A COMPARISON OF DIGITAL METHODS APPLIED TO POWER FLOW STUDIES

by

Edwin Lindsey Dove

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
in
Electrical Engineering

APPROVED:

L. L. Grigsby
Dr. L. L. Grigsby

T. E. Bechert
Dr. T. E. Bechert

A. Wayne Bennett
Dr. A. Wayne Bennett

August, 1974

Blacksburg, Virginia

ACKNOWLEDGMENTS

This author wishes to thank Dr. T. E. Bechert, Dr. A. W. Bennett and Dr. L. L. Grigsby who served on his graduate committee. Special thanks are extended to Dr. L. L. Grigsby whose guidance, judgment and suggestions were of incalculable aid both in this writing and throughout the author's graduate study.

The author is grateful to Sandy Crigger, Charlotte Joyce, and especially Juanita Marlett, who showed great skill and much patience in typing this thesis.

Fellow graduate students and friends Tony Payne and Mike Walker deserve particular thanks for having to listen to the many problems encountered in this work, and then offering particularly edifying suggestions.

Finally, thanks is offered to Dr. L. L. Grigsby and the Energy Research Group at VPI&SU for their financial assistance.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

The topic considered in this thesis is the digital solution of the power flow problem in electric power systems.

Power flow studies, the backbone of power system analysis and design, are valuable for such reasons as system planning, outage studies, performance evaluation and selecting of an economic operating schedule. All of these studies require an examination of the voltages and powers in a network that is operating under some predetermined condition. The problem of finding these voltages and powers is identical to solving a coupled set of multivariable, nonlinear algebraic equations, the solution of which is no small task.

Research in the area of load flow studies on the digital computer began about 1955 with the classic work of Ward and Hale [1]. In this early paper, a successive displacement method (since called Gauss-Seidel) was presented to solve the load flow problem. Ward and Hale's paper led to a rash of new papers towards the later part of the 1950's and early 1960's, the most notable of which being that of VanNess and Griffin [2] which presented the basic Newton-Raphson method.

Since publication of these two classic papers, twenty years of experience has been gained and the techniques of solving the load flow problem have matured greatly. In recent years, the Newton-Raphson method (as modified by the very excellent work of Tinney and Hart [3]) has emerged to the front as the standard load flow algorithm, even to the point that many (if not most) electric energy systems engineers do not

question the algorithm's superiority to all other techniques. This author feels that this view needs to be examined very carefully.

There are two very valid reasons for this need in scrutiny. Firstly, there has been very few recent papers dedicated to a definitive comparison of the different algorithms to solve the load flow problem. One of the last of such papers is that of Sasson and Jaimes [4], written in 1967, in which the authors investigated the theoretical foundations of most of the methods known at that time. Since 1967, many improvements have been made on load flow algorithms, however, the results of these improvements have not been compiled. With this in mind, it is felt that a compilation and a comparison of the more significant results and methods would be of value to the electric power industry. Secondly, the electric power industry is growing and becoming more and more diversified. As of 1970, there were more than 3600 separate electric power systems in the U.S., of which only 100 produced 90 per cent of the total generation (for a more complete discussion, see Elgerd [5], Chapter 1). Thus, due to the many differences in individual power systems, the principle of accepting one algorithm as a standard needs to be further questioned. Different types of systems may need different load flow algorithms. Some factors that should be considered before choosing the correct technique to solve the power flow problem are:

i) Computer system to be used,

ii) Size of the power system being considered,

iii) Structural pecularities of the power system,

iv) The degree of experience of the energy system engineer who will be using the results of the studies, and

v) Purpose for which the load flow studies will be used.

Thus, a second objective of this paper is to investigate the principle of a standard load flow algorithm, and if it is found that one algorithm should not be accepted as a standard, guidelines will be set and criteria will be defined for determining the best algorithm to be used in solving the particular load flow problem at hand.

To realize the objectives outlined above, it is necessary to compare the signal algorithms that have been proposed. This comparison should be made in terms of accuracy, numerical stability, programming ease and computer storage and time required. Analytical comparisons are very difficult due to the great size of the problem considered, thus, one must rely on experimental comparisons. In order for these comparisons to be valid, they must be done by the same programmer on the same computer. In the work done for this thesis, each algorithm considered was programmed as a complete subroutine (in FORTRAN on IBM System 370/158), all subroutines being called from the same main program.

CHAPTER II

MATHEMATICAL FORMULATION OF THE LOAD FLOW PROBLEM

The following mathematical formulation of the load flow problem is presented only to establish notation to be used throughout the paper. For a more complete development, see Elgerd [5], Chapter 1, or Stagg and El-Abiad [6], Chapter 8. In the sequel, the nodal admittance formulation (in per unit quantities) will be assumed. Thus, given an n-bus system, the mathematical model can be written as:

$$I_k = \sum_{i=1}^{n} Y_{ki} V_i \quad ; \quad k = 1,2,\ldots,n \tag{2.1}$$

where $I_k \triangleq$ complex current being injected into the power system at bus k,

$V_k \triangleq$ complex voltage at bus k with respect to ground, and

$Y_{ki} \triangleq$ complex ki'th entry in the bus admittance matrix.

Assuming no voltage controlled buses in the system (this restriction will be lifted later in the paper), the following can be written:

$$I_k = \frac{(P_k + j\, Q_k)^*}{V_k^*} \quad ; \quad k = 1,2,\ldots,n \tag{2.2}$$

where $P_k \triangleq$ real power generated at bus k minus real power demanded at bus k (scheduled real power at bus k),

$Q_k \triangleq$ reactive power generated at bus k minus reactive power demanded at bus k (scheduled reactive power at bus k),

$j \triangleq \sqrt{-1}$,

$* \triangleq$ complex conjugate operator.

With these definitions in mind, (2.2) can be substituted into (2.1) yielding

$$\frac{(P_k + j\, Q_k)^*}{V_k^*} = \sum_{i=1}^{n} Y_{ki}\, V_i \; ; \; k = 1,2,\ldots,n. \qquad (2.3)$$

Assuming that the slack bus corresponds to bus #1, (2.3) can be written as:

$$P_k - j\, Q_k = V_k^* \sum_{i=1}^{n} Y_{ki}\, V_i \; ; \; k = 2,3,\ldots,n \qquad (2.4)$$

and

$$P_1 - j\, Q_1 = V_1^* \sum_{i=1}^{n} Y_{1i}\, V_i \; . \qquad (2.5)$$

All load flow algorithms (in the nodal admittance frame of reference) are dedicated to solving Equations (2.4) for the bus voltages V_k. Note that (2.4) is a coupled set of multivariable, nonlinear algebraic equations. To date, no closed-form analytic solution has been found, thus one must rely on some type of iterative numerical technique.

CHAPTER III

DISCUSSION OF THE TWO BASIC SOLUTION ALGORITHMS

As mentioned in the introduction, there are two basic methods used to solve the power flow problem. In this chapter, both methods will be outlined and both qualitative and quantitative comparisons will be given.

3-1 GAUSS-SEIDEL METHOD

In 1956, Ward and Hale [1] published their work on digitally solving the load flow problem by using a successive displacement algorithm. Since this original work, the method has been modified slightly, resulting in what is known as the Gauss-Seidel algorithm. The following mathematical development is presented only to establish notation.

Equation (2.4) can be rewritten as:

$$V_k = \left(1/Y_{kk}\right)\left[\frac{P_k - j\,Q_k}{V_k^*} - \sum_{\substack{i=1 \\ i\neq k}}^{n} Y_{ki}\,V_i\right]; \quad k = 2,3,\ldots n. \tag{3-1.1}$$

Assuming some starting values for all voltages V_k, Equation (3-1.1) can be written as:

$$V_k^{(\ell+1)} = \frac{1}{Y_{kk}}\left[\frac{P_k - j\,Q_k}{V_k^{(\ell)\,*}} - \sum_{i=1}^{k-1} Y_{ki}\,V_i^{(\ell+1)} - \sum_{i=k+1}^{n} Y_{ki}\,V_i^{(\ell)}\right]; \quad k = 2,3,\ldots,n \tag{3-1.2}$$

where the superscript ℓ denotes the ℓ'th trial value or iterate. We can define the following:

$$A_k \triangleq \frac{P_k - j\,Q_k}{Y_{kk}}; \quad k = 2,3,\ldots,n \tag{3-1.3}$$

and

$$B_{ki} \triangleq \frac{Y_{ki}}{Y_{kk}}; \quad \begin{matrix} k = 2,3,\ldots,n \\ i = 1,2,3,\ldots,n,\ i\neq k. \end{matrix} \tag{3-1.4}$$

Substituting Equation (3-1.3) and (3-1.4) into (3-1.2) yields

$$V_k^{(\ell+1)} = \frac{A_k}{(V_k^{(\ell)})^*} - \sum_{i=1}^{k-1} B_{ki}\, V_i^{(\ell+1)} - \sum_{i=k+1}^{n} B_{ki}\, V_i^{(\ell)}; \quad k = 2,3,\ldots,n. \tag{3-1.5}$$

Equation (3-1.5) is solved recursively until

$$\left|\Delta V_{k,acc}^{(\ell+1)}\right| \triangleq \left|V_{k,acc}^{(\ell+1)} - V_k^{(\ell)}\right| \leq \varepsilon \qquad ; k = 2,3,..,n \tag{3-1.6}$$

where ε is some prespecified error tolerance, and $V_{k,acc}^{(\ell+1)}$ is defined as

$$V_{k,acc}^{(\ell+1)} \triangleq V_k^{(\ell)} + \alpha[V_k^{(\ell+1)} - V_k^{(\ell)}]. \tag{3-1.7}$$

The constant α is known as the accelerating factor. Typically, α is in the range $1.5 \leq \alpha \leq 1.8$.

In the Gauss-Seidel algorithm, voltage controlled busses are handled in the following manner:

let $$V_k^{(\ell)} \triangleq a_k^{(\ell)} + j\, b_k^{(\ell)} = |V_k^{(\ell)}|\exp(\delta_k^{(\ell)}), \tag{3-1.8}$$

$|V_{k,sch}| \triangleq$ scheduled voltage magnitude at bus k,

$$a_{k,sch}^{(\ell)} \triangleq |V_{k,sch}|\cos\,\delta_k^{(\ell)}, \tag{3-1.9}$$

and $$b_{k,sch}^{(\ell)} \triangleq |V_{k,sch}|\sin\,\delta_k^{(\ell)}. \tag{3-1.10}$$

At a voltage controlled bus k, it is required that (see Stagg and El-Abiad [6])

$$P_k - j\, Q_k = Y_{kk}[(a_{k,sch}^{(\ell)})^2 + (b_{k,sch}^{(\ell)})^2] + (a_{k,sch}^{(\ell)} + j\, b_{k,sch}^{(\ell)})^* \left[\sum_{i=1}^{k-1} Y_{ki}\, V_i^{(\ell+1)} + \sum_{i=k+1}^{n} Y_{ki}\, V_i^{(\ell)}\right]. \tag{3-1.11}$$

The reactive power Q_k is then checked to see if it violates the limits specified at this bus k. If so, Q_k is set to the limiting value and bus k is treated as a load bus for the remainder of the iteration. If not, $V_k^{(\ell)}$ is replaced by $|V_{k,sch}|\exp(\delta_k^{(\ell)})$, and the iteration continues according to Equation (3-1.5).

There are three major objections to using the Gauss-Seidel algorithm as presented above, these being:

i) the algorithm shows numerical instability for certain systems, these being systems containing series capacitive lines (as can arise from series capacitive compensation or from equivalents of 3-winding transformers) or busses to which both high and low impedance branches are connected.

ii) the number of iterations required to solve a given power flow problem is dependent on the accelerating factor α given in Equation (3-1.7). Furthermore, the optimum accelerating factor (optimum in that this value of α will require the fewest number of iterations for a complete solution) varies greatly from problem to problem. Since the power flow problem is nonlinear, there is no way to determine the optimum α except by trial and error.

iii) it has been reported (see especially Tinney and Hart [3]) that the Gauss-Seidel method converges relatively slowly for large order systems.

3-2 NEWTON-RAPHSON METHOD

In 1961, VanNess and Griffin [2] published their work on solving the load flow problem by using a Newton-like elimination method. The method was further modified by Tinney and Hart [3], resulting in the present day Newton-Raphson algorithm.

There are two forms of the Newton-Raphson algorithm; both forms will be briefly developed and qualitative comparisons will be given. A more complete discussion is given in Tinney and Hart [3], Elgerd [5] or Stagg and El-Abiad [6].

The first form considered is known as the rectangular form. The development goes as follows: the power flow equations can be written (at the ℓ'th iterate)

$$P_k^{(\ell)} - j\, Q_k^{(\ell)} = (V_k^{(\ell)})^* \sum_{i=1}^{n} Y_{ki}\, V_i^{(\ell)} \quad ; \; k = 2,3,\ldots,n. \tag{3-2.1}$$

Now let

$$V_k^{(\ell)} = a_k^{(\ell)} + j\, b_k^{(\ell)} = |V_k^{(\ell)}| \exp(\delta_k^{(\ell)}) \tag{3-2.2}$$

and

$$Y_{ki} = G_{ki} + j\, B_{ki} \; . \tag{3-2.3}$$

Now P_k and Q_k (scheduled real and reactive powers at bus k) differ from $P_k^{(\ell)}$ and $Q_k^{(\ell)}$ by $\Delta P_k^{(\ell)}$ and $\Delta Q_k^{(\ell)}$ respectively, i.e.

$$\Delta P_k^{(\ell)} \triangleq P_k - P_k^{(\ell)} \quad ; \; k = 2,3,\ldots,n \tag{3-2.4}$$

and

$$\Delta Q_k^{(\ell)} \triangleq Q_k - Q_k^{(\ell)} \quad ; \; k = 2,3,\ldots,n. \tag{3-2.5}$$

With these equations in mind, the following can be identified:

$\Delta \underline{P}^{(\ell)} \triangleq$ (n-1) vector whose k'th entry is given by Equation (3-2.4)

and $\Delta\underline{Q}^{(\ell)} \triangleq$ (n-1) vector whose k'th entry is given by Equation (3-2.5).

The load flow equations can be linearized by expanding Equation (3-2.1) in a Taylor Series and retaining only the linear portion of the expansion resulting in the following:

$$\begin{bmatrix} \Delta\underline{P}^{(\ell)} \\ \Delta\underline{Q}^{(\ell)} \end{bmatrix} = \underline{J}^{(\ell)} \begin{bmatrix} \Delta\underline{a}^{(\ell)} \\ \Delta\underline{b}^{(\ell)} \end{bmatrix} = \begin{bmatrix} \underline{J}_{11}^{(\ell)} & \underline{J}_{12}^{(\ell)} \\ \underline{J}_{21}^{(\ell)} & \underline{J}_{22}^{(\ell)} \end{bmatrix} \begin{bmatrix} \Delta\underline{a}^{(\ell)} \\ \Delta\underline{b}^{(\ell)} \end{bmatrix}. \tag{3-2.6}$$

Equation (3-2.6) can be solved recursively for $\Delta\underline{a}^{(\ell)}$ and $\Delta\underline{b}^{(\ell)}$ until $|a_k^{(\ell+1)} - a_k^{(\ell)}| < \varepsilon_1$ and $|b_k^{(\ell+1)} - b_k^{(\ell)}| < \varepsilon_2$ for $k = 2,3,\ldots,n$. The constants ε_1 and ε_2 are prespecified error tolerances (sometimes the convergence criteria is applied to $\Delta P_k^{(\ell)}$ and $\Delta Q_k^{(\ell)}$ in a likewise manner). The variables $a_k^{(\ell+1)}$ and $b_k^{(\ell+1)}$ are given by:

$$a_k^{(\ell+1)} = a_k^{(\ell)} + \Delta a_k^{(\ell)} \tag{3-2.7}$$

and

$$b_k^{(\ell+1)} = b_k^{(\ell)} + \Delta b_k^{(\ell)} \tag{3-2.8}$$

where $\Delta a_k^{(\ell)}$ and $\Delta b_k^{(\ell)}$ are the k'th entries of $\Delta\underline{a}^{(\ell)}$ and $\Delta\underline{b}^{(\ell)}$ respectively. The matrix $\underline{J}^{(\ell)}$ in Equation (3-2.6) is commonly called the Jacobian matrix. The kj'th entry in each of the four submatrices $\underline{J}_{11}^{(\ell)}$, $\underline{J}_{12}^{(\ell)}$, $\underline{J}_{21}^{(\ell)}$ and $\underline{J}_{22}^{(\ell)}$ is given by (for a rigorous derivation of the following equations, see VanNess [7]):

$$(J_{11})_{kj}^{(\ell)} \triangleq \frac{\partial P_k^{(\ell)}}{\partial a_j^{(\ell)}} = a_k^{(\ell)} G_{kj} + b_k^{(\ell)} B_{kj} \; ; \; j,k = 2,3,\ldots,n \text{ if } k \neq j \tag{3-2.9}$$

$$= a_k^{(\ell)} G_{kk} + b_k^{(\ell)} B_{kk} + c_k^{(\ell)} \; ; \; k = 2,3,\ldots,n. \tag{3-2.10}$$

$$(J_{12})^{(\ell)}_{kj} \triangleq \frac{\partial P_k^{(\ell)}}{\partial b_j^{(\ell)}} = -a_k^{(\ell)} B_{kj} + b_k^{(\ell)} G_{kj} \; ; \; j,k = 2,3,\ldots,n \text{ if } k \neq j \tag{3-2.11}$$

$$= -a_k^{(\ell)} B_{kk} + b_k^{(\ell)} G_{kk} + d_k^{(\ell)} \; ; \; k = 2,3,\ldots,n. \tag{3-2.12}$$

$$(J_{21})^{(\ell)}_{kj} \triangleq \frac{\partial Q_k^{(\ell)}}{\partial a_j^{(\ell)}} = -a_k^{(\ell)} B_{kj} + b_k^{(\ell)} G_{kj} \; ; \; j,k = 2,3,\ldots,n \text{ if } k \neq j \tag{3-2.13}$$

$$= -a_k^{(\ell)} B_{kk} + b_k^{(\ell)} G_{kk} - d_k^{(\ell)} \; ; \; k = 2,3,\ldots,n. \tag{3-2.14}$$

$$(J_{22})^{(\ell)}_{kj} \triangleq \frac{\partial Q_k^{(\ell)}}{\partial b_j^{(\ell)}} = -a_k^{(\ell)} G_{kj} - b_k^{(\ell)} B_{kj} \; ; \; j,k = 2,3,\ldots,n \text{ if } k \neq j \tag{3-2.15}$$

$$= -a_k^{(\ell)} G_{kk} - b_k^{(\ell)} B_{kk} + c_k^{(\ell)} \; ; \; k = 2,3,\ldots,n. \tag{3-2.16}$$

In Equations (3-2.9) through (3-2.16), the bus current $I_k^{(\ell)}$ is given by

$$I_k^{(\ell)} \triangleq c_k^{(\ell)} + j\, d_k^{(\ell)} = \frac{P_k^{(\ell)} - j\, Q_k^{(\ell)}}{(V_k^{(\ell)})^*}. \tag{3-2.17}$$

With the above background, the Newton-Raphson solution algorithm for the rectangular form can be outlined as:

i) calculate $\Delta \underline{P}^{(\ell)}$ and $\Delta \underline{Q}^{(\ell)}$,

ii) calculate $I_k^{(\ell)}$ from (3-2.17), $k = 2,3,\ldots,n$,

iii) calculate the Jacobian $\underline{J}^{(\ell)}$,

iv) solve (3-2.6) for $\Delta \underline{a}^{(\ell)}$ and $\Delta \underline{b}^{(\ell)}$,

v) calculate $a_k^{(\ell+1)}$ and $b_k^{(\ell+1)}$ from (3-2.7) and (3-2.8) respectively, and

vi) check to see if convergence has been achieved; if so, stop the iteration, if not, begin with i).

Voltage controlled busses are handled by modifying Equation (3-2.6) to read (assuming without loss of generality that voltage controlled busses are numbered last):

$$\begin{bmatrix} \underline{J}_{11}^{(\ell)} & \underline{J}_{12}^{(\ell)} \\ \underline{J}_{21}^{(\ell)} & \underline{J}_{22}^{(\ell)} \\ \underline{J}_{31}^{(\ell)} & \underline{J}_{32}^{(\ell)} \end{bmatrix} \begin{bmatrix} \Delta\underline{a}^{(\ell)} \\ \Delta\underline{b}^{(\ell)} \end{bmatrix} = \begin{bmatrix} \Delta\underline{P}^{(\ell)} \\ \Delta\underline{Q}^{(\ell)} \\ \Delta|\underline{V}^{(\ell)}|^2 \end{bmatrix} \tag{3-2.18}$$

where $\Delta\underline{P}^{(\ell)}$, $\underline{J}_{11}^{(\ell)}$ and $\underline{J}_{12}^{(\ell)}$ are as before, and $\underline{J}_{21}^{(\ell)}$, $\underline{J}_{22}^{(\ell)}$ and $\Delta\underline{Q}^{(\ell)}$ are as before except $k = 2,3,\ldots,n-p$, where p is the number of voltage controlled busses. A typical entry in $\Delta|\underline{V}^{(\ell)}|^2$ is

$$\Delta|V_k^{(\ell)}|^2 = |V_{k,sch}|^2 - |V_k^{(\ell)}|^2 \; ; \; k = n-p+1,\ldots,n. \tag{3-2.19}$$

It has been shown (see VanNess [7]) that:

$$(J_{31})_{kj}^{(\ell)} \triangleq \frac{\partial|V_k^{(\ell)}|^2}{\partial a_j^{(\ell)}} = 0 \text{ if } j\neq k \tag{3-2.20}$$

$$= 2\, a_k^{(\ell)} \text{ if } j=k, \tag{3-2.21}$$

$$(J_{32})_{kj}^{(\ell)} \triangleq \frac{\partial|V_k^{(\ell)}|^2}{\partial b_j^{(\ell)}} = 0 \text{ if } j\neq k \tag{3-2.22}$$

$$= 2\, b_k^{(\ell)} \text{ if } j=k. \tag{3-2.23}$$

The reactive power constraints at bus k must be satisfied if bus k is a voltage controlled bus. This is done in one of two ways:

i) a complete solution is obtained, and $Q_k^{(\ell)}$ is tested to see if it violates a given constraint; if so, $Q_k^{(\ell)}$ is set to

the violated constraint, and the iterative procedure is continued, treating this bus as a load bus.

ii) required adjustments are made at the end of every iteration until a final solution is obtained. After reactive power has been constrained to a limit, it is checked every following iteration to see if the imposed limit is still necessary.

The second form considered is known as the polar form. Considering Equations (3-2.1) through (3-2.5) along with Equation (3-2.17), the approximate (linear) model considered for this form is

$$\begin{bmatrix} \Delta \underline{P}^{(\ell)} \\ \Delta \underline{Q}^{(\ell)} \end{bmatrix} = \underline{J} \begin{bmatrix} \Delta \underline{\delta}^{(\ell)} \\ \Delta |\underline{\hat{V}}|^{(\ell)} \end{bmatrix} = \begin{bmatrix} \underline{J}_{11}^{(\ell)} & \underline{J}_{12}^{(\ell)} \\ \underline{J}_{21}^{(\ell)} & \underline{J}_{22}^{(\ell)} \end{bmatrix} \begin{bmatrix} \Delta \underline{\delta}^{(\ell)} \\ \Delta |\underline{\hat{V}}|^{(\ell)} \end{bmatrix} , \qquad (3\text{-}2.24)$$

where a typical entry in $\Delta|\underline{\hat{V}}|^{(\ell)}$ is

$$\Delta|\hat{V}_k|^{(\ell)} \triangleq \frac{\Delta|V_k^{(\ell)}|}{|V_k^{(\ell)}|} , \qquad (3\text{-}2.25)$$

and $\Delta\underline{\delta}^{(\ell)}$ is the (n-1) vector of incremental bus voltage angles. Typical entries in the polar form Jacobian $\underline{J}$ are (see VanNess [7]):

$$(J_{11})_{kj}^{(\ell)} \triangleq \frac{\partial P_k^{(\ell)}}{\partial \delta_j^{(\ell)}} = c_j^{(\ell)} b_k^{(\ell)} - d_j^{(\ell)} a_k^{(\ell)}; \; j,k=2,3,\ldots,n \text{ if } k \neq j \qquad (3\text{-}2.26)$$

$$= -Q_k^{(\ell)} - B_{kk}\left|V_k^{(\ell)}\right|^2; \; k=2,3,\ldots,n \text{ if } k=j \; . \qquad (3\text{-}2.27)$$

$$(J_{12})^{(\ell)}_{kj} \triangleq \frac{\partial P_k^{(\ell)}}{\partial \left|V_j^{(\ell)}\right|} \left|V_j^{(\ell)}\right| = c_j^{(\ell)}\, a_k^{(\ell)} + d_j^{(\ell)}\, b_k^{(\ell)};\ j,k=2,3,\ldots,n \text{ if } k\neq j \tag{3-2.28}$$

$$= P_k^{(\ell)} + G_{kk}\left|V_k^{(\ell)}\right|^2;\ k=2,3,\ldots,n \text{ if } k=j\ . \tag{3-2.29}$$

$$(J_{21})^{(\ell)}_{kj} \triangleq \frac{\partial Q_k^{(\ell)}}{\partial \delta_j^{(\ell)}} = -c_j^{(\ell)}\, a_k^{(\ell)} - d_j^{(\ell)}\, b_k^{(\ell)};\ j,k=2,3,\ldots,n \text{ if } k\neq j \tag{3-2.30}$$

$$= P_k^{(\ell)} - G_{kk}\left|V_k^{(\ell)}\right|^2;\ k=2,3,\ldots,n \text{ if } k=j\ . \tag{3-2.31}$$

$$(J_{22})^{(\ell)}_{kj} \triangleq \frac{\partial Q_k^{(\ell)}}{\partial \left|V_j^{(\ell)}\right|} \left|V_j^{(\ell)}\right| = c_j^{(\ell)}\, b_k^{(\ell)} - d_j^{(\ell)}\, a_k^{(\ell)};\ j,k=2,3,\ldots,n \text{ if } k\neq j \tag{3-2.32}$$

$$= Q_k^{(\ell)} - B_{kk}\left|V_k^{(\ell)}\right|^2;\ k=2,3,\ldots,n \text{ if } k=j. \tag{3-2.33}$$

The Newton-Raphson solution algorithm for the polar form follows the same outline as was given for the rectangular form, and thus will not be repeated.

In the polar form, if bus k is a voltage controlled bus, then $|V_k|$ is known, and thus the row and column corresponding to $\Delta Q_k^{(\ell)}$ and $\Delta\left|\hat{V}_k\right|^{(\ell)}$ can be deleted.

In comparing the two forms just developed, one sees that the polar form requires less storage than the rectangular form. Given an n bus system with p voltage controlled busses, the rectangular form requires the solution of (and thus storage of) 2(n-1) linear equations, while only 2(n-1 - p/2) or 2n-p-2 linear equations are required to be solved with the polar form. In other words, the rectangular Jacobian is (2n-2) x (2n-2) while the polar Jacobian is (2n-2-p) x (2n-2-p). However, it should be noted that what the polar form gains in storage

savings is partially lost in the additional mathematical operations required. Most computers have complex arithmetic capability which assumes that complex numbers are stored and operated on in rectangular form. Thus, additional mathematical functions (square roots, trigonometric functions, etc.) must be evaluated to manipulate $\Delta\underline{\delta}^{(\ell)}$ and $\Delta|\underline{\hat{V}}|^{(\ell)}$.

There are two major objections to using the Newton-Raphson algorithm in either form, these being:

i) in addition to the variables needed in the basic load flow problem (Equation (2.4)), vectors such as $\Delta\underline{P}^{(\ell)}$, $\Delta\underline{Q}^{(\ell)}$, $\Delta\underline{a}^{(\ell)}$ and $\Delta\underline{b}^{(\ell)}$ (or $\Delta\underline{\delta}^{(\ell)}$ and $\Delta|\underline{\hat{V}}|^{(\ell)}$) must be stored, not to mention the Jacobian. This added storage requirement is very significant when considering large systems; the additional storage requirements grows quadratically with problem size due to the square Jacobian.

ii) programming (and program modification) is very difficult and complex. Even if the electric energy systems engineer is not required to do the final coding, he is required to understand every detail of the algorithm. This is no small task for the Newton-Raphson algorithm.

3-3 QUALITATIVE AND QUANTITATIVE DISCUSSION

Both the Gauss-Seidel and the Newton-Raphson algorithms (and all other algorithms programmed for this thesis) were programmed in FORTRAN on VPI&SU's IBM 370/158. Each algorithm was written as a complete subroutine in that only calculations indigenous to the algorithm were done in the subroutine; calculations common to all algorithms were done in the calling program. Specifically, all input/output, line flows and losses, and the calculation of the bus admittance matrix $\underline{Y}$ was done in the calling program. For all algorithms considered, the prespecified voltage tolerance used was $\varepsilon = 0.0005$. All algorithms started from a flat start (1.0 + j0.0 per unit), excepting of course the slack bus voltage.

Table I shows a comparison of the two algorithms shown thus far. The solution time (in seconds) includes input/output, compilation, and all calculations. The storage required (in kilobytes) includes all vectors, matrices, and the generated object code. The four test systems considered are the standard IEEE load flow systems. The Newton-Raphson algorithm in Table I is the rectangular form.

Considering the four test cases in Table I, the following observations can be made:

i) The Gauss-Seidel algorithm took much less storage than the Newton-Raphson algorithm. This result should be expected considering the size of the Jacobian required for Newton-Raphson. (For the 118 bus system, the Jacobian had 54,756 entries.)

Table I

Comparison of Gauss-Seidel and Newton-Raphson

	GAUSS-SEIDEL				NEWTON-RAPHSON		
System	Time	Storage	Iterations	opt. α	Time	Storage	Iterations
14 Bus	1.07	48	14	1.6	1.69	54	4
30 Bus	3.11	54	26	1.7	10.71	70	5
57 Bus	10.64	76	38	1.7	55.86	128	4
118 Bus	107.37	166	58	1.8	560.8	384	5

Time in seconds

Storage in kilobytes

ii) Quite unexpectedly (considering the comments from recent load flow literature), Gauss-Seidel required much less time than did the Newton-Raphson. In examining closely the 560.8 seconds required by Newton-Raphson to solve the 118 bus test system, it was found that over 80% of the total time was in solving the Jacobian Equation (3-2.6). Different linear equation solution methods were tested, the fastest being Gauss elimination. (No pivoting was necessary since the Jacobian was always strongly diagonally dominant.)

iii) The Gauss-Seidel algorithm took many more iterations than did the Newton-Raphson method, however, Gauss-Seidel took considerably fewer number of calculations per iterate. The number of required iterations for Gauss-Seidel increased considerably with problem size, while with Newton-Raphson, the number of required iterations was independent of the problem size. This would lead one to believe that the Newton-Raphson algorithm possessed the superior convergence characteristics. Indeed, this has been reported in the literature [3][8][9]. As a test of the numerical stability of the two methods, series capacitance was added to different lines in the 14 bus system. (From Chapter 3,Section 1, one of the weaknesses of the Gauss-Seidel method is its inability to solve systems with series capacitive lines.) Neither the Gauss-Seidel nor the Newton-Raphson algorithms had any difficulty solving these systems. This should not be taken as proof that the Gauss-Seidel algorithm

does not show instability when solving systems with series capacitive lines; the only conclusions that can be drawn is that it had no problem solving these particular systems.

iv) The optimal accelerating factor (opt. α) for the Gauss-Seidel algorithm varied between $1.6 \leq$ opt. $\alpha \leq 1.8$ for the four test cases considered. Note that opt. α increased with increasing problem size. According to recent load flow literature ([9], [8], [6]) this should not be taken as the general case, i.e. opt. α does not generally increase with increasing problem size. For a network being solved for the first time, a safe guess for an accelerating factor would be $\alpha = 1.6$ to $\alpha = 1.7$. (The accelerating factor can be chosen such that convergence cannot be obtained. Thus, one must use some caution in choosing α.) As was mentioned in Chapter 3, Section 1, the only known way of determining opt. α is by trial and error.

v) By programming separately the two methods for checking reactive constraints at voltage controlled busses in the Newton-Raphson algorithm (Chapter 3, Section 2), it was found that an alteration of the second method was best. The reactive constraints should be checked only after the second or third iteration and each iteration thereafter, thus allowing the algorithm to "run unconstrained" towards a feasible solution for the first two or three iterations. This alteration saved as many as two or three iterations when compared with the first method mentioned in Chapter 3, Section 2.

vi) Programming the Gauss-Seidel algorithm was very much less difficult than programming Newton-Raphson. With the relative ease of writing Gauss-Seidel came an added flexibility not possible with Newton-Raphson. In addition, the generated object code for the Newton-Raphson algorithm was considerably longer than for the Gauss-Seidel, thus consuming more computer storage not available for data.

vii) Once the Gauss-Seidel algorithm along with the necessary line and bus data are in the memory of the computer, then the algorithm needs no more storage to compute a load flow. This is not the case with the Newton-Raphson algorithm; the Jacobian $\underline{J}$ along with $\Delta\underline{P}$, $\Delta\underline{Q}$, $\Delta\underline{\delta}$ and $\Delta|\underline{\hat{V}}|$ (or $\Delta\underline{a}$ and $\Delta\underline{b}$) must be generated after all bus and line data is in the computer. In other words, if the necessary data can be loaded into the machine, then Gauss-Seidel has enough storage space to solve the load flow problem.

In summary, for the four test systems considered, the Gauss-Seidel algorithm, as compared to the Newton-Raphson algorithm, required much less storage, less solution time, and was relatively less complicated to program and modify. Once the data was loaded into the computer (along with the program) Gauss-Seidel had enough region to solve the load flow equations,which is not the case for the Newton-Raphson algorithm. Finally, the optimum accelerating factor for the Gauss-Seidel algorithm remained in the range $1.6 \leq \text{opt. } \alpha \leq 1.8$.

CHAPTER IV

ALTERNATIVE SOLUTION ALGORITHMS

A number of variations of the two basic algorithms have been proposed to compensate for some of the shortcomings mentioned in Chapter III. A few of the more significant modification will be discussed and compared.

4-1 ALTERNATIVE FORMS OF THE GAUSS-SEIDEL ALGORITHM

Many methods have been proposed to improve the Gauss-Seidel algorithm. In 1969, Treece [8] proposed a form of the Gauss-Seidel algorithm (called the Bootstrap method) claiming to give speeded convergence. Heuristically, the Bootstrap method replaces the true (scheduled) powers P_k and Q_k in Equation (3-1.2) by some pseudo-powers derived from power mismatch relations. A few of the shortcomings of this method are an added storage requirment (with the Bootstrap method, the pseudo-powers must be saved), and, like the regular Gauss-Seidel algorithm, there is a constant "boost" (much like the accelerating factor α) that typically ranges from 0.75 to 0.95. A poor choice of the constant "boost" can make the method diverge.

As suggested in Chapter 3, one of the main reasons that the Newton-Raphson method has gathered wide acceptance as the standard load flow algorithm is that the method has no trouble solving networks containing series capacitive lines, which is not the case for the Gauss-Seidel algorithm. In 1973, a paper was published by Podmore and Undrill [9] in which these authors presented a modified nodal iterative technique (modified Gauss-Seidel) especially capable of handling series capacitive branches. Briefly, the development of the modified nodal iterative technique (MNIT) proceeds in three steps (assuming no voltage controlled busses):

i) derive a secondary adjustment to be applied at nodes j immediately adjacent to node k when a primary adjustment is being made at node k,

ii) revise the primary adjustment formula (Equation (3-1.7) to account for the secondary adjustments in i),

iii) revise the acceleration scheme to take best advantage of the modified voltage adjustment processes.

Mathematically, the development goes as follows (only the primary results of the original work [9] will be presented; the original work should be consulted for more detail):

$$\text{let} \qquad V_k^{eff} \triangleq Y_{kk} - \sum_{\substack{j=1 \\ j\neq k}}^{n} \frac{Y_{kj}\, Y_{jk}}{Y_{kk}} \quad ; \; k = 2,3,\ldots,n. \tag{4-1.1}$$

At the ℓ'th iterate, the correction to bus k is:

$$\Delta V_k^{(\ell+1)} = \left(\frac{1}{Y_k^{eff}}\right)\left[\left(\frac{P_k + j\, Q_k}{V_k^{(\ell)}}\right)^* - V_k^{(\ell)}\, Y_{kk} - \sum_{i=1}^{k-1} V_i^{(\ell+1)} Y_{ki} - \sum_{i=k+1}^{n} V_i^{(\ell)}\, Y_{ki}\right] \tag{4-1.2}$$

$$\text{and} \qquad V_k^{(\ell+1)} = \hat{\alpha}\left[V_k^{(\ell)} - V_k^{save} + \Delta V_k^{(\ell+1)}\right] + V_k^{save}\,, \tag{4-1.3}$$

where $\hat{\alpha}$ is an accelerating factor. (V_k^{save} will be defined later.) Equation (4-1.3) is the revised primary adjustment formula. At all busses j immediately adjacent to bus k, the secondary adjustment formula is:

$$V_j^{(\ell+1)} = V_j^{(\ell)} - \frac{Y_{jk}}{Y_{jj}}\left[V_k^{(\ell+1)} - V_k^{save}\right]; \; j = 2,3,\ldots,\, k-1, \tag{4-1.4}$$

$$\text{and} \quad V_j^{(\ell)} = V_j^{(\ell)} - \frac{Y_{jk}}{Y_{jj}}\left[V_k^{(\ell+1)} - V_k^{save}\right]; \; j = k+1,\ldots,n. \tag{4-1.5}$$

Finally,

$$V_k^{save} = V_k^{(\ell+1)} \quad . \tag{4-1.6}$$

(Note: the equality sign in Equation (4-1.4) and (4-1.5) does not represent mathematical equality, but rather "replacement.")

Thus, the solution outline is (after $\underline{Y}^{eff}$ (4-1.1) has been calculated)

i) calculate $\Delta V_k^{(\ell+1)}$ from (4-1.2)

ii) make the primary voltage correction (4-1.3) at bus k,

iii) make the secondary voltage correction (4-1.4) or (4-1.5) at bus j immediately adjacent to bus k, and

iv) store $V_k^{(\ell+1)}$ in V_k^{save} (Equation (4-1.6)).

If voltage controlled busses are considered (as is certainly the case in all practical systems), the MNIT then takes on the following changes:

i) when bus j, adjacent to bus k, is a voltage controlled bus, the corresponding $\frac{Y_{kj}\ Y_{jk}}{Y_{kk}}$ term in (4-1.1) is taken as zero; this has the effect of canceling all secondary corrections at this adjacent bus j,

ii) primary corrections at voltage controlled busses are made, after adjustment of reactive power, by either the conventional method (see voltage control discussion - Chapter 3, Section 1), or by Equations (4-1.2) and (4-1.3) considering i) above.

There are two main drawbacks in using the MNIT of Podmore and Undrill. Firstly, calculation of $\underline{Y}^{eff}$ in Equation (4-1.1) requires much additional computation time. (A close examination reveals that on the order of $3n^2$ multiply-divisions are required to calculate $\underline{Y}^{eff}$, a significant increase in solution time for large n.) Secondly, storing $\underline{Y}^{eff}$ and $\underline{V}^{save}$ requires 4n more storage locations than the regular Gauss-Seidel method.

Table II shows experimental results using the four standard IEEE test cases. The programming of the MNIT was done under the same conditions as mentioned at the beginning of Chapter 3, Section 2. Considering Table II, the following observations can be made:

i) In all cases, Gauss-Seidel took less solution time than did the MNIT. One of the reasons for this added solution time is the necessity to calculate $\underline{Y}^{eff}$. Another more subtle reason is that the MNIT suppresses all secondary corrections at voltage controlled busses, thus losing much of its proposed benefit with systems with many voltage controlled busses. The 118 bus, 57 bus, 30 bus and 14 bus systems have 53, 6, 5 and 4 voltage controlled busses respectively. As noted by Podmore and Undrill [9], this suppression of secondary corrections would account for the extremely long solution time for the 118 bus test system. In order to improve convergence for this system, Podmore and Undrill converted a number of the synchronous condensors to fixed static capacitors.

ii) The required computer region for the MNIT increased slightly over the Gauss-Seidel method due to the storage of $\underline{Y}^{eff}$ and $\underline{V}^{save}$. (The reason for the 2K increment in storage in Table II is that the IBM 370/158 allocates storage only in 2K increments.)

iii) The optimal accelerating factor opt. $\hat{\alpha}$ was within the range $1.14 \leq$ opt. $\hat{\alpha} \leq 1.18$ for all four cases considered. As in

Table II

Comparison of Gauss-Seidel and Modified Nodal Iterative Technique

	GAUSS-SEIDEL				MNIT			
System	Time	Storage	Iterations	opt. α	Time	Storage	Iterations	opt. $\hat{\alpha}$
14 bus	1.07	48	14	1.6	1.24	50	11	1.14
30 bus	3.11	54	26	1.7	5.16	56	24	1.18
57 bus	10.64	76	38	1.7	30.31	78	45	1.14
118 bus	107.37	166	58	1.8	310.34	168	112	1.18

Time in seconds

Storage in kilobytes

the Gauss-Seidel algorithm, $\hat{\alpha}$ could be chosen to give a divergent system of equations (typically, with $\hat{\alpha} > 1.22$ the solution method diverged). As in the Gauss-Seidel method, the opt. $\hat{\alpha}$ was found by trial and error . For a new system, to insure convergence, a conservative $\hat{\alpha}$ of 1.14 to 1.16 should be used.

iv) the main reason for considering this method is its ability to consistently solve networks with series capacitive lines. The 14-bus system was perturbed by making one or more lines highly series capacitive. Neither the Gauss-Seidel nor the MNIT had trouble solving these systems. However, it should be noted that in the discussion of the original paper [9], Messrs. Sachdev and Billinton verify the ability of the MNIT to solve systems that the conventional Gauss-Seidel method cannot.

In summary, in comparing the conventional Gauss-Seidel algorithm with the MNIT, it was found that the Gauss-Seidel method took less storage and less computation time for a complete solution. However, it has been reported (there is no reason to doubt these reports) that the MNIT will solve certain systems that Gauss-Seidel cannot.

4-2 ALTERNATE FORMS OF THE NEWTON-RAPHSON ALGORITHM

Many variations of the Newton-Raphson algorithm have been proposed to reduce both required storage and solution time. One of the simplest of all alternate forms is to neglect the off diagonal submatrices $\underline{J}_{12}^{(\ell)}$ and $\underline{J}_{21}^{(\ell)}$ in Equation (3-2.24), resulting in:

$$\Delta\underline{P}^{(\ell)} = \underline{J}_{11}^{(\ell)} \quad \Delta\underline{\delta}^{(\ell)} \tag{4-2.1}$$

and

$$\Delta\underline{Q}^{(\ell)} = \underline{J}_{22}^{(\ell)} \quad \Delta|\underline{\hat{V}}|^{(\ell)} \tag{4-2.2}$$

where the matrices and vectors are defined as in the polar formulation. The obvious attraction to this method is that the only matrix storage required is for $\underline{J}_{11}^{(\ell)}$ and $\underline{J}_{22}^{(\ell)}$ (not the full Jacobian $\underline{J}^{(\ell)}$). In addition, with the above formulation the real and reactive power calculations have been somewhat decoupled. However, it has been found [6] that the resulting method exhibits very poor convergence qualities (mainly due to the small degree of real and reactive power coupling still existing in $\underline{J}_{11}^{(\ell)}$ and $\underline{J}_{22}^{(\ell)}$).

Many other methods have been suggested to reduce storage and time, the most significant ones being by Stott [10], Peterson, Tinney and Bree [11], Despotovic, Babic and Mastilovic [12] and Despotovic [13]. Perhaps the simplest but yet most accurate method was proposed in June, 1974 by Stott and Alsac [14]. This Fast Decoupled Load Flow (FDLF) method begins with Equations (4-2.1) and (4-2.2). The entries in $J_{11}^{(\ell)}$ and $J_{22}^{(\ell)}$ can be transformed (by some algebra) from Equations (3-2.26), (3-2.27), (3-2.32) and (3-2.33) to read:

$$\left(J_{11}\right)_{kj}^{(\ell)} \triangleq \frac{\partial P_k^{(\ell)}}{\partial \delta_j^{(\ell)}} = |V_k^{(\ell)}||V_j^{(\ell)}|\left[G_{kj}\sin\left(\delta_k^{(\ell)}-\delta_j^{(\ell)}\right) - B_{kj}\cos\left(\delta_k^{(\ell)}-\delta_j^{(\ell)}\right)\right]$$

$$; \; j,k = 2,3,\ldots,n \text{ if } j \neq k \qquad (4\text{-}2.3)$$

$$= -Q_k^{(\ell)} - B_{kk}\left|V_k^{(\ell)}\right|^2 \; ; \; h = 2,3,\ldots,n \text{ if } j = k. \qquad (4\text{-}2.4)$$

and

$$\left(J_{22}\right)_{kj}^{(\ell)} \triangleq \frac{\partial Q_k^{(\ell)}|V_k^{(\ell)}|}{\partial |V_k^{(\ell)}|} = \left(J_{11}\right)_{kj}^{(\ell)} \; ; \; j,k = 2,3,\ldots,n \text{ if } j \neq k \qquad (4\text{-}2.5)$$

$$= Q_k^{(\ell)} - B_{kk}\left|V_k^{(\ell)}\right|^2 \; ; \; k = 2,3,\ldots,n \text{ if } j = k. \qquad (4\text{-}2.6)$$

In practical power systems, the following are almost always valid:

i) $\cos\left(\delta_k^{(\ell)} - \delta_j^{(\ell)}\right) \approx 1$,

ii) $G_{kj}\sin\left(\delta_k^{(\ell)} - \delta_j^{(\ell)}\right) << B_{kj}$, and

iii) $Q_k^{(\ell)} << B_{kk}|V_k^{(\ell)}|^2$.

With these three assumptions, Equations (4-2.1) and (4-2.2) take on the form:

$$\Delta \underline{P}^{(\ell)} = \underline{\tilde{V}}^{(\ell)} \, \underline{B}^{(1)} \, \underline{\tilde{V}}^{(\ell)} \, \Delta \underline{\delta}^{(\ell)} \qquad (4\text{-}2.7)$$

and

$$\Delta \underline{Q}^{(\ell)} = \underline{\tilde{V}}^{(\ell)} \, \underline{B}^{(2)} \, \underline{\tilde{V}}^{(\ell)} \, \Delta |\underline{\hat{V}}|^{(\ell)} \qquad (4\text{-}2.8)$$

where

$$\underline{\tilde{V}}^{(\ell)} \triangleq \text{diag}\,\{|V_2|, |V_3|, \ldots, |V_n|\}. \qquad (4\text{-}2.9)$$

At this stage, $\underline{B}^{(1)}$ and $\underline{B}^{(2)}$ are simply the last (n-1) rows and columns of $-\underline{B}$ (the matrix whose kj'th entry is $-B_{kj}$). The decoupling and final

stages are completed by:

i) omitting from $\underline{B}^{(1)}$ any network elements that affect reactive power flows, namely shunt reactances and off-nominal in-phase transformers,

ii) omitting from $\underline{B}^{(2)}$ any effects of phase shifters, and finally,

iii) transforming equations (4-2.7) and (4-2.8) into

$$\left(\tilde{\underline{V}}^{(\ell)}\right)^{-1} \Delta\underline{P}^{(\ell)} = \underline{B}^{(1)} \Delta\underline{\delta}^{(\ell)} \qquad (4\text{-}2.10)$$

and

$$\left(\tilde{\underline{V}}^{(\ell)}\right)^{-1} \Delta\underline{Q}^{(\ell)} = \underline{B}^{(2)} \Delta|\underline{V}|^{(\ell)}. \qquad (4\text{-}2.11)$$

This transformation has in effect assumed that all right hand side voltages are one per unit.

Equations (4-2.10) and (4-2.11), along with i) and ii) above, become the FDLF equations.

Voltage controlled busses are handled in exactly the same way as in the conventional polar formulation of Newton-Raphson, i.e. the rows and columns corresponding to voltage controlled busses are deleted from $\underline{B}^{(2)}$. If at any time the reactive power at a voltage controlled bus exceeds the constraints, the corresponding row and column are inserted in $\underline{B}^{(2)}$, and the bus is treated as a load bus.

There are many different ways of solving Equations (4-2.10) and (4-2.11). One method (and the best method in this case as will be explained later) is Triangular Factorization (LU decomposition). (For a complete theoretical development, see Conte and deBoor [15] or Wilkinson [16].)

Consider a general n'th order system of linear equations expressed

as:

$$\underline{A}\,\underline{x} = \underline{b}. \tag{4-2.12}$$

If $\underline{A}$ is non-singular, it may be factored by Gauss elimination into a product of a lower triangular matrix $\underline{L}$ and an upper triangular matrix $\underline{U}$, such that

$$\underline{A} = \underline{L}\,\underline{U}. \tag{4-2.13}$$

A solution for $\underline{x}$ given any $\underline{b}$ may be computed directly by solving in turn two triangular systems

$$\underline{L}\,\underline{x}' = \underline{b} \quad \text{(forward substitution)} \tag{4-2.14}$$

and

$$\underline{U}\,\underline{x} = \underline{x}' \quad \text{(backward substitution)}. \tag{4-2.15}$$

The $\underline{L}$ term contains the multipliers used in the Gauss elimination, and the $\underline{U}$ term contains the upper triangular coefficient matrix.

An examination of the number of multiply/division operations indicates a total of $\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$ operations needs be performed for the LU decomposition. (As a matter of interest, solving (4-2.12) requires $4/3\ n^3 + 2n^2 - n/3$ operations when $\underline{A}$ is inverted, and $n^3/3 + n^2 - n/3$ operations by regular Gauss elimination with back substitution.) The clear cut advantage of the LU decomposition is realized when $\underline{A}$ remains constant, but $\underline{b}$ is variant. Once the LU decomposition has been performed, the only calculations required to solve for $\underline{x}$ given a new $\underline{b}$ is the forward and back substitution process, thus giving the clear cut advantage over the regular Gauss elimination.

In solving Equations (4-2.10) and (4-2.11), one sees that both $\underline{B}^{(1)}$ and $\underline{B}^{(2)}$ are constant throughout the iterative procedure. Thus, Triangular Factorization should be used to solve these equations. (The matrix $\underline{B}^{(2)}$ must be retriangularized each time a voltage controlled bus exceeds a reactive power limit. The matrix $\underline{B}^{(1)}$ need be triangularized only once.)

The appeal of (4-2.10) and (4-2.11) is thus enhanced by being able to solve the equations very fast by using the constant triangularizations of $\underline{B}^{(1)}$ and $\underline{B}^{(2)}$. Once $\Delta\underline{\delta}^{(\ell)}$ and $\Delta|\underline{V}|^{(\ell)}$ have been obtained, the solution procedure follows that of the polar form of Newton-Raphson.

The FDLF method was programmed using the LU decomposition procedure. The programming conditions specified in Chapter 3, Section 2 were satisfied. Table III shows the results of comparing the Newton-Raphson algorithm with the Fast Decoupled Load Flow Algorithm. From this table, and the above discussion, it can be seen that

i) in all cases, the FDLF algorithm took less time for a solution than did the Newton-Raphson method; once the $\underline{B}^{(1)}$ and $\underline{B}^{(2)}$ matrices were triangularized, each iterate in the FDLF amounted to a simple recalculation of $\left(\tilde{\underline{V}}^{(\ell)}\right)^{-1}\Delta\ \underline{P}^{(\ell)}$ and $\left(\tilde{\underline{V}}^{(\ell)}\right)^{-1}\Delta\ \underline{Q}^{(\ell)}$ with appropriate forward and back substitution.

ii) the required storage for all but the 14-bus system was considerably less in the FDLF method than with the Newton-Raphson algorithm. The 14-bus system anomaly can be explained by noting that due to the factorization

TABLE III

Comparison of Newton-Raphson and Fast Decoupled Load Flow

	Newton-Raphson			FDLF		
System	Time	Storage	Iterations	Time	Storage	Iterations
14 bus	1.69	54	4	1.04	56	15
30 bus	10.71	70	5	2.79	68	13
57 bus	55.86	128	4	9.11*	108	13*
118 bus	560.8	384	5	35.57	246	7

Time in seconds

Storage in kilobytes

*Voltage tolerance of 0.0062

and substitution routines, the object code for the FDLF method was slightly larger than for the Newton-Raphson algorithm.

iii) with a voltage tolerance of 0.0005, the 57-bus system oscillated around the exact solution, then after 80 to 85 iterations, the algorithm diverged. With a voltage tolerance of 0.0062, the solution was reached for all but two busses (busses 32 and 33), these busses being off in the second decimal place in both magnitude and angle. This phenomenon can be explained by the following: In the Newton-Raphson method, the Jacobian $\underline{J}$ is evaluated each iterate, thus being a "variable tangent" method. It is well known that if a variable tangent method is going to converge, it converges quadratically (the error at the n'th iterate is on the order of the square of the error at the (n-1) 'th iterate). The FDLF method is Newton-like only in that Equations (4-2.10) and (4-2.11) are "fixed tangent" equations. This fixed tangent method has geometric convergence, i.e. roughly speaking, the error of the method is proportional to the Euclidean distance between the Jacobian $\underline{J}$ and the matrix

$$\underline{\hat{B}} \triangleq \begin{bmatrix} \underline{B}^{(1)} & \underline{0} \\ \underline{0} & \underline{B}^{(2)} \end{bmatrix},$$

where the Jacobian $\underline{J}$ is evaluated at the solution. In

other words, if the decoupling matrix $\hat{\underline{B}}$ (which can be thought of as an approximate tangent matrix evaluated at a flat start) is "near" enough to the final Jacobian, then the method will converge to the proper solution. In all but the 57-bus system, $\hat{\underline{B}}$ was "close" enough to $\underline{J}$ to insure convergence. This was not the case for the 57-bus system, thus forcing oscillations near the exact solution. (Stott and Alsac solved the 57-bus system but did not report this phenomenon. The reason is that the authors used a less strict convergence criterion than the 0.0005 voltage tolerance.) It should be noted that the solution obtained for the 57-bus system was very near the exact solution. (Note: for an analog one dimensional theoretical discussion of the above, see [24]).

iv) it was reported by Stott and Alsac [14] that the FDLF method solved systems unsoluable by the Newton-Raphson method. Heuristically, this can be explained as follows: the "fixed tangent" method (FDLF) is insensitive to "humps" in the defining function (the left hand side of Equations (4-2.10) and (4-2.11)), whereas the "variable tangent" method (Newton-Raphson) may, due to these humps, misdirect the solution process away from the exact solution, resulting in divergence.

In conclusion, in comparing the two methods, the FDLF algorithm is capable of solving systems faster and with less required storage than the Newton-Raphson algorithm. In some systems, the FDLF method will not converge upon the exact solution point, but will instead oscillate near

the point. In other systems, because of the insensitivity to variations in the defining function, the FDLF algorithm will converge upon a solution that the Newton-Raphson algorithm will not.

CHAPTER V

SPARSITY CONSIDERATIONS

Consider a general n'th order system of linear equations

$$\underline{A}\,\underline{x} = \underline{b} \; . \qquad (5.1)$$

If $\underline{A}$ is a very sparse matrix (very few non-zero entries), then when solving for $\underline{x}$, it is possible to gain considerable savings in both the required storage and time by special programming techniques (sparsity techniques). Much research has been done trying to develop these special programming methods and apply them to many areas analyzed with sparse matrices.

Research in the sparsity area began around 1957 with a paper by H. M. Markowitz [17] who applied special programming techniques to the linear programming problem. In the power system field, sparsity was not considered until around 1963 when Carpentiér [20] and Sato and Tinney [21] published the results of their research in the sparsity area. Since this original work, many tens of publications have appeared applying sparsity techniques specifically to the power flow problem.

When considering the power flow problem, one sees that the network equations are very sparse due to the very few physical interconnections in a typical power system. As an example, the bus admittance matrix $\underline{Y}$ for the 118-bus system has 13,924 complex entries, of which only 476 are non-zero. Even more revealing, the rectangular Jacobian $\underline{J}$ for the 118-bus system has 54,756 real entries, of which only 1,409 are non-zero. (The matrices $\underline{B}^{(1)}$ and $\underline{B}^{(2)}$ in the Fast Decoupled Load Flow

method are likewise very sparse.) Due to this inherent sparsity, considerable savings can be realized in the power flow problem by utilizing sparsity techniques.

Very briefly, sparsity techniques can be divided into four areas, these being:

i) non-zero storing; Some scheme must be developed to store and retrieve only the non-zero entries of the system matrix. One very good method is a linked list structure (see Ogbuobiri [23]).

ii) ordering; Ordering may be viewed as a renumbering of the system equations (renumbering of the busses in a system) such that when the factorization is performed in the renumbered order, the sparsity is preserved. (In the Triangular Factorization, the $\underline{L}$ and $\underline{U}$ factors remain sparse.) In addition to preserving the sparsity of the equations, the numerical accuracy must be preserved by pivoting considerations. (However, with good fortune, power system equations are well behaved, resulting in the system matrices being strongly diagonally dominant. Thus, numerical considerations are not a part of ordering schemes in the power flow problem.) Ordering schemes for sparse sets of equations have been covered in great detail in Ogbuobiri, Tinney and Walker [18], and in less detail (but more understandability) in Tinney and Walker [22] and Tinney and Meyer [19].

iii) factorization; The LU decomposition is calculated by processing and manipulating only the non-zero elements of the system matrix.

iv) direct solution; The forward and back substitution is done using only the non-zero elements of $\underline{L}$ and $\underline{U}$.

Additional savings in storage and time can be realized when the matrix being considered is either strictly symmetric (as is the case for $\underline{B}^{(2)}$, or $\underline{B}^{(1)}$ and $\underline{Y}$ if no phase shifters are present) or asymmetric in element value but with symmetric sparsity structure ($\underline{J}$ in both forms of the Newton-Raphson).

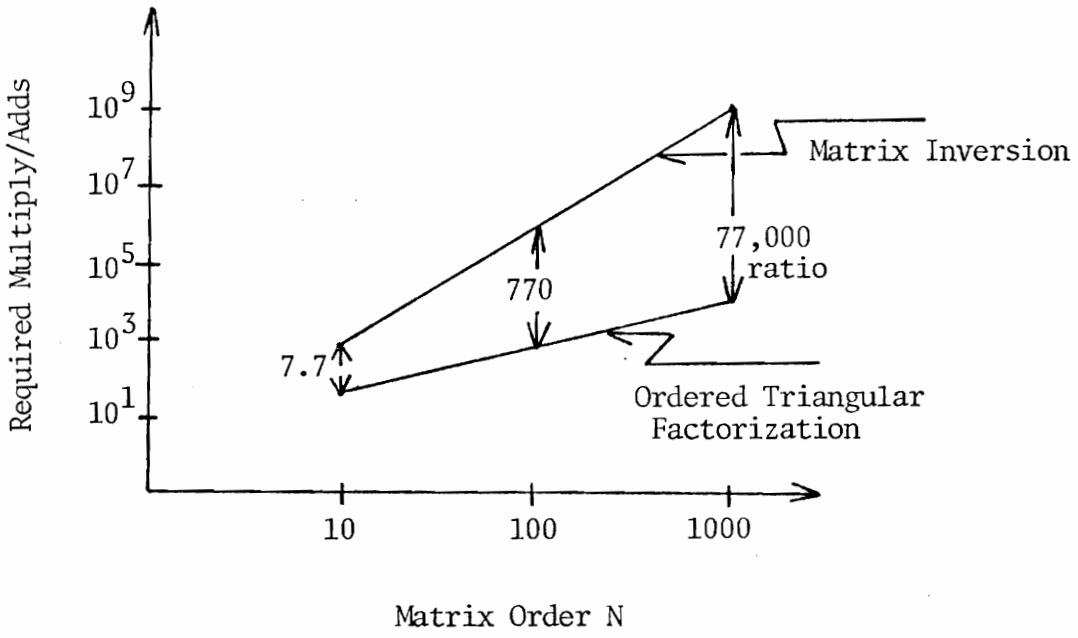
In general, sparse matrix methods' time and storage requirements increase linearly with problem size. Thus, the Ordered Triangular Factorization (with non-zero element storage) realizes its greatest gain with large problems. As an example, consider Figure I (from Tinney and Meyer [19]) which shows the relative ratio in effort of solving (5.1) with matrix inversion versus Ordered Triangular Factorization (OTF) for typical power system problems.

Sparsity programming has two major difficulties, these being:

i) The programming logic is very complicated, resulting in many man-hours of program debugging and coding. Sparsity coding requires more than a casual knowledge of the basic programming language being used (FORTRAN, BASIC, PL-1, etc.). In effect, sparsity techniques are nothing less than a substitution of a higher level compiler for the basic programming language compiler being used.

Figure I

Comparison of Sparsity/Non-Sparsity Programming



ii) The sparsity routines result in an increase in the object code storage requirements of the power flow program. For small networks, this increase in storage requirements may out weigh the gains achieved from non-zero element storage. In addition, for small networks, the added logic execution time may outweigh the gains of non-zero element processing and manipulation. Thus, there is a break even point after which sparsity techniques should be included, but before which conventional programming techniques are better.

Sparsity routines were written in FORTRAN using the storage and retrieval algorithm of Ogbuobiri [23] and the sparsity programmed LU decomposition of Tinney and Walker [22]. Unfortunately, time did not permit those routines to be applied to the complete solution of the load flow problem, however, for the Newton-Raphson method, the break even point mentioned in ii) above appeared to be around the 30-bus system. In other words, for any network smaller than the 30-bus system, general matrix storage and conventional retrieval methods should be used. For systems larger than 30-busses, time and storage could be saved by using sparsity techniques. In the Gauss-Seidel algorithm, the break even point appeared to be between the 30 bus and the 57-bus system. As mentioned above, the larger the problem being considered, the more likely the sparsity techniques will indeed be an advantage. Thus, due to the Jacobian, one would expect the Newton-Raphson break even point to be at a smaller system than the Gauss-Seidel algorithm.

Both the Modified Nodal Iterative Technique and the Fast Decoupled Load Flow method should have break even points similar to their parent algorithms, i.e. the MNIT break even point should be around the 30-bus system, and the FDLF break even point between the 30-bus and 57-bus system.

It must be emphasized that the above suggested results are from a prefactory study. Complete programming of the different power flow algorithms must be done before hard and fast conclusions should be drawn. It is hoped that the above results may be used as a suggested starting point for further research.

CHAPTER VI

CONCLUSIONS

If there is any one theme in the first five chapters, hopefully it is that no one algorithm possesses all of the desirable features of the others. Each method considered has its desirable characteristics, and its undesirable limitations. In choosing a power flow algorithm, these characteristics and limitations must be weighed with the type and size of the system, computer size, experience of the programming engineer, and perhaps most importantly to a power utility, the amount of time and money willing to be invested into writing a power flow program.

Further insight into choosing an algorithm may be gained by briefly summarizing the results of the first five chapters (all empirical results are in Table IV). The summary can be divided into two sections: (1) With sparsity not being considered, the results would indicate that the Gauss-Seidel algorithm is the most economical method, especially in limited core applications. If the amount of core area is not extremely restricted, the Fast Decoupled Load Flow method of Stott and Alsac is very attractive, especially when considering the solution times given in Table IV. In systems with series capacitive lines and few voltage controlled busses, the Modified Nodal Iterative Technique of Podmore and Undrill perhaps should be used. (2) (It must be emphasized that the conclusions drawn in this section are from an incomplete study of the effects of sparsity techniques on the results in Table IV.) With sparsity being considered, the most attractive method is the Fast Decoupled Load Flow algorithm. If the symmetry of the FDLF equations is

TABLE IV

Empirical Comparison of Power Flow Algorithms

	Gauss-Seidel		MNIT		Newton-Raphson		FDLF	
System	Time	Storage	Time	Storage	Time	Storage	Time	Storage
14 Bus	1.07	48	1.24	50	1.69	54	1.04	56
30 Bus	3.11	54	5.16	56	10.71	70	2.79	68
57 Bus	10.64	76	30.31	78	55.86	128	9.11*	108
118 Bus	107.37	166	310.34	168	560.8	384	35.57	246

Time in seconds

Storage in kilobytes

*Voltage tolerance of 0.0062

exploited, the core requirements for this method are similar to the minimal requirements of the Gauss-Seidel method. Additionally, the FDLF algorithm exhibits the fastest solution time of any of the methods studied. If one wants to use a more conventional method (other than the relatively new and untested FDLF), then with sparsity, the required storage and solution time of the Newton-Raphson method is tolerable. In addition, the Newton-Raphson method is capable of solving certain systems not solvable by the Gauss-Seidel algorithm. (This fact, more than any other, is responsible for the Newton-Raphson method being accepted as the universal power flow algorithm.)

Future work would obviously begin by completing the sparsity routines for each algorithm. Additionally, the FORTRAN coding should be optimized, or perhaps a different programming language should be tried (PL-1 has very good possibilities when using sparsity programming.) Another area of future work would be in testing to see which algorithms are particularly suited for bulk data transfer from core to some bulk memory device. One would expect that due to the real and reactive power decoupling, the Fast Decoupled Load Flow algorithm would be particularly applicable. More test cases (including larger systems) along with a larger number of algorithms should be considered. Finally, work should be done trying to find an analytical method for determining the optimum accelerating factor for the Gauss-Seidel algorithm. Perhaps a linear search using the power mismatch equations at each bus could be used to determine the optimum accelerating factor at each iteration.

BIBLIOGRAPHY

[1] J. B. Ward, H. W. Hale, "Digital Computer Solution of Power-Flow Problems," AIEE Trans. on P.A.S., Vol. PAS-75, pp. 398-404, June, 1956.

[2] J. E. VanNess, J. H. Griffin, "Elimination Methods for Load Flow Studies," AIEE Trans. on P.A.S., Vol. PAS-80, pp. 299-304, June, 1961.

[3] W. F. Tinney, C. E. Hart, "Power Flow Solution by Newton's Method," IEEE Trans. on P.A.S., Vol. PAS-86, pp. 1449-1460, November, 1967.

[4] A. M. Sasson, F. J. Jaimes," Digital Methods Applied to Power Flow Studies," IEEE Trans. on P.A.S., Vol. PAS-86, pp. 860-867, July, 1967.

[5] O. I. Elgerd, Electric Energy Systems Theory: An Introduction, McGraw-Hill, New York, 1971.

[6] G. W. Stagg, A. H. El-Abied, Computer Methods in Power System Analysis, McGraw-Hill, New York, 1968.

[7] J. E. VanNess, "Iterative Methods for Digital Load Flow Studies," AIEE Trans. on P.A.S., Vol. PAS-78, pp. 583-588, August, 1959.

[8] J. A. Treece, "Bootstrap Gauss-Seidel Load Flow," Proc. IEEE, Vol. 116, pp. 866-870, May, 1969.

[9] R. Podmore, J. M. Undrill, "Modified Nodal Iterative Load Flow Algorithm to Handle Series Capacitive Branches," IEEE Trans. on P.A.S., Vol. PAS-92, pp. 1379-1387, July/August, 1973.

[10] B. Stott, "Decoupled Newton Load Flow," IEEE Trans. on P.A.S., Vol. PAS-91, pp. 1955-1959, September/October, 1972.

[11] N. M. Peterson, W. F. Tinney, D. W. Bree, "Iterative Linear AC Power Flow Solution for Fast Approximate Outage Studies," IEEE Trans. on P.A.S., Vol. PAS-91, pp. 2048-2056, September/October, 1972.

[12] S. T. Despotovic, B. S. Babic, V. P. Mastilovic, "A Rapid and Reliable Method for Solving Load Flow Problems," IEEE Trans. on P.A.S., Vol. PAS-90, pp. 123-130, January/February, 1971.

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

[14] B. Stott, O. Alsac, "Fast Decoupled Load Flow," IEEE Trans. on P.A.S., Vol. PAS-93, pp. 859-837, May/June, 1974.

[15] S. D. Conte, C. deBoor, Elementary Numerical Analysis, Second Edition, McGraw-Hill, New York, 1972.

[16] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, Belfast, 1965.

[17] H. M. Markowitz, "The Elimination Form of the Inverse and its Application to Linear Programming," Management Science, Vol. 3, pp. 281-285, 1957.

[18] E. C. Ogbuobiri, W. F. Tinney, J. W. Walker, "Sparsity-Directed Decomposition for Gaussian Elimination on Matrices," IEEE Trans. on P.A.S., Vol. PAS-89, pp. 141-150, January, 1970.

[19] W. F. Tinney, W. S. Meyer, "Solution of Large Sparse Systems by Ordered Triangular Factorization," IEEE Trans. on Automatic Control, Vol. AC-18, pp. 333-346, August, 1973.

[20] J. Carpentier, "Ordered Eliminations," Proc. PSCC, London, 1963.

[21] N. Sato, W. F. Tinney, "Techniques for Exploiting the Sparsity of the Network Admittance Matrix," IEEE Trans. on P.A.S., Vol. PAS-82, pp. 944-950, December, 1963.

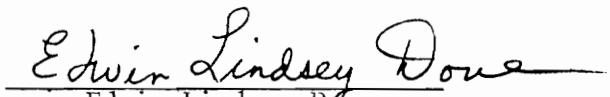
[22] W. F. Tinney, J. W. Walker, "Direct Solution of Sparse Network Equations by Optimally Ordered Triangular Factorization," Proc. IEEE, Vol. 55, pp. 1801-1809, November, 1967.

[23] E. C. Ogbuobiri, "Dynamic Storage and Retrieval in Sparsity Programming," IEEE Trans. on P.A.S., Vol. PAS-89, pp. 150-155, January, 1970.

[24] C. E. Froberg, Introduction to Numerical Analysis, Second Edition, Addison-Wesley, Reading, Mass., pp. 21-26, 1969.

VITA

Edwin Lindsey Dove was born March 5, 1951 in Linville, Virginia, where he received his primary and secondary education. In September 1969, he entered VPI&SU and was graduated in June 1973, with a B.S. degree in Electrical Engineering.



Edwin Lindsey Dove

A COMPARISON OF DIGITAL METHODS APPLIED TO POWER FLOW STUDIES

by

Edwin Lindsey Dove

(ABSTRACT)

Four power flow algorithms were developed, and qualitative and quantitative comparisons were given in terms of solution time required, computer storage necessary, convergence characteristics, and ease of programming and program modification. Sparsity techniques were outlined, and a suggestion of the effects of these techniques on the four algorithms was included. An outline was given of criterion that must be considered before selecting a power flow algorithm.