C.1

A NEW FORMULATION OF THE LOAD-FLOW PROBLEM

by

HOSSEIN JALALI-KUSHKI

B.A.Sc., Arya-Mehr University of Technology, 1971

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in the Department

of

Electrical Engineering

We accept this thesis as conforming to the

required standard

THE UNIVERSITY OF BRITISH COLUMBIA

June 1973

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 Elec. Eng.

The University of British Columbia
Vancouver 8, Canada

Date June, 11, 1973

ABSTRACT

The new formulation of the load-flow problem presented in this thesis yields a set of equations each of which has only one nonlinear term. The equations are derived from the corrections required to make the final values equal to the initial estimated values. The resultant set of equations can be used when the initial estimated values are adjusted to their final values. However, derivation of the equations for this latter case results in a set of equations with (n-1) nonlinear terms in each equation for an n-bus power system. Five algorithms based upon the new formulation are described. Numerical tests on several sample power systems show that some of the new algorithms possess better convergence and speed characteristics than the commonly used Ward-Hale and Newton algorithms.

TABLE OF CONTENTS

LIST OF SYMBOLS

VOLTAGE

V_k	voltage of bus k
$\|V_k\|$	magnitude of V_k
θ_k	phase angle of V_k
e_k	real part of V_k
f_k	imaginary part of V_k
U_k	magnitude of voltage at (P-V) type bus k
$[V]$	voltage vector
ΔV_k	voltage correction at bus k
α_k	real part of ΔV_k
β_k	imaginary part of ΔV_k

CURRENT

I_k	current entering bus k
a_k	real part of I_k
b_k	imaginary part of I_k
$[I]$	current vector

ADMITTANCE

Y_{ik}	element (i,k) of the nodal admittance matrix
G_{ik}	real part of Y_{ik} (conductance)
B_{ik}	imaginary part of Y_{ik} (suseptance)
$\|Y_{ik}\|$	magnitude of Y_{ik}
δ_{ik}	phase angle of Y_{ik}

$[Y]$	nodal admittance matrix

POWER

S_k	power entering bus k
P_k	real power entering bus k
Q_k	reactive power entering bus k
ΔS_k^t	total change of power at bus k
$[S]$	power vector

JACOBIAN

H_{kj}	sensitivity of P_k with respect to θ_j
N_{kj}	sensitivity of P_k with respect to $\lvert V_j \rvert$ (normalized)
J_{kj}	sensitivity of Q_k with respect to θ_j
L_{kj}	sensitivity of Q_k with respect to $\lvert V_j \rvert$ (normalized)
D_{kj}	Jacobian sub-matrix

GENERAL

'	present assumed or calculated value
unprimed	final or scheduled value
*	complex conjugate
$\mathcal{R}$	real part of a complex variable
j	complex operator, $\sqrt{-1}$
n	total number of buses in the system
n_1	number of (P-Q) type buses in the system
n_2	number of (P-V) type buses in the system
k_1	set of all the (P-Q) type buses

k_2	set of all the (P-V) type buses
A_{ik}	real part of $V_i' \ Y_{ik}^*$
B_{ik}	imaginary part of $V_i' \ Y_{ik}^*$

ACKNOWLEDGMENT

I would like to thank my supervisor Dr. H.R.Cheann for his interest, advice and encouragement throughout the research and writing of this thesis.

I would also like to thank Dr. Y.N.Yu for proof-reading the thesis and offering further suggestions and also for financial support during the month of May.

I am also grateful to Misses Norma Duggan and Betty Cockburn for their help in typing the thesis .

The financial support of the University of British Columbia in the form of UBC graduate fellowship is acknowledged.

1. INTRODUCTION

Load-flow studies are required in the design of any power system. Such studies are also required to optimally dispatch power in the system and to find the initial conditions of the system for transient stability studies. Numerous load-flow studies of various system configurations have to be made in planning new systems and extending present systems.

The load-flow problem is to find the voltages of the various buses (nodes) of a power system such that certain constraints are satisfied. These constraints may require that a desired prescribed or scheduled value of voltage magnitude exist at some of the buses, or that prescribed or scheduled values of active, or real, and reactive powers exist at some of the other buses. Additional constraints in the form of an acceptable flow of reactive power in the system or an acceptable loading of the transmission lines may also be imposed. To satisfy these additional constraints repeated use of the basic load-flow computer program is necessary and different system configurations may have to be studied [1].

A load-flow program consists of three parts as follows:

1) The input part which assembles all the input data and forms the necessary matrices.

2) The procedure or the main part of a load-flow program which finds the bus voltages in some iterative process.

3) The output part which calculates the individual line power flows and prints out the results of the load-flow study.

In the input part, each bus of the power system is defined as one of the following types:

a) A "(P-Q) type bus" or "load bus" at which the real and reactive powers are scheduled.

b) A "(P-V) type bus", "generator bus" or "voltage controlled bus" at which the real power and the voltage magnitude are scheduled.

c) A "slack bus", "swing bus", "floating bus" or "reference bus" at which the voltage is scheduled both in magnitude and phase angle. Real and reactive powers are not specified for these buses. These buses compensate for the unknown transmission losses in the system. Each system has at least one slack bus but some systems have more than one. In this thesis we will assume that there is only one slack bus in the system. However, all the conclusions and results are valid even if there is more than one slack bus in the system since these buses do not increase the number of unknowns and the number of equations used in the computation.

Additional constraints, such as upper and lower bounds for the (P-Q) type bus voltage magnitudes and (P-V) type bus reactive powers, may also be given as part of the bus input data. Transmission lines are normally represented by their nominal Π circuits, as shown in Appendix A.

The procedure part of a load-flow program includes some kind of iterative numerical process. Algorithms differ in procedure but, in general, a set of initial values for the bus voltages is estimated and corrected in successive steps until

the final solution is achieved.

A favorable initial value is necessary for a successful solution. Normally the flat-start is used with all voltage magnitudes set to their scheduled values or, if not scheduled, to the magnitude of the slack bus and all angles set equal to zero. Sometimes the solution of a previous study may be used as the initial values for the study of a system configuration which is similar to the previous system.

The iterative part of an algorithm uses some stop criterion to determine if the solution is complete. Any of the following stop criteria or any combination of them may be used for this purpose:

1) The greatest voltage correction made in an iteration cycle should be less than a desired tolerance.

2) The magnitude of voltages and the power flows should be within a desired tolerance of their scheduled values.

3) The sum of the residuals of real and reactive powers of all the buses, which is called the slack bus power mismatch, should be less than a desired prescribed value.

The third stop criterion is used in all the algorithms described in this thesis.

Although some of the earlier load-flow programs used mesh equations, recent methods use nodal equations [1] because nodal equations are easier to form and modify and need less computer storage. Laughton and Humphrey Davies [1] divide existing load-flow algorithms into two categories, the "iterative

methods" and the "direct methods". The direct methods are those that solve a set of linear equations at each step of their iterative procedure. These methods require inversion of a matrix in one way or another. The iterative methods make use of the nodal admittance matrix in an iterative process and correct the bus voltages one by one. No matrix inversion is necessary. The iterative methods do not have the huge computer storage requirements of the direct methods.

In 1956 J.B.Ward and H.W.Hale [2] presented an algorithm based upon the Gauss-Seidel iterative procedure. Because of its simplicity and ability to converge for most problems this algorithm was commonly used until very recently. Use of different acceleration techniques with this algorithm were implemented by others [3,4,5]. The algorithm benefits most from the use of linear acceleration factors [3] and relaxation techniques [4]. In 1959, a new formulation of the load-flow problem was presented by Van Ness [6]. This formulation related small changes in real and reactive powers to the small changes in voltage magnitudes and phase angles by means of first order derivatives. Van Ness tried several iterative algorithms based upon the new formulation. These algorithms were essentially the Ward-Hale method used with polar co-ordinates. Later in 1961, Van Ness and Griffin [7] used the elimination method to solve the set of linear equations which related the small changes in power to the small changes in voltages. Nowadays this method is called Newton's method. Although use of the elimination process reduced, to a certain degree, the computer storage problem involved in inverting the matrix of coefficients, called the

Jacobian matrix, the method was not yet capable of handling large power systems and, therefore, was not practically useful. In 1967 Tinney and Walker [8] suggested several schemes for ordering the nodes of a network such that when the elimination process is used the number of non-zero terms accumulated in the upper triangular matrix will tend to be minimum and Tinney and Hart [9] suggested an algorithm that used the elimination process along with sub-optimal ordering of the buses. This made Newton's method applicable to large power systems and because of its faster and better convergence, this method soon replaced the Ward-Hale method. Other contributions have been made to Newton's method to make it more general and more efficient [10,11,12,13,14,15].

Use of the Hessian matrix instead of the Jacobian matrix has also been investigated [16] but so far it has not shown any significant improvement over the Newton's method.

In Chapter 2 of this thesis the Ward-Hale method and Newton's method are reviewed. In Chapter 3 a new set of load-flow equations is derived and several algorithms based upon this formulation are described. These new algorithms are compared with the previous methods for a number of sample power systems and the results are presented in Chapter 4.

2. PREVIOUS METHODS

2.1 The Ward-Hale Method

This method of solving the load-flow problem was first suggested by J.B.Ward and H.W.Hale in 1956 [2]. Further contributions to the original paper [3,4,5] made the algorithm faster and more efficient. This method was widely used until very recently.

The method makes use of the Gauss-Seidel iterative procedure which usually converges to a satisfactory final solution after a certain number of iterations. However, the number of iterations depends upon the size, configuration and parameters of the power system. It also depends upon the closeness of the initial estimate to the final solution. There are some cases where this method is unable to converge to the solutions which are known to exist.

The algorithm is simple and easy to program. It is also capable of handling very large power systems by using improved programming techniques and taking advantage of sparsity of the admittance matrix.

The Algorithm

Nodal equations are used to express the relationship between currents and voltages:

$$[I] = [Y]\,[V] \tag{2-1}$$

where [Y] is the nodal admittance matrix. This matrix is formed easily if transmission lines and transformers are represented by their Π equivalents. Appendix A shows how the Π equivalent circuits of transmission lines and transformers are found and how [Y] is formed.

The iterative procedure starts once [Y] is formed. With primed symbols referring to the present values at each iteration step and unprimed symbols to the final or scheduled values, the procedure is as follows:

Step 1) An initial estimate for the voltages is made. A favorable starting approximation is necessary for a successful solution. If the initial estimate is too far from the final solution the method may diverge or the number of iterations will be prohibitive. Normally, the flat-start is sufficient.

Step 2) Using equation (2-1), the current entering any bus k can be calculated:

$$I'_k = \sum_{j=1}^{n} Y_{kj}\, V'_j \tag{2-2}$$

where n is the number of buses.

The power entering bus k can then be calculated:

$$S'_k = V'_k\, I'^{*}_k \tag{2-3}$$

Obviously if [V'] is not equal to [V], then S'_k will not be equal to the scheduled value for power, S_k.

Step 3) The correction to the voltage of bus k, ΔV_k, is

determined from the differences between calculated and scheduled powers. It is assumed that the voltages of all the other buses remain constant when this correction is calculated. ΔV_k should satisfy the following condition:

$$S_k = (V'_k + \Delta V_k)(I'_k + \Delta I_k)^* \qquad (2\text{-}4)$$

Because V'_k is the only voltage that changes we can write:

$$\Delta I_k = Y_{kk}\ \Delta V_k \qquad (2\text{-}5)$$

So (2-4) can be written as:

$$\Delta S_k = \Delta V_k\ I'^*_k + V'_k\ Y^*_{kk}\ \Delta V^*_k + \Delta V_k\ Y^*_{kk}\ \Delta V^*_k \qquad (2\text{-}6)$$

Assuming that ΔV_k is small, the last term on the right hand side of (2-6) can be neglected:

$$\Delta S_k = I'^*_k\ \Delta V_k + V'_k\ Y^*_{kk}\ \Delta V^*_k \qquad (2\text{-}7)$$

Equating real and imaginary parts in (2-7) we will have:

$$\begin{bmatrix} \Delta P_k \\ \Delta Q_k \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \cdot \begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix} \qquad (2\text{-}8)$$

where $\Delta V_k = \alpha_k + j\beta_k$ and m_1, m_2, m_3 and m_4 are functions of the real and imaginary parts of the known values, I'_k, Y_{kk}, V'_k as follows:

Let:

$$Y_{kk} \triangleq G_{kk} + j\, B_{kk}$$

$$I'_k \triangleq a_k + j\, b_k$$

and

$$V'_k \triangleq e_k + j\, f_k$$

then:

$$m_1 = a_k + e_k\, G_{kk} + f_k\, B_{kk}$$

$$m_2 = b_k - e_k\, B_{kk} + f_k\, G_{kk}$$

$$m_3 = -b_k + f_k\, G_{kk} - e_k\, B_{kk}$$

$$m_4 = a_k - f_k\, B_{kk} - e_k\, G_{kk}$$

For a (P-Q) type bus, (2-8) is sufficient for calculating corrections α_k and β_k. On the other hand, at a (P-V) type bus, the value of Q_k is not scheduled so that ΔQ_k is not specified, but the magnitude of V_k is to be held constant. This constraint is formulated as:

$$(V'_k + \Delta V_k)(V'_k + \Delta V_k)^* = |V_k|^2 \qquad (2\text{-}9)$$

or:

$$V'_k\, \Delta V_k^* + V_k'^*\, \Delta V_k + |\Delta V_k|^2 = |V_k|^2 - |V'_k|^2 \qquad (2\text{-}10)$$

Neglecting the second order term $|\Delta V_k|^2$ and using $\Delta V_k = \alpha_k + j\beta_k$ yields:

$$2e_k \alpha_k + 2 f_k \beta_k = \Delta(|V_k|^2) \qquad (2\text{-}11)$$

thus in the case of (P-V) type buses we have:

$$\begin{bmatrix} \Delta P_k \\ \Delta(|V_k|^2) \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ 2e_k & 2f_k \end{bmatrix} \cdot \begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix} \qquad (2\text{-}12)$$

Step 4) Step 3 is repeated for all buses except the slack bus. Corrections are applied to the voltages as soon as they are calculated.

Step 5) Various stop criteria may be used at each iteration cycle. In the original Ward-Hale method, the stop criterion made use of successive calculated values of voltages. If the stop criterion is satisfied, then the process is terminated. Otherwise, the process will be repeated starting from step 2.

Acceleration

The algorithm described above can be made more efficient by implementing certain changes such as the use of acceleration factors. The idea is to multiply the correction voltage values by factors before adding them to the present values of voltages. These factors can be found experimentally so that using them will speed up the algorithm. Usually the values that show the most satisfactory results lie in the range of 1.5 to 1.8. Ordinarily different factors are applied to the real and

imaginary parts of ΔV_k.

Another method of speeding up the algorithm is through the use of "relaxation techniques" [4]. In this method, the order in which the unknowns are calculated may be varied in each iteration cycle. The residuals are calculated and the next bus to be corrected is the one having the largest residual. Sometimes the corrections are based upon the values greater or smaller than the actual residuals. These are called "over-relaxation" and "under-relaxation" techniques, respectively.

Storage Requirements And Programming Techniques

Computer storage is needed mainly for the nodal admittance matrix [Y]. However, if all the zero and non-zero elements of [Y] are stored, this imposes a serious restriction on the size of the power system that can be handled. For example, a 400-bus system cannot be solved on a computer that has 1000k bytes of memory.

Fortunately, [Y] is a very sparse matrix. Taking advantage of its sparsity by storing only non-zero elements, programs can be written to handle power systems of much larger size on the same computer. Appendix B describes the storage scheme used in writing this program. This is a new scheme that makes it possible to access any element of the matrix with little scanning effort. Also it stores the elements and the pointers such that, in a time-sharing computer system with virtual

memory, the number of READ/WRITE operations on a secondary storage device is less than that used in conventional schemes.

2.2 Newton's Method

Newton's method [7,9] falls into the category of so-called direct methods and has largely superseded the Ward-Hale method in load-flow studies of power systems. It consists of the direct solution of a set of linear equations in an iterative process. These linear equations relate the small changes in real and reactive powers to the small changes in voltage magnitudes and phase angles. When the changes are not small, these equations will be approximations to the actual expressions and using them in an iterative process will result in the final answers. If changes are too large, then the equations will not be able to converge to the final solution. The method might diverge or converge to a wrong answer. Thus, the closeness of initial estimates to the final solution is a very important factor in determining the convergence of Newton's method. Usually the range of initial estimates for which Newton's method converges to the right solution is smaller than that for the Ward-Hale method.

Storage is the main problem in this method if sparsity of the matrices is not taken into account through sub-optimal ordering of buses [8,9]. This is explained later. Even then the storage requirements are larger than in the Ward-Hale method. If storage is not critical , this method is preferable

to the Ward-Hale method because it can solve some problems that the Ward-Hale method cannot. Also for problems that both methods can solve, Newton's method converges faster.

The Algorithm

Newton's method can be formulated using either polar or rectangular co-ordinates. Polar co-ordinates are commonly used because it has proven to be more advantageous [17]. Formulation of Newton's method using polar co-ordinates can be briefly explained as follows:

Let $$V_k \triangleq |V_k| \, e^{j\theta_k} \triangleq e_k + j\, f_k \qquad (2\text{-}13)$$

and $$Y_{kj} \triangleq |Y_{kj}| \, e^{j\delta_{kj}} \triangleq G_{kj} + j\, B_{kj} \qquad (2\text{-}14)$$

then $$P_k + j\, Q_k = \sum_j |V_k| \, |Y_{kj}| \, |V_j| \, e^{j(\theta_k - \delta_{kj} - \theta_j)} \qquad (2\text{-}15)$$

Also let $$a_j + j\, b_j \triangleq (e_j + j\, f_j)(G_{kj} + j\, B_{kj}) \qquad (2\text{-}16)$$

From (2-15) the first order sensitivities can be found [6] :

$$\frac{\partial P_k}{\partial \theta_j} = H_{kj} \qquad \frac{\partial P_k}{\partial |V_j|} |V_j| = N_{kj}$$

$$\frac{\partial Q_k}{\partial \theta_j} = J_{kj} \qquad \frac{\partial Q_k}{\partial |V_j|} |V_j| = L_{kj} \tag{2-17}$$

where for $k \neq j$

$$K_{kj} = L_{kj} = a_j f_k - b_j e_k$$

$$N_{kj} = -J_{kj} = a_j e_k + b_j f_k \tag{2-18}$$

and for $k=j$

$$H_{kk} = - Q_k - B_{kk} |V_k|^2$$

$$L_{kk} = Q_k - B_{kk} |V_k|^2$$

$$N_{kk} = P_k + G_{kk} |V_k|^2 \tag{2-19}$$

$$J_{kk} = P_k - G_{kk} |V_k|^2$$

Thus the Jacobian matrix, which relates small changes in voltages to small changes in power flows, has the following form:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ J & L \end{bmatrix} \cdot \begin{bmatrix} \Delta \theta \\ \frac{\Delta |V|}{|V|} \end{bmatrix} \tag{2-20}$$

For any (P-Q) type bus, equation (2-20) can be written. But for

a (P-V) type bus only the first equation of (2-20) can be written because Q is not scheduled for such a bus and, therefore, ΔQ is unknown. Also at these buses only $\Delta\theta$ is unknown. This makes the number of equations and unknowns equal. In a system with n_1 (P-Q) type and n_2 (P-V) type buses this number will be:

$$m=2n_1+n_2$$

The corrections to the magnitudes and phase angles of bus voltages can then be found by solving the set of linear equations (2-20). Applying these corrections to the voltages will result in a new set of voltages which are not necessarily the final solution since the corrections may not be very small. These voltages are used as initial values for the next iteration and corrections are calculated in the same way. This process is repeated as many times as necessary until the final solution is achieved.

Storage Problem

If in forming the Jacobian matrix each sub-matrix

$$D_{kj} = \begin{bmatrix} H_{kj} & N_{kj} \\ J_{kj} & L_{kj} \end{bmatrix} \qquad (2\text{-}21)$$

is considered as one element, then the Jacobian matrix will have the same format as the [Y] matrix except that it is not necessarily symmetric. Thus the Jacobian matrix is a sparse matrix. However, in order to solve the set of linear equations

(2-20), the inverse of the Jacobian matrix has to be determined and the inverse is not necessarily sparse. In fact, it is usually a full matrix. For large power systems, the amount of storage needed to store a full matrix, such as the inverse of the Jacobian matrix, is prohibitive. This puts a serious restriction on the application of this method. The same problem occurs in any of the so-called direct methods where a set of linear sparse equations has to be solved.

When the Gauss elimination method is used, it may be possible to order the equations such that the number of non-zero terms appearing in the upper triangular matrix that results will be minimal. However, no method has yet been found for finding such an absolute optimal ordering of the equations. Some sub-optimal ordering schemes are discussed in the next section [9]. Any of these schemes will make Newton's method practical for large power systems.

Ordering Of The Equations

Any one of the following schemes can be used to order the equations in a sub-optimal manner. They are listed in increasing order of efficacy and execution time:

1) Order the rows starting with the one having the least non-zero elements, prior to the elimination process, and ending with the one having the most.

2) Order the rows such that at each step of the elimination process the row to be processed next is the one having the least

non-zero terms.

3) Order the rows such that at each step of the elimination process the row to be processed next is the one that will produce the least number of new non-zero terms in the matrix after eliminating this row.

Scheme 1 is very simple but it does not take into account any changes that occur during the elimination process. Scheme 2 requires the simulation of the elimination process and scheme 3 requires the simulation of every feasible alternative at each step of the elimination process.

In this thesis scheme 2 is chosen and the elimination process is simulated in the following way:

A connection matrix that represents the topology of the system is formed. This matrix has a one in position (i,j) if and only if $Y_{ij} \neq 0$; otherwise it has a zero in that position. Since the pattern of non-zero elements in the admittance and Jacobian matrices are the same, this connection matrix can be used to order the rows in the Jacobian matrix. Because the value of each element in this matrix is either 1 or 0, only one bit of memory need be used to represent each element. Thus, the connection matrix for a 1000 node system will take only 125k bytes. Therefore, large power systems can be handled without requiring large computer memory.

The elimination process is simulated by using logical AND, OR and EXCLUSIVE OR operations. After eliminating one bus the non-zero elements of all the other rows are counted and the next bus to be eliminated is chosen.

When the simulation process is finished and the order of buses has been specified, the rows of the Jacobian matrix are eliminated in that order. Each row in the Jacobian matrix is either a single row (for (P-V) type buses) or a double row (for (P-Q) type buses).

The ordering of the equations need be determined only once, prior to elimination, and is used for each and every iteration that the problem might take.

Programming techniques

The scheme for storing the [Y] matrix is the same as that used in the Ward-Hale method. The data structure and the logic for forming and storing the Jacobian matrix is explained in Appendix C. Except for some changes in the working-row scheme, it is essentially the same as that suggested by Tinney and Hart [9]. It is obvious that the computer storage required by this method is more than that required in the Ward-Hale method because storage is needed for both the Jacobian and the [Y] matrices, and the Jacobian matrix requires about twice as much storage as [Y]. Consequently, for very large power systems, or whenever computer storage is very critical, the Ward-Hale method is preferred.

If rectangular co-ordinates are used instead, then the Jacobian matrix will need even more computer storage space because two equations describe every bus. However, the logic

for storing the Jacobian matrix and for performing the elimination process will be somewhat simpler. Also the need for using sine and cosine functions at each iteration is removed. Overall, both formulations take about the same time to execute. Any difference in convergence characteristics is in favor of polar co-ordinates [9].

3. PROPOSED METHOD

3.1 Introduction

A new approach to the solution of the load-flow problem will be described in this chapter. The proposed method is based upon a different formulation of the load-flow equations. The resultant equations have only one nonlinear term each instead of more nonlinear terms as in previously reported methods. The nonlinear term can be linearly approximated, and the solution of the linearized equations can be used in an iterative procedure to determine the answers to the load-flow problem. Thus the proposed method is a direct method [1].

3.2 Derivation Of The Equations

Let the final values of voltages, currents and powers be represented, respectively, by:

$$[V] = \{V_1, V_2, \ldots, V_n\}$$
$$[I] = \{I_1, I_2, \ldots, I_n\} \qquad (3\text{-}1)$$
$$[S] = \{S_1, S_2, \ldots, S_n\}$$

and the present assumed or calculated values of these quantities be represented by the corresponding primed symbols:

$$[V'] = \{V'_1, V'_2, \ldots, V'_n\}$$
$$[I'] = \{I'_1, I'_2, \ldots, I'_n\} \qquad (3\text{-}2)$$
$$[S'] = \{S'_1, S'_2, \ldots, S'_n\}$$

where n is the number of buses in the power system.

The usual formulation of the load-flow equations is derived by working from the present assumed or calculated values of (3-2) and correcting them until the final values of (3-1) are reached. The proposed method, on the other hand, is based upon equations that are derived by assuming that the power system is in the final state defined by the values of (3-1) and finding the changes that will bring it to the present state defined by the values of (3-2).

Initially assume all bus voltages are held constant at their final values except for the voltage of bus i which is changed to :

$$V'_i = V_i + \Delta V_i \qquad (3\text{-}3)$$

This will cause a change in the power at bus i itself as well as all the other buses interconnected to this bus. The change in the power at bus k can be calculated as follows:

$$\begin{aligned}\Delta S_k &= S'_k - S_k \\ &= V'_k I'^{*}_k - V_k I^{*}_k \\ &= (V_k + \Delta V_k)(I_k + \Delta I_k)^{*} - V_k I^{*}_k \end{aligned} \qquad (3\text{-}4)$$

or $$\Delta S_k = V_k \Delta I_k^* + \Delta V_k I_k^* + \Delta V_k \Delta I_k^* \qquad (3\text{-}5)$$

but $\Delta V_k = 0$ for $k \neq i$

hence $$\Delta S_i = V_i \Delta I_i^* + \Delta V_i I_i^* + \Delta V_i \Delta I_i^* \qquad (3\text{-}6)$$

and $$\Delta S_k = V_k \Delta I_k^* \qquad k \neq i \qquad (3\text{-}7)$$

Also $$I_k = \sum_j Y_{kj} V_j \qquad (3\text{-}8)$$

therefore for any k

$$\Delta I_k = Y_{ki} \Delta V_i \qquad (3\text{-}9)$$

and substituting in (3-6) and (3-7), we get:

$$\Delta S_i = V_i Y_{ii}^* \Delta V_i^* + \Delta V_i I_i^* + \Delta V_i Y_{ii}^* \Delta V_i^* \qquad (3\text{-}10)$$

$$\Delta S_k = V_k Y_{ki}^* \Delta V_i^* \qquad k \neq i \qquad (3\text{-}11)$$

Equation (3-10) may also be written as:

$$\Delta S_i = V_i' Y_{ii}^* \Delta V_i^* + \Delta V_i I_i^* \qquad (3\text{-}12)$$

Similarly all the final (unprimed) values of the bus

voltages are each changed in turn to their present (primed) values. At the end of this process [S] will be changed to [S']. It is obvious that this can be done in any order. For convenience the bus voltages will be changed in the order that the buses are numbered and assume that bus 1 is the slack bus

where $$V_1' = V_1 \qquad (3\text{-}13)$$

and $$\Delta V_1 = 0$$

For any bus i in the system, the total change in power may be defined as the sum of three changes as follows:

i) ΔS_i^I which is the change of power at bus i due to changes in the voltages of buses 1 to (i-1);

ii) ΔS_i^{II} which is the change of power at bus i due to the change in the voltage of bus i and;

iii) ΔS_i^{III} which is the change of power at bus i due to changes in the voltages of buses (i+1) to n.

Using the above definitions and equations (3-11) and (3-12) we get:

$$\Delta S_i^I = \sum_{k=1}^{i-1} V_i \, Y_{ik}^* \, \Delta V_k^* \qquad (3\text{-}14)$$

$$\Delta S_i^{II} = V_i' \, Y_{ii}^* \Delta V_i^* + \Delta V_i I_i^* \qquad (3\text{-}15)$$

$$\Delta S_i^{III} = \sum_{k=i+1}^{n} V_i' \, Y_{ik}^* \, \Delta V_k^* \qquad (3\text{-}16)$$

but

$$I_i = \sum_{k=1}^{i-1} Y_{ik} V'_k + \sum_{k=i}^{n} Y_{ik} V_k$$

$$= \sum_{k=1}^{i-1} Y_{ik} \Delta V_k + \sum_{k=1}^{n} Y_{ik} V_k \qquad (3\text{-}17)$$

Substituting for I_i in (3-15) and adding (3-14) through (3-16) together to find the total power change we therefore have:

$$\Delta S_i^t = \Delta S_i^I + \Delta S_i^{II} + \Delta S_i^{III}$$

$$= \sum_{k=1}^{i-1} V_i Y_{ik}^* \Delta V_k^* + \Delta V_i \{ \sum_{k=1}^{i-1} Y_{ik} \Delta V_k + \sum_{k=1}^{n} Y_{ik} V_k \}^*$$

$$+ \sum_{k=i}^{n} V'_i Y_{ik}^* \Delta V_k^* \qquad (3\text{-}18)$$

or $$\Delta S_i^t = \sum_{k=1}^{n} V'_i Y_{ki}^* \Delta V_k^* + \Delta V_i \sum_{k=1}^{n} Y_{ik}^* V_k^* \qquad (3\text{-}19)$$

Now $$S_i = V_i \sum_{k=1}^{n} Y_{ik}^* V_k^* \qquad (3\text{-}20)$$

therefore $$\sum_{k=1}^{n} Y_{ik}^* V_k^* = \frac{S_i}{V_i} \qquad (3\text{-}21)$$

and (3-19) becomes

$$\Delta S_i^t = \sum_{k=1}^{n} V'_i Y_{ik}^* \Delta V_k^* + \Delta V_i \frac{S_i}{V_i} \qquad (3\text{-}22)$$

Note that only the last term on the right hand side of (3-22) is nonlinear and can be linearized by using some approximate value for V_i. For the complete system, (n-1)

equations, of the form of (3-22), would be obtained.

Equation (3-22) was derived by using the set of [V'] as "final" values and the set of [V] as "initial" values. But equation (3-22) is still valid when the [V] values are "final" and the [V'] values are "initial" as defined by (3-1) and (3-2), respectively, because ΔS_i^t and ΔV_i will both change sign. Note, however, that equation (3-22) is not derivable by using the latter approach and, in fact, each of the equations obtained would have one linear term and (n-1) nonlinear terms.

As expected, equation (3-22) does not depend upon the order in which the bus voltages are changed to their final values. Therefore, we can make use of this fact and for any bus i we can assume that it is the first bus to be changed to its final value with all the other bus voltages set at their present values. Equation (3-22) would result.

3.3 The Algorithms

The manipulation of equation (3-22) in conjunction with different assumptions yields various algorithms to solve the load-flow problem. Some of the many possible algorithms will now be described.

Algorithm A

Rewriting (3-22) as

$$S_i - S_i' - \frac{\Delta V_i}{V_i} S_i = \sum_{k=1}^{n} V_i' Y_{ik}^* \Delta V_k^* \qquad (3\text{-}23)$$

we get $$\frac{V_i'}{V_i} S_i - S_i' = \sum_{k=1}^{n} V_i' Y_{ik}^* \Delta V_k^* \qquad (3\text{-}24)$$

If we assume that $V_i' \simeq V_i$,

i.e., $$\frac{V_i'}{V_i} \simeq 1$$

then $$\Delta S_i = \sum_{k=1}^{n} V_i' Y_{ik}^* \Delta V_k^* \qquad (3\text{-}25)$$

Equation (3-25) can be used as a linear approximation to (3-22) with the nonlinear term set to zero. This seems reasonable since the coefficient of ΔV_i in the nonlinear term is in the order of 1.0 P.U. and, therefore, much smaller than the coefficient of ΔV_i^* which is $V_i' Y_{ii}^*$. Equating real and imaginary parts of equation (3-25) will result in

$$\Delta P_i = \sum_k A_{ik}\alpha_k + \sum_k B_{ik}\beta_k \qquad (3\text{-}26)$$

$$\Delta Q_i = \sum_k B_{ik}\alpha_k - \sum_k A_{ik}\beta_k \qquad (3\text{-}27)$$

where $$V_i' Y_{ik}^* \triangleq A_{ik} + j B_{ik}$$

and $$\Delta V_k \triangleq \alpha_k + j\beta_k$$

For a (P-Q) type bus both equations (3-26) and (3-27) can be written. However, for a (P-V) type bus, Q_i is not scheduled and, therefore, (3-27) cannot be written, but the magnitude of voltage is to be kept constant at such a bus. This constraint can be written as:

$$(V_i' + \Delta V_i)(V_i' + \Delta V_i)^* = U_i^2 \qquad (3\text{-}28)$$

or $$|V_i'|^2 + V_i' \Delta V_i^* + \Delta V_i V_i'^* + \Delta V_i \Delta V_i^* = U_i^2 \qquad (3\text{-}29)$$

i.e., $$2\mathcal{R}\{\frac{V_i + V_i'}{2} \Delta V_i^*\} = U_i^2 - |V_i'|^2 \qquad (3\text{-}30)$$

Equation (3-30) is exact but nonlinear. By using V_i' instead of V_i we find the linear approximation:

$$\frac{U_i^2 - |V_i'|^2}{2} = e_i\alpha_i + f_i\beta_i \qquad (3\text{-}31)$$

where $V_i' \triangleq e_i + jf_i$

Writing (3-26) and (3-27) for any (P-Q) type bus and (3-26) and (3-31) for any (P-V) type bus, and solving the resultant set of equations, the approximate voltage corrections can be found.

Algorithm B

If V_i' is used instead of V_i in the nonlinear term of (3-22) then

$$\Delta S_i = \sum_k V_i' Y_{ik}^* \Delta V_k^* + \Delta V_i \frac{S_i}{V_i'} \qquad (3\text{-}32)$$

Let $$\Delta V_k \triangleq \alpha_k + j\beta_k \qquad (3\text{-}33)$$

$$V_i' Y_{ik}^* \triangleq A_{ik} + jB_{ik} \qquad (3\text{-}34)$$

and $$\frac{S_i}{V_i'} \triangleq a_i + j\, b_i \qquad (3\text{-}35)$$

then equating real and imaginary parts of (3-32) we obtain:

$$\Delta P_i = \sum_k A_{ik}\alpha_k + \sum_k B_{ik}\beta_k + a_i\alpha_i - b_i\beta_i \qquad (3\text{-}36)$$

$$\Delta Q_i = \sum_k B_{ik}\alpha_k - \sum_k A_{ik}\alpha_k + b_i\alpha_i + a_i\beta_i \qquad (3\text{-}37)$$

For a (P-Q) type bus both equations (3-36) and (3-37) can be written. However, for a (P-V) type bus Q_i is not scheduled and, therefore, (3-37) cannot be written and a_i and b_i cannot be calculated. In algorithm B, equations (3-36) and (3-37) are used in the case of (P-Q) type buses and equations (3-26) and (3-31) in the case of (P-V) type buses.

Algorithm C

This algorithm is essentially the same as algorithm B except that for (P-V) type buses Q_i' is used instead of Q_i in (3-35) to calculate a_i and b_i.

$$\frac{P_i + jQ_i'}{V_i'} = a_i + j\, b_i \qquad (3\text{-}38)$$

Thus for (P-V) type buses, equations (3-36) and (3-31) apply while for (P-Q) type buses, equations (3-36) and (3-37) again apply.

Voltage acceleration factors may be used in this algorithm as in the Ward-Hale method.

Algorithm D

This algorithm is also essentially based upon algorithm B. By assuming that $U_i \simeq |V_i'|$ in equation (3-35) and rearranging, we get:

$$\alpha_i = -\frac{f_i}{e_i}\beta_i \tag{3-39}$$

where $$V_i' \triangleq e_i + j\, f_i$$

and $$\Delta V_i \triangleq \alpha_i + j\beta_i$$

Equation (3-39) is used to replace α_i in all the other equations. This results in a set of linear equations that have only one equation and one unknown for each (P-V) type bus.

Let k_1 and k_2 represent the set of all (P-Q) type buses and the set of all (P-V) type buses, respectively. Then for any (P-Q) type bus we have:

$$\Delta P_i = \sum_{k_1} (A_{ik}\alpha_k + B_{ik}\beta_k) + \sum_{k_2} (B_{ik} - A_{ik}\frac{f_k}{e_k})\beta_k + a_i\alpha_i - b_i\beta_i \tag{3-40}$$

$$\Delta Q_i = \sum_{k_1} (B_{ik}\alpha_k - A_{ik}\beta_k) - \sum_{k_2} (A_{ik} + B_{ik}\frac{f_k}{e_k})\beta_k + b_i\alpha_i + a_i\beta_i \tag{3-41}$$

where

$$V_i' Y_{ik}^* \triangleq A_{ik} + j\, B_{ik}$$

and

$$\frac{P_i + jQ_i}{V_i'} = a_i + j\, b_i$$

and for any (P-V) type bus we have:

$$\Delta P_i = \sum_{k_1} (A_{ik}\alpha_k + B_{ik}\beta_k) + \sum_{k_2} (B_{ik} - A_{ik}\frac{f_k}{e_k})\beta_k - (b_i + a_i\frac{f_i}{e_i})\beta_i \qquad (3\text{-}42)$$

where

$$\frac{P_i + jQ_i'}{V_i'} = a_i + j\, b_i$$

thus, for each (P-Q) type bus we will have two equations, (3-40) and (3-41) and two unknowns, α_i and β_i. But for each (P-V) type bus there is only one equation (3-42) and one unknown β_i.

Algorithm E

Rearranging equation (3-31) of algorithm B, we have :

$$\alpha_i = -\frac{f_i}{e_i}\beta_i + \frac{U_i^2 - |V_i'|^2}{2e_i} \qquad (3\text{-}43)$$

and, as in the case of algorithm D, (3-43) can be substituted in (3-36) and (3-37). Then for (P-Q) type buses we have:

$$\Delta P_i - \sum_{k_2} A_{ik}\left(\frac{U_k^2 - |V_k'|^2}{2e_k}\right) = \sum_{k_1} (A_{ik}\alpha_k + B_{ik}\beta_k) + \sum_{k_2} (B_{ik} - A_{ik}\frac{f_k}{e_k})\beta_k + a_i\alpha_i - b_i\beta_i \qquad (3\text{-}44)$$

$$\Delta Q_i - \sum_{k_2} B_{ik}\left(\frac{U_k^2 - |V_k'|^2}{2e_k}\right) = \sum_{k_1} (B_{ik}\alpha_k - A_{ik}\beta_k) - \sum_{k_2} (A_{ik} + B_{ik}\frac{f_k}{e_k})\beta_k + b_i\alpha_i + a_i\beta_i \qquad (3\text{-}45)$$

whereas for (P-V) type buses:

$$\Delta P_i - \sum_{k_2} A_{ik} \left(\frac{U_k^2 - |V_k'|^2}{2e_k}\right) - a_i \frac{U_i^2 - |V_i'|^2}{2e_i}$$

$$= \sum_{k_1} (A_{ik}\alpha_k + B_{ik}\beta_k) + \sum_{k_2} (B_{ik} - A_{ik}\frac{f_k}{e_k})\beta_k$$

$$- (b_i + a_i \frac{f_i}{e_i})\beta_i \qquad (3\text{-}46)$$

3.4 Programming And Storage

As in the case of Newton's method, the solution of a set of linear simultaneous equations poses a computer storage problem but this can be overcome by the use of sub-optimal ordering of the buses and sparsity programming. Storage requirements for algorithms A, B and C are slightly more than that for Newton's method. For algorithms D and E the storage requirements are the same as for Newton's method. The sparsity of the matrix of coefficients in these algorithms is the same as that for the Jacobian matrix. So by using a scheme to order the buses in a sub-optimal fashion, the amount of storage required by algorithms D and E will be the same as Newton's method in polar co-ordinates.

For algorithms D and E, the storage scheme for the matrix of coefficients is identical to that used for the Jacobian matrix in the polar form of Newton's method, as described in Appendix C. For algorithms A, B and C the logic for storing the elements is simpler but storage requirements are greater.

4. NUMERICAL RESULTS

4.1 Introduction

The algorithms described in Chapters 2 and 3 were programmed and compared with each other using several test systems. The test systems were of different sizes and configurations, as shown in Table (4-1). The data given [22] for system 3 is erroneous and incomplete: the leakage reactance of one transformer that should be 0.034 P.U. is not specified and the reactance of one transmission line is recorded as a negative value when it should be positive. However, the system as described [22] with negligible transformer leakage (zero) and negative line reactance is used as test system 3' and the corrected system as test system 3.

In addition to the original systems mentioned in Table (4-1), new test systems were created by changing the buses in the original system system to be all of either (P-Q) type or (P-V) type. To differentiate between test systems with the same configuration the addition of the letters a, b and c were used for the original system, the all (P-Q) type system and the all (P-V) type system, respectively. For example, system 1.a is the original system, system 1.b is system 1 with all its buses of (P-Q) type and system 1.c is the same system with all its buses of (P-V) type.

The notation used to represent the different algorithms are summarized in Table (4-2). Note that the Ward-Hale method was

programmed using no acceleration technique. Algorithms A, B, C, D and E are the proposed algorithms described in Chapter 3.

Two sets of initial values are used for each case, the "flat-start" and the so-called "final-start". For the flat-start all the voltage magnitudes are set to their scheduled values or, if not scheduled, to the magnitude of the slack bus and all the angles are set equal to zero. The final-start is the set of initial values close to the final answers, and may be either the rounded final values, the final solution of the system obtained with a bigger tolerance or the final solution of a similar system. Comparison of the results obtained by using the flat-start and the final-start will indicate the dependence of the convergence of an algorithm upon the initial values.

The stop criterion for all the algorithms is the slack bus power mismatch. The tolerance of the power mismatch is different for the various systems as follows:

1) 0.0005 p.u. for systems 1.a, 1.b and 1.c;
2) 0.00005 p.u. for systems 2.a, 2.b and 2.c;
3) 0.005 p.u. for systems 3 and 3';
4) 0.0005 p.u. for systems 4.a, 4.b and 4.c;
5) 0.5 p.u. for system 5.

NOTATION	# OF P-V	# OF P-Q	TOTAL# OF BUS	# OF LINES	# OF TX.	# OF BR.	SOURCE OF DATA
System 1	1	3	5	6	0	6	Ref [21]
System 2	1	4	6	7	2	7	Ref [2]
System 3	2	18	21	29	3	32	Ref [22]
System 4	4	28	33	23	11	34	Ref [18]
System 5	22	70	93	99	57	156	Ref [19]

TABLE (4-1) Test Systems.

W	Ward-Hale method
N	Newton's method (polar form)
A	Algorithm A
B	Algorithm B
C	Algorithm C
D	Algorithm D
E	Algorithm E

TABLE (4-2) Algorithms.

4.2 Results

Tables (4-3) and (4-4) show the execution time and the number of iterations taken by each algorithm when the flat-start is used; and Tables (4-5) and (4-6) when the final-start is used.

Comparison Of Convergence Characteristics

As can be seen from Tables (4-4) and (4-6), all the algorithms based upon the proposed method, except algorithm A, converge for all the cases with both the flat-start and the final-start. Algorithm A did not converge for system 5, even when the final-start was used. The reason for this was found after a careful study of the data of system 5. For several buses of this system the net injected power is about 40 P.U., and the branches connected to these buses have either small admittance or negative resistance. Thus, the value of $\frac{S_i}{V_i}$ is comparable to $V_i Y_{ii}^*$ and therefore (3-25) is not a good approximation to (3-22).

Algorithm B converges for all the cases but its weakness regarding (P-V) type buses is noticeable. As a matter of fact if all the buses in a system were (P-V) type, then this algorithm would be exactly the same as algorithm A. Therefore, algorithm B is weak for systems with too many (P-V) type buses, such as system 5.

		SYSTEMS											
		1.a	1.b	1.c	2.a	2.b	2.c	3	3'	4.a	4.b	4.c	5
ALGORITHMS	W	57	59	59	239	295	239	23418	(**)	108045	108447	109064	812622
	N	38	42	20	72	79	39	806	1434	1721	(**)	1054	24083
	A	69	63	59	191	178	99	1041	741	27071	15594	3483	(**)
	B	45	34	58	87	69	99	561	962	2536	2318	3550	463931
	C	33	37	42	68	68	66	823	969	2677	2348	1483	30055
	D	32	35	26	67	73	46	935	1048	1951	3116	1262	25085
	E	31	34	28	63	68	59	832	806	1783	2938	988	25832

TABLE (4-3) Comparing the execution times taken by different algorithms when flat-start is used.

Times in milliseconds.

(**) means no convergence

S Y S T E M S

ALGORITHMS	1.a	1.b	1.c	2.a	2.b	2.c	3	3'	4.a	4.b	4.c	5
W	16	17	15	49	64	45	>500	(**)	>1000	>1000	>1000	1028
N	3	3	2	4	4	3	3	6	3	(**)	3	3
A	5	5	4	10	10	5	4	3	57	38	8	(**)
B	3	2	4	4	3	5	2	4	5	5	8	49
C	2	2	2	3	3	3	3	4	5	5	3	3
D	2	2	2	3	3	3	3	4	3	5	3	3
E	2	2	2	3	3	3	3	4	3	5	2	3

TABLE (4-4) Comparing the number of iterations taken by different algorithms when flat-start is used.

(**) means no convergence

		SYSTEMS											
		1.a	1.b	1.c	2.a	2.b	2.c	3	3'	4.a	4.b	4.c	5
ALGORITHMS	W	36	39	30	82	128	62	11445	(**)	45511	32712	62998	196815
	N	17	18	14	26	28	18	346	(**)	1260	2309	775	17386
	A	47	42	20	82	76	29	556	720	18605	12884	1490	(**)
	B	20	21	20	29	30	29	338	533	711	632	1488	16120
	C	23	24	23	31	34	35	340	308	705	661	1042	12661
	D	20	22	17	30	31	24	405	375	874	911	930	11265
	E	19	20	17	29	29	24	365	347	798	828	611	11705

TABLE (4-5) Comparing the execution times taken by different algorithms when final-start is used.

Times in milliseconds.

(**) means no convergence

S Y S T E M S

ALGORITHMS	1.a	1.b	1.c	2.a	2.b	2.c	3	3'	4.a	4.b	4.c	5
W	8	11	7	15	27	11	250	(**)	415	298	578	250
N	1	1	1	1	1	1	1	(**)	2	4	2	2
A	3	3	1	4	4	1	2	3	40	31	3	(**)
B	1	1	1	1	1	1	1	2	1	1	3	1
C	1	1	1	1	1	1	1	1	1	1	2	1
D	1	1	1	1	1	1	1	1	1	1	2	1
E	1	1	1	1	1	1	1	1	1	1	1	1

TABLE (4-6) Comparing the number of iterations taken by different algorithms when final-start is used.

(**) means no convergence

Algorithms C, D and E have almost identical convergence characteristics. They converged for all the test systems in a few iterations. However, for system 4.c algorithm E converged faster than the other two. This should be true for any system with too many (P-V) type buses since the exact equation (3-43) is used instead of the approximate equation (3-39) in algorithm E.

The Ward-Hale method converged for all the test systems, except system 3'. It can be seen that it takes a large number of iterations to converge for large systems.

Newton's method failed to converge for system 4.b when the flat-start was used but converged when the final-start was used. This method also did not converge for system 3' even when the final-start was used, the final-start being the solution of system 3. These two cases indicate that Newton's method is more dependent upon the initial values than any of the algorithms based upon the proposed method. Dependency of Newton's method to a favorable initial value has also been discussed by others [20].

Comparison Of Computing Speed

The execution times mentioned in Tables (4-3) and (4-5) are not very accurate, since a time-sharing computer system was used to compare the algorithms. Some cases were run more than once and the average execution time is recorded in Tables (4-3) and (4-5). Experience has shown that the figures given in these tables have a deviation of, at most, 6%.

Algorithm A is the slowest among the algorithms of the proposed method. In the case of system 3', however, we can see that it converges faster than any of the other algorithms. This is only fortuitous because of the tolerance used.

Algorithm B is better than algorithm A. It converges for all the test systems and is faster than algorithm A. For system 3, this algorithm converged faster than any of the other algorithms, but this again is fortuitous because of the tolerance used.

Algorithm C is faster than algorithm B but it is still weak as far as (P-V) type buses are concerned, the reason being that it has two equations and two unknowns for a (P-V) type bus while the polar form of the Newton's method has only one equation and one unknown for such a bus. Therefore, algorithm C solves a system of linear equations with a rank greater than the rank of equations in Newton's method and this results in a greater execution time per iteration.

Algorithms D and E combine the two equations of a (P-V) type bus into one equation and, therefore, use a system of

linear equations with the same number of equations and unknowns as in the polar form of Newton's method. Tables (4-3) and (4-5) show that these two algorithms are faster than Newton's method. Algorithm E uses the exact form of (3-43) and is better when all the buses of a large system are of (P-V) type. This kind of system is unusual and, therefore, we can assume that algorithm D can always be considered as the best algorithm from the point of view of speed.

Comparison Of Storage Requirements

Algorithms D and E require the same storage as the polar form of Newton's method because the matrix of coefficients has exactly the same form as the Jacobian matrix. The data structure and the logic for storing the matrix of coefficients is identical to that described in Appendix C for storing the Jacobian matrix.

The storage requirements of algorithms A, B and C is slightly higher, but these algorithms need less storage than the Newton's method in rectangular form.

The storage requirements for the Ward-Hale method is about half of the storage needed for any of the other algorithms. This is the main advantage of the method and is used when storage is a problem.

4.3 Summary

When the Ward-Hale method is used with acceleration factors it is more efficient. For example, in the case of system 1.a using the acceleration factor of 1.2 will reduce the number of iterations by half. The problem is that the best acceleration factors must be determined for each system individually and this requires several runs of the program in order to find the best factors. Use of acceleration factors was also tried with algorithm C but showed no advantage. All the results previously discussed in this chapter use no acceleration factors.

Overall, we can see that algorithms D and E are faster and have better convergence characteristics than either Newton's method or the Ward-Hale method. Also their storage requirements is about the same as the polar form of Newton's method and about twice as much as the Ward-Hale method. If storage is not a problem, these algorithms are preferable to both the Ward-Hale method and Newton's method.

5. CONCLUSIONS

A new formulation of the load-flow problem has been presented in this thesis. Some of the many possible algorithms that can be derived based upon this new formulation were programmed and compared with the Ward-Hale method and Newton's method. Several test systems of different sizes and configurations were used to compare the algorithms. It was shown that:

1) Algorithms based upon the new formulation converge over a wider range of initial values and therefore have better convergence characteristics than Newton's method and the Ward-Hale method.

2) Algorithms D and E of those based upon the new formulation are faster than the Newton's method and the Ward-Hale method.

3) The storage requirements for any of the proposed algorithms is about the same as the polar form of Newton's method and, therefore, by use of sub-optimal ordering of the equations and improved programming techniques storage will not be a major problem.

4) Use of acceleration factors will not improve the performance of the algorithms based upon the new formulation.

REFERENCES

1. M.A.Laughton and M.W.Humphrey Davies: "Numerical Techniques Of Power-System Load-Flow Problems", Proc. IEE, Vol. III, No. 9 pp. 1575-1588, September 1964.

2. J.B.Ward and H.W.Hale: "Digital Computer Solution Of Power-Flow Problems", AIEE Trans. on PAS Vol. 75, pp. 398-404, June 1957.

3. R.J.Brown and W.F.Tinney: "Digital Solutions For Large Power Networks", AIEE Trans. on PAS Vol. 76, pp. 347, June 1957.

4. R.H.Jordan: "Rapid Converging Digital Load-Flow", AIEE Trans. on PAS Vol. 76, pp. 1433-1438, February 1958.

5. A.F.Glimm and G.W.Stagg: "Automatic Calculation Of Load Flows", AIEE Trans. on PAS Vol. 76, pp. 817-825, October 1957.

6. J.E.Van Ness: "Iteration Methods For Digital Load-Flow Studies" AIEE Trans. on PAS Vol. 78A. Pt. III, pp. 583-588, August 1959.

7. J.E.Van Ness and J.H.Griffin: "Elimination Methods For Load-Flow Studies", AIEE Trans. on PAS Vol. 80, pp. 299-304, June 1961.

8. W.F.Tinney and J.W.Walker: "Direct Solutions Of Network Equations By Optimally Ordered Triangular Factorization", Proc. Of the IEEE, Vol. 55, pp. 1801-1809, November 1967.

9. W.F.Tinney and C.E.Hart: "Power Flow Solution By Newton's Method", IEEE Trans. on PAS Vol. 86, pp. 1449-1460, November 1967.

10. H.W.Dommel, W.F.Tinney and W.L.Powell: "Further Developments In Newton's Method For Power System Application", Conference Paper. No. 70 CP 161, for the IEEE Winter Power Meeting, 1970.

11. J.P.Britton: "Improved Load-Flow Performance Through A More General Equation Form", IEEE Trans. on PAS Vol. 90, pp. 109-116, June 1971.

12. A.M.Sasson, C.Trevino and F.Aboytes: "Improved Newton's Load-Flow Through A Minimization Technique", IEEE Trans. On PAS, Vol. 90, pp. 1974-1981 September 1971.

13. N.Nabona and L.L.Freris: "New Programming Approach To The Newton-Raphson Load-Flow", Conference Paper, IEEE PES Winter Meeting, Jan 28-Feb 2, 1973.

14. R.Billinton, K.E.Bollinger and S.B.Dhar, "A New Modified Newton's Method For Load-Flow Analysis", Conference Paper No. C72 004-5 for the IEEE Winter Power Meeting, 1972.

15. F.Gomez and A.Semlyen: "The Newton-Raphson Load-Flow With Complex Variables", Conference Paper, IEEE PES, Summer Meeting, July 1972.

16. A.M.Sasson, C.Trevino and F.Aboytes: "Optimal Load-Flow Solution Using The Hessian Matrix", Proc. Of the 1971 PICA, pp. 203-209 May 1971.

17. H.W.Dommel Discussion on Ref 9.

18. T.Nordstrom: "Digital Load-Flow", Report for EE 553 Course, ELEC. ENG. Dept. U.B.C., 1967.

19. T.Nordstrom: "B.C. Hydro Sample Data"

20. W.F.Tinney and H.W.Dommel Discussion on:
Y.P.Dusonchet, S.N.Talukdar, H.E.Sinnot and A.H.El-Abiad: "Load-Flows Using A Combination Of Point Jacobi And Newton's Method", IEEE Trans. on PAS Vol. 90, pp. 941-949, May/June 1971.

21. W.D.Stevenson, Jr: "Power System Analysis", (Book), 2nd Edition, McGraw-Hill, Tokyo, 1962, pp. 219.

22. J.E.Bonaparte and W.W.Maslin: "Simplified Load Flow", Proc. of 1967 PICA, pp. 385-394, May 15-17,1967.

Appendix A.

Assume that a transformer with turns-ratio n and leakage admittance a is connected between buses i and j, as shown in Fig. A.1.

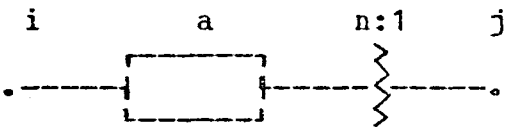


Fig. A.1

This can be represented by the equivalent π circuit shown in Fig. A.2 [2]

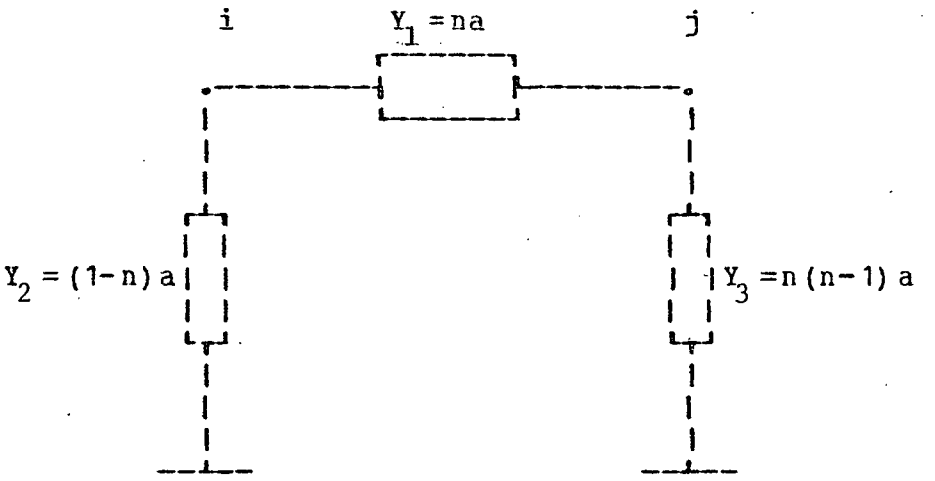


Fig. A.2

For transmission lines the exact π equivalent can be found [21], but this is not necessary unless the lines are very long. Normally the approximate π equivalent, called the nominal π equivalent, is used and is sufficiently good. (Fig. A.3)

The nodal admittance matrix is formed using the equivalent

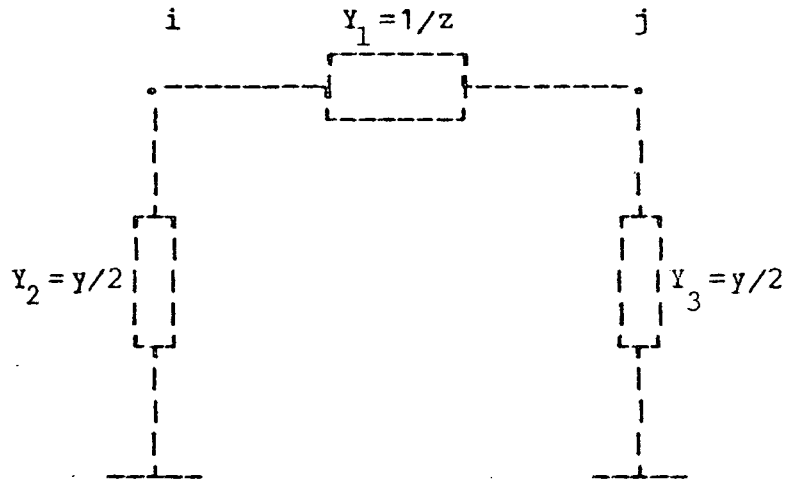


Fig. A.3

circuits of transmission lines and transformers as follows

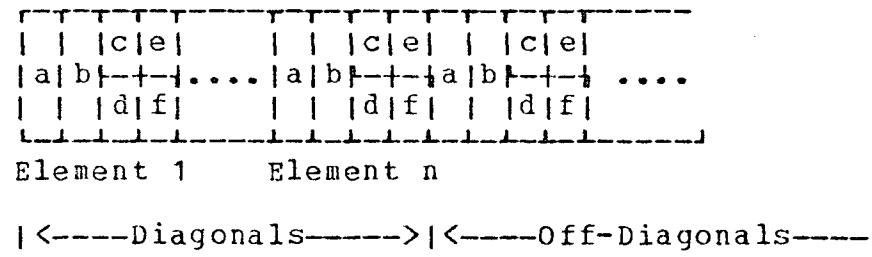
$$Y_{ij} = Y_{ji} = -Y_1$$

$$Y_{ii} = Y_{ii} + Y_2$$

$$Y_{jj} = Y_{jj} + Y_3$$

Appendix B.

The storage scheme used to store the nodal admittance matrix is described here. Only the non-zero elements, along with the appropriate pointers are stored as shown in Fig. B.1



a= real part of admittance
b= imaginary part of admittance
c= column indicator
d= next-in-row pointer
e= row indicator
f= next-in-column pointer

Fig. B.1

Diagonal elements occupy the first n locations of the vector. The pointers are used to chain together the elements that are in the same row or in the same column. The last element in a row, or in a column, points to the first element (the diagonal element) again. Inserting new elements is done by breaking the chain after the diagonal element and making a new chain via the new inserted element. A garbage pointer is used to indicate the first empty location in the vector. The empty locations are chained together and the garbage pointer is modified after any insertion or deletion. Note that the new empty locations are always added to the top of the garbage chain and are used first.

The above scheme has the following advantages:

1) Because of the location of the diagonal elements, no scanning is necessary to find these elements. Since the diagonal elements are used more often in any algorithm, this reduces the execution time.

2) To find an off-diagonal element, the scanning effort can be minimized through scanning either in the appropriate row or column whichever has less elements.

3) Since the pointers are stored close to the value of the elements, in a time-sharing computer system, the number of READ/WRITE operations on a secondary storage device will be less than that of conventional storage schemes.

4) The scheme can be easily programmed in FORTRAN through the use of the EQUIVALENCE statement.

Appendix C.

The data structure and the logic for forming and storing the triangularized Jacobian matrix is described here. Except for some changes in the working-row this is the scheme suggested by Tinney and Hart [9].

Each row of the Jacobian matrix is formed in the working-row and is later stored in the table of the Jacobian elements when the elimination process is performed upon that. The table of the Jacobian elements is a vector that contains all the elements of the triangularized Jacobian matrix and their pointers. The pointers are integer numbers which are positive for (P-Q) type buses and negative for (P-V) type buses. Another vector is used to store the integers that point to the starting location of each row. These pointers also follow the same rule for (P-Q) type and (P-V) type buses. The residuals and the diagonal elements are stored at the first locations where a row starts. For example, if the starting pointer of row i is a negative integer, then i is a single row (bus i is (P-V) type). In this case the first entry in the table of the Jacobian elements is ΔP_i, followed by an integer number j. If j is negative it represents a (P-V) type bus and it is followed by H_{ij} and another integer number representing another bus, while if j is positive it represents a (P-Q) type bus and is followed by H_{ij}, N_{ij} and another integer number. This notation is used for all the pointers in a single row. However, if row i is a double row (bus i is (P-Q) type) then the first three locations of this row will be ΔP_i, ΔQ_i and N_{ii} followed by the

first integer pointer in that row. In the case of double rows each negative pointer j in the row is followed by H_{ij}, J_{ij} and another pointer while each positive pointer is followed by H_{ij}, J_{ij}, N_{ij}, L_{ij} and another pointer. The end of each row is implied by the start of the next row.

Each row is formed and triangularized in the working-row, prior to being stored in the table of the Jacobian elements. Each element of the working-row consists of four locations, in which H, J, N and L are stored. It also has a column pointer. Its length is sufficient to store a full Jacobian row. 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 will have a zero pointer. At the end of the process only the elements with non-zero column pointers will be stored in the compact table of the Jacobian elements.