56170	F
260		24	135	0.0025	0.0332	1.67260	0.44810	F
261		122	128	0.1197	0.3397	0.08920	0.08920	F
262		142	145	0.0666	0.2680	0.06270	0.06270	F

LINE INFORMATION:

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
263		144	149	0.0244	0.0857	0.00700	0.00700	F
264		145	148	0.0364	0.1460	0.03430	0.03430	F
265		147	149	0.0229	0.0804	0.01820	0.01820	F
266		149	152	0.0055	0.0323	1.25560	0.00560	F
267		151	256	0.0021	0.0214	0.02180	0.02180	F
271		40	44	0.0032	0.0203	0.22390	0.22390	F
801		251	252	0.0004	0.0098	0.0	0.0	F
802		251	253	-0.0227	0.1683	0.0	0.0	F
803		251	254	-0.0422	0.3900	0.0	0.0	F
804		251	255	-20.0650	59.5974	0.0	0.0	F
806		251	257	-0.0700	0.4537	0.0	0.0	F
807		251	258	-102.5900	182.2950	0.0	0.0	F
808		251	259	-2.8125	5.7180	0.0	0.0	F
809		251	260	-0.5662	1.4469	0.0	0.0	F
810		251	261	-4.5883	8.5837	0.0	0.0	F
811		251	262	-3.0584	4.4074	0.0	0.0	F
812		251	263	-4.3832	4.0890	0.0	0.0	F
813		251	264	-6.0393	6.6411	0.0	0.0	F
814		251	265	-6.5004	10.6549	0.0	0.0	F
815		251	266	-35.8660	40.8093	0.0	0.0	F
816		251	267	-12.9350	8.3706	0.0	0.0	F
817		252	253	0.0012	0.0240	0.0	0.0	F
818		253	254	-0.0075	0.0479	0.0	0.0	F
819		253	255	-1.0302	3.0674	0.0	0.0	F
820		253	257	-0.0443	0.1832	0.0	0.0	F
821		253	258	-6.3519	9.9404	0.0	0.0	F
822		253	259	-0.2185	0.4428	0.0	0.0	F
823		253	260	-0.0220	0.0789	0.0	0.0	F
824		253	261	-0.0955	0.3380	0.0	0.0	F
825		253	262	-0.1424	0.2520	0.0	0.0	F
826		253	263	-0.1436	0.1915	0.0	0.0	F
827		253	264	-0.1667	0.2929	0.0	0.0	F
828		253	265	-0.1666	0.4468	0.0	0.0	F
829		253	266	-1.3147	1.9227	0.0	0.0	F
830		253	267	-0.4159	0.4190	0.0	0.0	F
831		254	255	0.0038	0.0350	0.0	0.0	F
832		254	257	-0.0892	0.4263	0.0	0.0	F
833		254	258	-0.3450	3.3338	0.0	0.0	F
834		254	259	-0.5214	1.1066	0.0	0.0	F
835		254	260	-0.0074	0.0752	0.0	0.0	F
836		254	261	-0.0960	0.5298	0.0	0.0	F
837		254	262	-0.2510	0.5082	0.0	0.0	F
838		254	263	-0.1070	0.2710	0.0	0.0	F

LINE INFORMATION:

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
839		254	264	-0.2294	0.5160	0.0	0.0	F
840		254	265	-0.1163	0.5713	0.0	0.0	F
841		254	266	-0.4783	1.7267	0.0	0.0	F
842		254	267	-0.6873	0.8224	0.0	0.0	F
843		255	256	0.0075	0.0665	0.0	0.0	F
844		255	257	-22.0660	50.7000	0.0	0.0	F
845		255	258	-0.0146	0.0997	0.0	0.0	F
846		255	259	-35.9370	58.8130	0.0	0.0	F
847		255	260	-0.1595	1.4119	0.0	0.0	F
848		255	261	-2.3240	7.3432	0.0	0.0	F
849		255	262	-7.9582	10.6685	0.0	0.0	F
850		255	263	-0.3532	1.0663	0.0	0.0	F
851		255	264	-3.8132	5.5421	0.0	0.0	F
852		255	265	-0.5188	2.4778	0.0	0.0	F
853		255	266	-0.1765	0.4482	0.0	0.0	F
854		255	267	-6.5988	7.0145	0.0	0.0	F
855		257	258	-105.6200	152.6340	0.0	0.0	F
856		257	259	-2.7575	4.7670	0.0	0.0	F
857		257	260	-0.5803	1.2263	0.0	0.0	F
858		257	261	-4.4030	7.1198	0.0	0.0	F
859		257	262	-2.8699	3.6185	0.0	0.0	F
860		257	263	-3.9949	3.2579	0.0	0.0	F
861		257	264	-5.5379	5.3706	0.0	0.0	F
862		257	265	-6.1980	8.7961	0.0	0.0	F
863		257	266	-33.4670	33.0671	0.0	0.0	F
864		257	267	-11.5410	6.4812	0.0	0.0	F
865		258	259	-168.8700	176.5590	0.0	0.0	F
866		258	260	-1.5420	4.3306	0.0	0.0	F
867		258	261	-22.0780	26.1083	0.0	0.0	F
868		258	262	-48.4210	29.7899	0.0	0.0	F
869		258	263	-3.3237	3.7931	0.0	0.0	F
870		258	264	-27.7870	8.6967	0.0	0.0	F
871		258	265	-5.5745	9.7386	0.0	0.0	F
872		258	266	-1.1027	0.7724	0.0	0.0	F
873		258	267	-40.7720	7.3724	0.0	0.0	F
874		259	260	-0.9370	1.5647	0.0	0.0	F
875		259	261	-5.4770	7.5895	0.0	0.0	F
876		259	262	-2.0136	2.8698	0.0	0.0	F
877		259	263	-4.1062	3.0298	0.0	0.0	F
878		259	264	-5.9510	5.1445	0.0	0.0	F
879		259	265	-7.5768	9.0937	0.0	0.0	F
880		259	266	-40.6000	32.8438	0.0	0.0	F
881		259	267	-11.5520	5.9102	0.0	0.0	F

L I N E I N F O R M A T I O N :

LINE #	LINE NAME	S.E.	R.E.	PER UNIT R	PER UNIT X	S.E. SUSEP.	R.E. SUSEP.	TX?
882		260	261	-0.0730	0.3250	0.0	0.0	F
883		260	262	-0.4514	0.6506	0.0	0.0	F
884		260	263	-0.1047	0.2208	0.0	0.0	F
99		90	91	0.0	0.0968	0.0	0.40600	T