THE UNIVERSITY OF MANITOBA

OPTIMIZATION METHODS IN POWER SYSTEM LOAD FLOW STUDIES

BY

GARBO JENG

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

WINNIPEG, MANITOBA

MARCH 1972

ABSTRACT

This thesis presents solution techniques for power system load flow problems using digital computers.

Based on the fact that load flow solution corresponds to the point at which power mismatches at all nodes of the system are minimum, a reformulation of the problem as an optimization problem is introduced. The least p-th formulation of the objective function is adopted. An efficient minimization algorithm, the variable metric method (VMM) without linear search, is shown to give better convergence than the algorithm with linear search. The former requires a total number of function evaluations about 60% of that required by the latter.

It is concluded that this formulation together with the use of an efficient minimization algorithm, enables solution to be obtained for ill-conditioned systems for which the commonly used iterative method fails. A 13-bus system with series capacitors, which diverges with the successive overrelaxation (SOR) method, is solved by VMM. Convergence was obtained with no difficulty.

ACKNOWLEDGEMENTS

The author wishes to express her deepest gratitude to Dr. A. Wexler for his untiring supervision and encouragement.

The author is also indebted to Dr. G. W. Swift for his invaluable advice and assistance.

Thanks are also extended to Mrs. D. Lowry for her excellent typing.

Financial support from the National Research Council of Canada is greatly appreciated.

CONTENTS

1

introduction

The power system load flow analysis consists mainly of determining node voltages, both magnitudes and phases, and consequently the branch currents for a given schedule of generations and loads at the nodes of the network under study [29]. Mathematically, the problem involves the solution of a set of nonlinear equations which describes the network and satisfies the nodal restraints governing the power and voltage requirements of the load and generation points.

Two commonly used digital methods for solving the problem are the Gauss-Seidel method [14], also known as the method of successive displacements, and the Newton-Raphson method [35]. The Gauss-Seidel method expresses explicitly each voltage variable in terms of all other variables and calculates, at each iteration, the new voltage value from the previous values. The method converges slowly but the storage requirement is minimal. Since an acceleration factor is used to hasten

the convergence of the process, it is referred to as successive over-relaxation (SOR) method subsequently. The Newton-Raphson (NR) method of load flow solves the set of nonlinear equations by first deriving a set of linear relationships relating small changes in voltages to small changes in power and then solving this set of linear equations for the voltage corrections. The method converges with few iterations but requires relatively large storage.

Since load flow solution corresponds to the set of node voltages at which power mismatches at all nodes of the system are minimum, the problem is reformulated as an optimization problem [33] by defining an objective function which measures the total power mismatches. This formulation of the problem results in a solution process in which successive voltage corrections are found with the corresponding monotonic decrease in power mismatch. The solution process is terminated when the power mismatch is acceptable. Hence, it is more physically meaningful than methods based on voltage convergence criteria. Further, the technique is capable of solving problems which diverge with the commonly used iterative methods. Another promising feature is that, with the optimization approach, the nonexistence of a solution to a particular problem considered is clearly indicated from the nonzero minimum power mismatch obtained at the end of the minimization process. Whereas with the normal iterative methods, when a diverging solution is obtained, it is not clear whether divergence has been due to instability in the method or to the fact that there may not be a solution at all due, for example, to system instability.

As explicit expression for the gradient vector of the load flow function

so defined is available, gradient methods for unconstrained function minimization can be used. In Chapter 2 two basic gradient methods, the steepest descent method and the generalized Newton-Raphson method, are included as a preliminary study. A second order method called the variable metric method (VMM), which takes into account the curvature of the function being minimized but does not require the matrix of second partial derivatives to be calculated, is described. Some important properties of the method are derived and proved. Chapter 3 formulates the load flow problem. Numerical examples and results are included. A]3-bus system with negative transfer reactance is solved demonstrating the instability of the normal iterative method. Finally, Chapter 4 summarizes the conclusions.

2

optimization methods

In this chapter, the general problem of finding an unconstrained local minimum[1] of a function $f(\underline{x})$ of n variables $\underline{x} = [x_1, x_2, \ldots, x_n]^T$ is considered.

The methods available for solving this problem are generally classified as 'direct' methods and 'gradient' methods [2]. A direct method [9,17,20,21,25] for function minimization is one which does not require the derivatives of the function to be computed. Only function values are needed in the course of the minimization process. Gradient methods [8,11,12,36] are those which require the derivatives of the function to be computed as well as the function itself. The latter, taking into account an additional derivative information of the function, has generally better convergence rate than the direct methods. An exhaustive review on optimization methods has been given by Bandler [2]. A list of references is available in that paper.

---

1 The problem of locating the maximum of a function can be regarded as that of locating the minimum of the negative of the function.

In the first two sections of this chapter, commonly used terminologies are defined and fundamental concepts are introduced. Two basic gradient methods, the steepest descent method [36] and the generalized Newton-Raphson method [23] are described. This is followed by a detailed description of a class of methods called the variable-metric method (VMM) [1,6,7,8,10,11] which combines the characteristics of the two methods. Two versions of the method are studied. One, due to Davidon [8] and reformulated by Fletcher and Powell [11], is commonly known as the Fletcher and Powell algorithm. The other is a modification [6,7,10] of the first algorithm by abandoning the one-dimensional minimization process at each iteration. The former is referred to as Algorithm 1 and the latter Algorithm 2, hereforth. Both algorithms utilize partial derivative information to determine the direction of search and are directly applicable to the solution of load flow problems.

A recurrence formula for the successive search directions of Algorithm 1 has been established in this thesis. It is also proved that the deflection matrix relating two successive search directions is positive semidefinite. From this, it is concluded here that successive search directions generated by Algorithm 1 contain an angle less than, or at most equal to, $\frac{\pi}{2}$.

2.1 Definitions and Notations

The following definitions and notations are used throughout the chapter.

(1) Superscript T denotes the transpose of a matrix.

(2) $\underline{x}$ denotes the position vector with components $x_1, x_2, \ldots, x_n$.

(3) $\underline{\delta}$ denotes the displacement vector.

(4) $F(\underline{x})$ denotes a scalar function of $\underline{x}$ with continuous first and second derivatives.

(5) $\underline{g}(\underline{x})$ or $\nabla F(\underline{x})$ is the gradient vector of $F(\underline{x})$. The elements of $\underline{g}(\underline{x})$ are the first partial derivatives of F.

i.e., $\underline{g}(x) = \nabla F(\underline{x}) = [\frac{\partial F}{\partial x_1} \quad \frac{\partial F}{\partial x_2} \quad \cdots \quad \frac{\partial F}{\partial x_n}]^T$.

(6) $G(\underline{x})$ is an $n \times n$ symmetric matrix, called the Hessian matrix, the elements of which are the second partial derivatives of F with respect to $x_1, x_2, \ldots, x_n$.

$$G(x) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & & & \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}$$

(7) $H(\underline{x})$ is the inverse Hessian matrix, i.e., $H = G^{-1}$.

(8) Subscripts associated with vectors denote iteration number.

(9) $\underline{y}_i$ denotes the difference of gradients at the $(i + 1)$th iteration and ith iteration, i.e., $\underline{y}_i = \underline{g}_{i+1} - \underline{g}_i$

(10) $||\underline{g}||$ denotes the norm of vector $\underline{g}$ and is defined as $||\underline{g}||^2 = \underline{g}^T\underline{g}$

2.2 Preliminary Study

Consider first the one-dimensional problem of finding the minimum of a function, F, of one variable x as in Fig. 2.1(a). The necessary

and sufficient conditions [16] for a function to have a minimum are that the first derivative vanishes and the second derivative is positive.

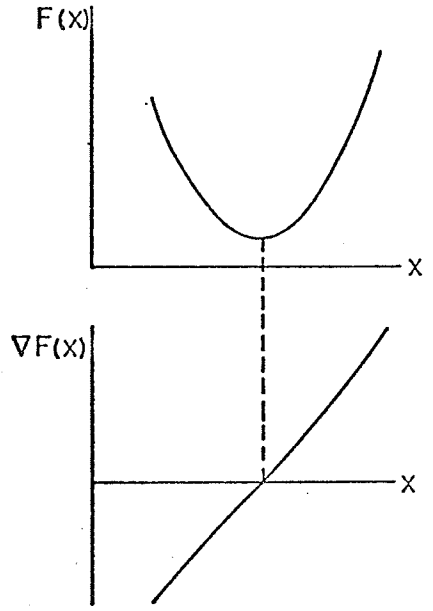
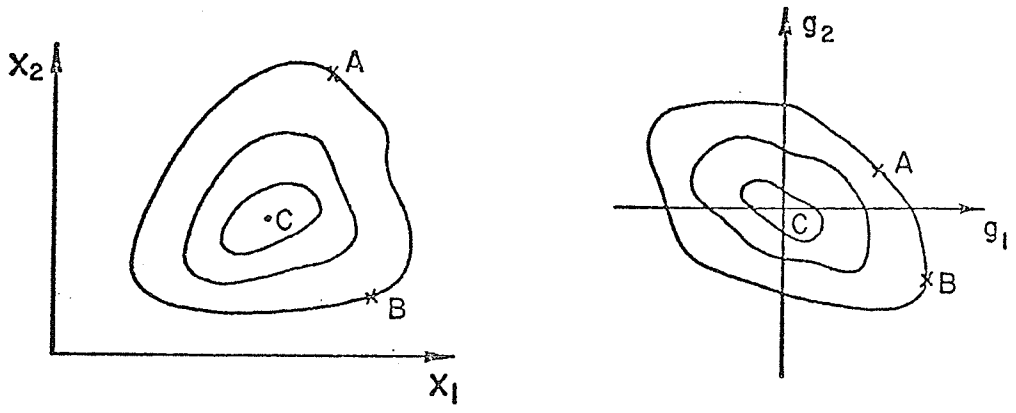


Fig. 2.1(a) a convex function F of one variable (top)
(b) gradient of F (bottom)

Figs. 2.1(a) and 2.1(b) show a convex function and its first derivative, respectively. A convex function is one which can never be underestimated by a linear interpolation between any two points on the curve. It is obvious from the graphs that the problem of locating the minimum of a function is the same as that of finding the zero of its first derivative.

Now, consider the two-dimensional case. Fig. 2.2(a) shows the contours of a function, F, with two independent variables x_1, x_2. The



(a) position space (b) gradient space

Fig. 2.2 (a) Two-dimensional contour sketch
(b) Locus of gradient vector of contour

corresponding locus of the gradient vector of the contour are sketched in the gradient space of g_1 and g_2 as shown in Fig. 2.2(b). For this case, as is implied by the necessary and sufficient conditions stated earlier, the origin of the gradient space corresponds to the minimum point of the function $F(x_1, x_2)$ if the 2×2 matrix of second partial derivatives, the Hessian matrix, is positive definite. By positive definiteness of a matrix G, we mean that $\underline{x}^T G \underline{x} > 0$ for all nontrivial $\underline{x}$ and $\underline{x}^T G \underline{x} = 0$ iff $\underline{x} = 0$.

Generalized to a space of n dimensions, i.e., a function with n independent variables, the conditions required are that the gradient vector is zero and the $n \times n$ Hessian matrix is positive definite.

The gradient of a function at any point thus gives a good indication of the direction in which to proceed during the search process.

Steepest Descent Method

This method is one of the oldest and simplest of gradient methods. Compared to the generalized Newton-Raphson method, it has the property of being stable and requires only the first partial derivatives of the function. But convergence is very slow. The mathematical derivation of the method and its properties are given below.

A function $F(x_1', x_2')$ of two variables x_1' and x_2' at $(x_1 + \delta_1, x_2 + \delta_2)$ has a Taylor series expansion about x_1 and x_2 of the form

$$F(x_1 + \delta_1, x_2 + \delta_2) = F(x_1, x_2) + \frac{\partial F}{\partial x_1'} \delta_1 + \frac{\partial F}{\partial x_2'} \delta_2 +$$

$$+ \tfrac{1}{2} \delta_1^2 \frac{\partial^2 F}{\partial x_1'^2} + \tfrac{1}{2} \delta_2^2 \frac{\partial^2 F}{\partial x_2'^2} + \delta_1 \delta_2 \frac{\partial^2 F}{\partial x_1' \partial x_2'} + \ldots \qquad (2.1)$$

In matrix form,

$$F(\underline{x} + \underline{\delta}) = F(\underline{x}) + \underline{g}^T(\underline{x})\underline{\delta} + \frac{1}{2}\underline{\delta}^T G(\underline{x})\underline{\delta} + \ldots \tag{2.2}$$

where

$$\underline{x} = [x_1, x_2]^T \tag{2.3a}$$

$$\underline{\delta} = [\delta_1, \delta_2]^T \tag{2.3b}$$

$$\underline{g} = \left[\frac{\partial F}{\partial x_1'} \quad \frac{\partial F}{\partial x_2'}\right]^T \tag{2.3c}$$

and

$$G = \begin{bmatrix} \dfrac{\partial^2 F}{\partial x_1'^2} & \dfrac{\partial^2 F}{\partial x_1' \partial x_2'} \\ \dfrac{\partial^2 F}{\partial x_2' \partial x_1'} & \dfrac{\partial^2 F}{\partial x_2'^2} \end{bmatrix} \tag{2.3d}$$

Generalized to a function $F(\underline{x})$ with n variables $\underline{x} = [x_1, \ldots, x_n]^T$, the Taylor series expansion, in matrix form, of $F(\underline{x})$ has an expression similar to that given by eqn. (2.2) with $\underline{x}$, $\underline{\delta}$ and $\underline{g}$ each being a n-th order vector and G, a n x n matrix of second partial derivatives. Now consider the problem of minimizing an objective function $F(\underline{x})$ of n variables $\underline{x}$.

From eqn. (2.2), to first order terms, the variation, ΔF, in the objective function $F(\underline{x})$ is given by

$$\Delta F = F(\underline{x} + \underline{\delta}) - F(\underline{x})$$

$$= \underline{g}^T(\underline{x})\underline{\delta} \tag{2.4}$$

Eqn. (2.4) indicates that the maximum change, ΔF, in function value occurs when $\underline{\delta}$ is in the direction of the gradient vector $\underline{g}$. The steepest descent direction, $\underline{P}$, is thus given by

$$\underline{P} = - \nabla F = - \underline{g} \tag{2.5}$$

The iterative process can thus be stated as

$$\underline{x}_{i+1} = \underline{x}_i + \underline{\delta}_i \tag{2.6}$$

$$= \underline{x}_i + \alpha_i \underline{P}_i = \underline{x}_i - \alpha_i \underline{g}_i \tag{2.7}$$

where α_i is a positive scalar which is introduced to control the step of movement along the direction $\underline{P}_i$. Its value is determined such that the maximum decrease in function value along $\underline{P}_i$ is obtained. In other words, α_i is the value of α which minimizes $F(\underline{x}_i + \alpha \underline{P}_i)$ along $\underline{P}_i$. i.e.,

$$\frac{d}{d\alpha} F(\underline{x}_i - \alpha \underline{g}_i)\Big|_{\alpha=\alpha_i} = 0 \tag{2.8}$$

Therefore,

$$-\underline{g}_{i+1}^T \underline{g}_i = 0 \tag{2.9}$$

In words, eqn. (2.9) implies that the gradient at the $(i + 1)$th iteration $\underline{g}_{i+1}$, is orthogonal to the gradient at the ith iteration $\underline{g}_i$. Further, $\underline{g}_{i+1}$ is also orthogonal to $\underline{P}_i$ in view of eqn. (2.5).

We now show that the function value decreases at each step, i.e., the process is stable.

From eqns. (2.4) to (2.7) inclusive,

$$\Delta F_i = \underline{g}_i^T \underline{\delta}_i$$

$$= - \alpha_i \underline{g}_i^T \underline{g}_i = - \alpha_i ||\underline{g}_i||^2 \qquad (2.10)$$

Eqn. (2.10) shows that the first-order variation is negative for $\alpha > 0$ so that $F(x_{i+1}) < F(x_i)$ for α sufficiently small.

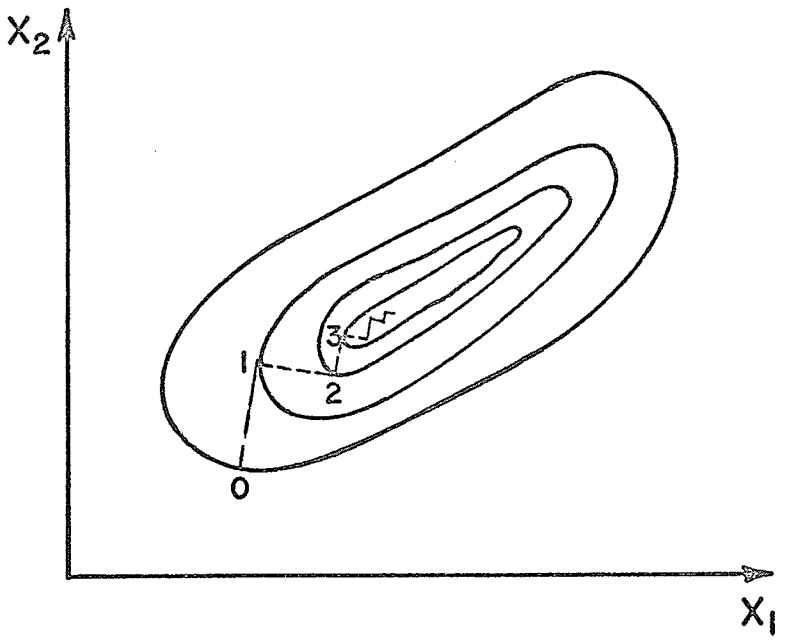


Fig. 2.3 Minimization by a Steepest Descent Method

Fig. 2.3 shows the progress of the method towards the minimum. As shown, the rate of convergence is very slow near the minimum of a narrow valley.

Generalized Newton-Raphson Method

In this method, the search direction is not solely determined by the first order gradient information as in the case of steepest descent. It

also utilizes the second derivatives information to determine the direction of search. Consider the Taylor series expansion of eqn. (2.2)

$$F(\underline{x} + \underline{\delta}) = F(\underline{x}) + \underline{g}^T(\underline{x})\underline{\delta} + \tfrac{1}{2}\,\underline{\delta}^T\, G(\underline{x})\,\underline{\delta} + \ldots$$

Differentiating and neglecting higher order terms give

$$\begin{aligned} \nabla F(\underline{x} + \underline{\delta}) &= \nabla F(\underline{x}) + \nabla \underline{g}^T(\underline{x})\underline{\delta} \\ &= \nabla F(\underline{x}) + G(\underline{x})\underline{\delta} \end{aligned} \tag{2.11}$$

The necessary condition at a minimum of a function is that the gradient vanishes there. Hence, if it is desired that $\underline{x}_{i+1} = \underline{x}_i + \underline{\delta}_i$ is the minimum of $F(\underline{x})$, we must have

$$\nabla F(\underline{x}_i + \underline{\delta}_i) = 0 \tag{2.12}$$

Using eqn. (2.12), eqn. (2.11) gives

$$\begin{aligned} \underline{\delta}_i &= -\, G^{-1}(\underline{x}_i)\, \nabla F(\underline{x}_i) \\ &= -\, H(\underline{x}_i)\, \underline{g}(\underline{x}_i) = -\, H_i\, \underline{g}_i \end{aligned} \tag{2.13}$$

where $H_i = G^{-1}(\underline{x}_i)$.

For a quadratic function, H is a constant matrix, eqn. (2.13) provides the parameter increments for the minimum to be reached in exactly one step. If the function is not quadratic, eqn. (2.13) provides the basis of an iterative scheme

$$\underline{x}_{i+1} = \underline{x}_i - \alpha_i\, H_i\, \underline{g}_i \tag{2.14}$$

where the positive scalar α_i is again included as in the case of steepest descent.

It is noted from eqn. (2.14) that the search direction will be in the gradient direction if H_i happened to be a multiple of the unit matrix I. The steepest descent method is a special case of the generalized Newton-Raphson method with $H = I$.

Eqn. (2.14), by taking into account the curvature of the function, suggests that one proceeds in a direction which is not necessarily along the gradient. Though the greatest rate of decrease of function value at any point occurs in the direction of negative gradient, this does not necessarily imply that the overall convergence of the process is fast. This is explained in Fig. 2.4 where dotted lines indicate the process using eqn. (2.14) and solid straight lines refer to the steepest descent case.

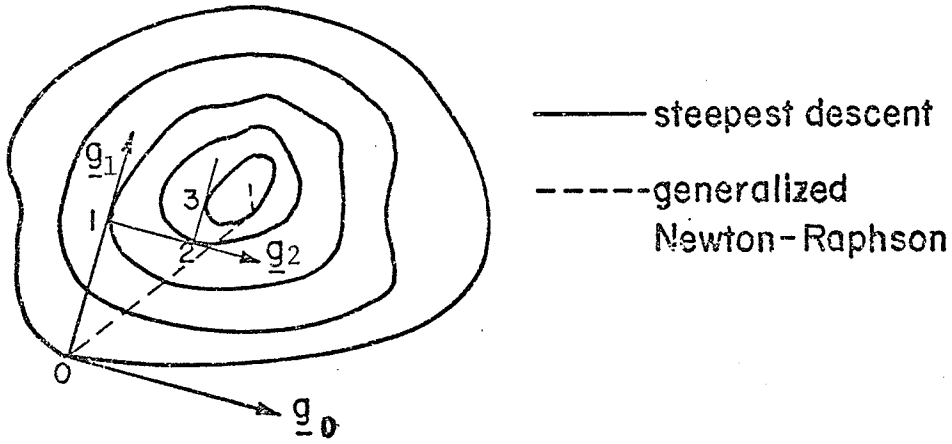


Fig. 2.4 Steepest Descent and Generalized Newton-Raphson Processes

As shown in Fig. 2.4 the minimization process in which the corrections are taken along the direction given by eqn. (2.14) takes fewer steps to reach the minimum point.

The stability condition of this process will now be established.

Eqn. (2.4) gives the variation, ΔF, of the function value

$$\Delta F_i = \underline{g}^T(x_i)\underline{\delta}_i$$

From eqn. (2.13), it follows that

$$\Delta F_i = - \underline{g}_i^T H_i \underline{g}_i \tag{2.15}$$

It is noted that the right hand side of eqn. (2.15) is quadratic in form. Therefore,

$$F(\underline{x}_{i+1}) < F(\underline{x}_i) \tag{2.16}$$

if H_i is positive definite.

As is obvious from eqns. (2.15) and (2.16), a positive definite H guarantees monotonic convergence of the process. The convergence of the method thus depends on the behaviour of the function.

The generalized Newton-Raphson method, although it has a fast rate of convergence if it converges, requires the second derivatives of the function to be evaluated and a subsequent inversion of the Hessian matrix. These are time consuming operations.

In the following section, a class of methods, known as the variable metric method [1, 6, 7, 8, 10, 11], is described in which no matrix inversion is required.

2.3 Variable Metric Method of Unconstrained Function Minimization

The variable metric method [8] is devised such that matrix H is not calculated and inverted from the Hessian matrix G. Instead an approximation H to G^{-1} is kept and updated at each iteration. The updating formula has the property that if the initial approximation H_0 is chosen

to be positive definite, the sequence of matrices H_1, H_2, ... generated are also positive definite. This guarantees convergence of the process as required by eqn. (2.15). Further, if the function $F(\underline{x})$ to be minimized is quadratic, after n steps, where n is the number of variables, H_n is precisely the inverse Hessian G^{-1}. If $F(\underline{x})$ is not quadratic, after n steps H is a good approximation to G^{-1}. Thus the method converges rapidly as soon as it gets close enough to make a quadratic approximation valid.

2.4 Derivation of Matrix Updating Formulae

The approximating matrix H_i, at the ith iteration, is modified [5] by adding a correction matrix, A_i, to form the new matrix H_{i+1} which gives a better approximation to the inverse Hessian G^{-1}. Thus

$$H_{i+1} = H_i + A_i \tag{2.17}$$

Consider a quadratic objective function

$$F(\underline{x}) = \tfrac{1}{2}\, \underline{x}^T G\underline{x} + \underline{b}^T\underline{x} + \underline{c} \tag{2.18}$$

where G is a $n \times n$ constant matrix, $\underline{b}$ and $\underline{c}$ are nth order constant vectors of coefficients. The gradient at the ith iteration is given by

$$\underline{g}_i = \underline{g}(\underline{x}_i) = G\underline{x}_i + \underline{b} \tag{2.19}$$

Let $\underline{y}_i$ be the difference of gradients at the (i+1)th and ith iterations,

$$\underline{y}_i = \underline{g}_{i+1} - \underline{g}_i$$

$$= G\underline{x}_{i+1} - G\underline{x}_i \qquad \text{from (2.19)} \tag{2.20}$$

$$= G\,\underline{\delta}_i \tag{2.21}$$

where

$$\underline{\delta}_i = \underline{x}_{i+1} - \underline{x}_i \tag{2.22}$$

Rearranging,

$$G^{-1}\underline{y}_i = \underline{\delta}_i \tag{2.23}$$

Since H_{i+1} should be a closer approximation to G^{-1}, the correction A_i is chosen such that the relation

$$H_{i+1}\,\underline{y}_i = \underline{\delta}_i \tag{2.24}$$

is satisfied.

Substituting eqn. (2.17) into eqn. (2.24) and rearranging give

$$A_i\,\underline{y}_i = \underline{\delta}_i - H_i\,\underline{y}_i \tag{2.25}$$

It is obvious from eqn. (2.25) that the choices of A_i are quite arbitrary. The simplest form that A can have such that eqn (2.25) is satisfied is

$$A_i = (\underline{\delta}_i - H_i\,\underline{y}_i)\,\underline{Z}_i^T \tag{2.26}$$

where $\underline{Z}_i$ satisfies the condition

$$\underline{Z}_i^T\,\underline{y}_i = 1 \tag{2.27}$$

A more general alternative is

$$A_i = \underline{\delta}_i\,\underline{q}_i^T - H_i\,\underline{y}_i\,\underline{\omega}_i^T \tag{2.28}$$

where $\underline{q}_i$ and $\underline{\omega}_i$ are chosen such that

$$\underline{q}_i^T \underline{y}_i = \underline{\omega}_i^T \underline{y}_i = 1 \tag{2.29}$$

Combining eqns. (2.17) and (2.28),

$$H_{i+1} = H_i + \underline{\delta}_i \underline{q}_i^T - H_i \underline{y}_i \underline{\omega}_i^T \tag{2.30}$$

Eqn. (2.30) gives a class of formulae for updating the H matrix depending on the ways that $\underline{q}$ and $\underline{\omega}$ are chosen. Two updating formulae are given below.

Formula (I) [11] - By letting

$$\underline{q}_i = \left(\frac{1}{\underline{\delta}_i^T \underline{y}_i}\right) \underline{\delta}_i \tag{2.31}$$

and

$$\underline{\omega}_i = \frac{H_i \underline{y}_i}{\underline{y}_i^T H_i \underline{y}_i} \tag{2.32}$$

where both eqns (2.31) and (2.32) satisfy eqn. (2.29), we have

$$H_{i+1} = H_i + \frac{\underline{\delta}_i \underline{\delta}_i^T}{\underline{\delta}_i^T \underline{y}_i} - \frac{H_i \underline{y}_i \underline{y}_i^T H_i}{\underline{y}_i^T H_i \underline{y}_i} \tag{2.33}$$

Formula (II) [6, 7, 10] - By letting

$$\underline{q}_i = \left(1 + \frac{\underline{y}_i^T H_i \underline{y}_i}{\underline{\delta}_i^T \underline{y}_i}\right) \frac{\underline{\delta}_i}{\underline{\delta}_i^T \underline{y}_i} - \frac{H_i \underline{y}_i}{\underline{\delta}_i^T \underline{y}_i} \tag{2.34}$$

and

$$\underline{\omega}_i = \left(\frac{1}{\underline{\delta}_i^T \underline{y}_i}\right) \underline{\delta}_i \tag{2.35}$$

where eqns. (2.34) and (2.35) also satisfy the conditions given by eqn (2.29).

We have

$$H_{i+1} = H_i + \left(1 + \frac{y_i^T H_i y_i}{\delta_i^T y_i}\right) \frac{\delta_i \delta_i^T}{\delta_i^T y_i} - \frac{\delta_i y_i^T H_i}{\delta_i^T y_i} - \frac{H_i y_i \delta_i^T}{\delta_i^T y_i}$$

or

$$H_{i+1} = \left(I - \frac{\delta_i y_i^T}{\delta_i^T y_i}\right) H_i \left(I - \frac{y_i \delta_i^T}{\delta_i^T y_i}\right) + \frac{\delta_i \delta_i^T}{\delta_i^T y_i} \qquad (2.36)$$

It is obvious from the expressions that the H matrix at the current step is obtained from the values of displacements, $\underline{\delta}$, and gradient difference, $\underline{y}$, at that iteration. The amount of computation involved is much less than that required by direct evaluation of the Hessian and subsequent matrix inversion.

2.5 Algorithm 1: VMM With Linear Search

An algorithm for locating the minimum of a function $F(\underline{x})$, using updating formulae derived in the previous section for the successive approximation of the H matrix is summarized. This algorithm, due to Davidon [8] and reformulated by Fletcher and Powell [11], involves a linear search subminimization process, i.e., at each iteration, the function is minimized along the search direction.

The iterative scheme can be stated as

$$\underline{x}_{i+1} = \underline{x}_i + \underline{\delta}_i \qquad (2.37)$$

At the ith iteration, $\underline{x}_i$ is known and hence the gradient $g(\underline{x}_i)$ is known. The direction of search is defined by

$$\underline{p}_i = - H_i \underline{g}_i \qquad (2.38)$$

where H_i, the deflection matrix, which deviates the search direction from that of the steepest descent direction, is evaluated using eqn. (2.33) or eqn. (2.36). The correction $\underline{\delta}_i$ to the position vector $\underline{x}_i$ is given by

$$\underline{\delta}_i = \alpha_i \underline{p}_i \tag{2.39}$$

where α_i is a positive scalar and is chosen such that $F(\underline{x}_i + \alpha_i \underline{p}_i)$ is a minimum along $(\underline{x}_i + \alpha \underline{p}_i)$,

i.e.,

$$\frac{d}{d\alpha} F(\underline{x}_i + \alpha \underline{p}_i)\Big|_{\alpha=\alpha_i} = 0$$

In other words, the orthogonal property

$$\underline{g}_{i+1}^T \underline{p}_i = 0 \tag{2.40}$$

holds for all i.

The new position vector is given by

$$\underline{x}_{i+1} = \underline{x}_i + \alpha_i \underline{p}_i \tag{2.41}$$

Setting $i = i+1$, the process is repeated until every component of $\underline{\delta}$ is less than a prescribed tolerance.

In the Fletcher and Powell algorithm [11], formula (I) of section (2.4) is used for updating matrix H. It is shown here that formula (II) of section (2.4), used by Fletcher [10] in an algorithm where linear search is not done, could also be used in an algorithm with linear search. A recurrence formula relating two successive search directions is

established. It is then deduced that two successive search directions contain an angle, θ, less than or at most equal to $\frac{\pi}{2}$. The case where $\theta = \frac{\pi}{2}$ corresponds to the steepest descent scheme.

Stability

It will be shown that the search direction defined by eqn. (2.38) is downhill. This means that the function value decreases at each step and thus the process is stable. Eqn. (2.38) gives

$$\underline{p}_i = - H_i \underline{g}_i$$

Because $\underline{g}_i$ is the direction of steepest ascent, the direction $\underline{p}_i$ will be downhill if and only if

$$\underline{p}_i^T \underline{g}_i = -\underline{g}_i^T H_i \underline{g}_i \qquad (2.42)$$

is negative.

To satisfy the above requirement, H_i must be positive definite. Thus to guarantee convergence of the process, one needs to prove that the sequence of H matrices generated by eqns. (2.33) and (2.36) are positive definite. The proofs follow.

For formula (I), eqn (2.33) gives

$$\underline{x}^T H_{i+1} \underline{x} = \underline{x}^T H_i \underline{x} + \frac{\underline{x}^T \underline{\delta}_i \underline{\delta}_i^T \underline{x}}{\underline{\delta}_i^T \underline{y}_i} - \frac{\underline{x}^T H_i \underline{y}_i \underline{y}_i^T H_i \underline{x}}{\underline{y}_i^T H_i \underline{y}_i} \qquad (2.43)$$

Define

$$\underline{s} = H^{\frac{1}{2}} \underline{x} \qquad \text{and} \qquad \underline{t} = H^{\frac{1}{2}} \underline{y}$$

then

$$\underline{x}^T H_{i+1} \underline{x} = \underline{s}^T \underline{s} + \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i} - \frac{(\underline{s}^T \underline{t})^2}{\underline{t}^T \underline{t}}$$

$$= \frac{(\underline{s}^T \underline{s})\ (\underline{t}^T \underline{t}) - (\underline{s}^T \underline{t})^2}{\underline{t}^T \underline{t}} + \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i}$$

$$> \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i} \qquad \text{using Schwarz inequality [16]}$$

But, by eqns. (2.20) and (2.40)

$$\underline{\delta}_i^T \underline{y}_i = \underline{\delta}_i^T \underline{g}_{i+1} - \underline{\delta}_i^T \underline{g}_i$$

$$= - \underline{\delta}_i^T \underline{g}_i$$

$$= \alpha_i \underline{g}_i^T H_i \underline{g}_i \qquad \text{from eqns (2.38) and (2.39)}$$

$$> 0 \qquad (2.44)$$

since H_i is assumed positive definite. Therefore,

$$\underline{x}^T H_{i+1} \underline{x} > 0 \qquad (2.45)$$

for all nontrival $\underline{x}$.

Next for formula (II), eqn. (2.36) gives

$$\underline{x}^T H_{i+1} \underline{x} = \underline{x}^T \left(I - \frac{\underline{\delta}_i \underline{y}_i^T}{\underline{\delta i}^T \underline{y}_i}\right) H_i \left(I - \frac{\underline{y}_i \underline{\delta}_i^T}{\underline{\delta}_i^T \underline{y}_i}\right) \underline{x} + \frac{\underline{x}^T \underline{\delta}_i \underline{\delta}_i^T \underline{x}}{\underline{\delta}_i^T \underline{y}_i} \qquad (2.46)$$

Let $B = I - \dfrac{\underline{y}\ \underline{\delta}_i^T}{\underline{\delta}_i^T \underline{y}_i}$ and $\underline{z} = B\underline{x}$ then $B^T = I - \dfrac{\underline{\delta}_i \underline{y}_i^T}{\underline{\delta}_i^T \underline{y}_i}$. Substituting

into eqn. (2.46) gives

$$\underline{x}^T H_{i+1} \underline{x} = \underline{x}^T B^T H_i \, B\underline{x} + \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i}$$

$$= \underline{z}^T H_i \, \underline{z} + \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i}$$

$$> \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i} \qquad \text{since } H_i \text{ is positive definite}$$

$$> 0 \qquad \text{since } \underline{\delta}_i^T \underline{y}_i > 0 \text{ from eqn. (2.44)}$$

Therefore

$$\underline{x}^T H_{i+1} \underline{x} > 0 \tag{2.47}$$

for all nontrival $\underline{x}$.

From eqns. (2.45) and (2.47), it is concluded that the H matrices generated by Formulae (I) and (II) are positive definite if the initial matrix H_o is chosen to be positive definite. Therefore, algorithms outlined in the previous paragraph using either updating formula are stable.

Recurrence Relation

The search direction at the (i + 1)th iteration is

$$\underline{p}_{i+1} = - H_{i+1} \, \underline{g}_{i+1} \tag{2.48}$$

Using eqn. (2.33) and the orthogonality property of eqn (2.40), we have

$$\underline{p}_{i+1} = \left(I - \frac{H_i \, \underline{y}_i \, \underline{y}_i^T}{\underline{y}_i^T H_i \, \underline{y}_i}\right) \underline{p}_i \tag{2.49}$$

$$= K_i \, \underline{p}_i \tag{2.50}$$

where

$$K_i = I - \frac{H_i \, \underline{y}_i \, \underline{y}_i^T}{\underline{y}_i^T \, H_i \, \underline{y}_i} \tag{2.51}$$

The derivation of the recurrence formula, eqn. (2.49), is given in Appendix A1 . The above mathematical relations for the search direction can be interpreted readily using geometrical concepts.

Matrix H in eqn. (2.48) measures the rotation of the search direction $\underline{p}$, from the direction of the gradient $\underline{g}$, at the current point, whereas matrix K in eqn. (2.50) corresponds to the rotation from the previous search direction. These relations are shown graphically in Fig. 2.5 and Fig. 2.6.

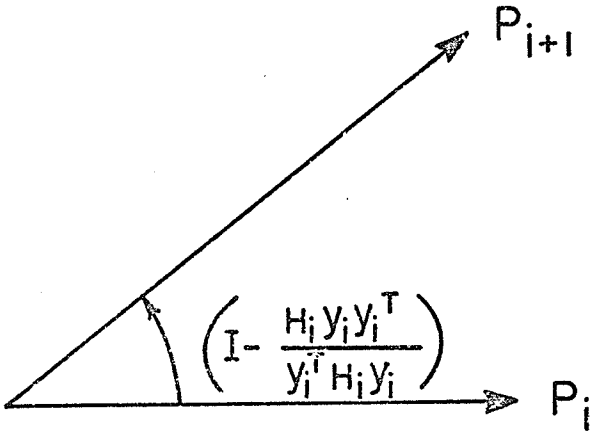


Fig. 2.5 Recurrence Relation

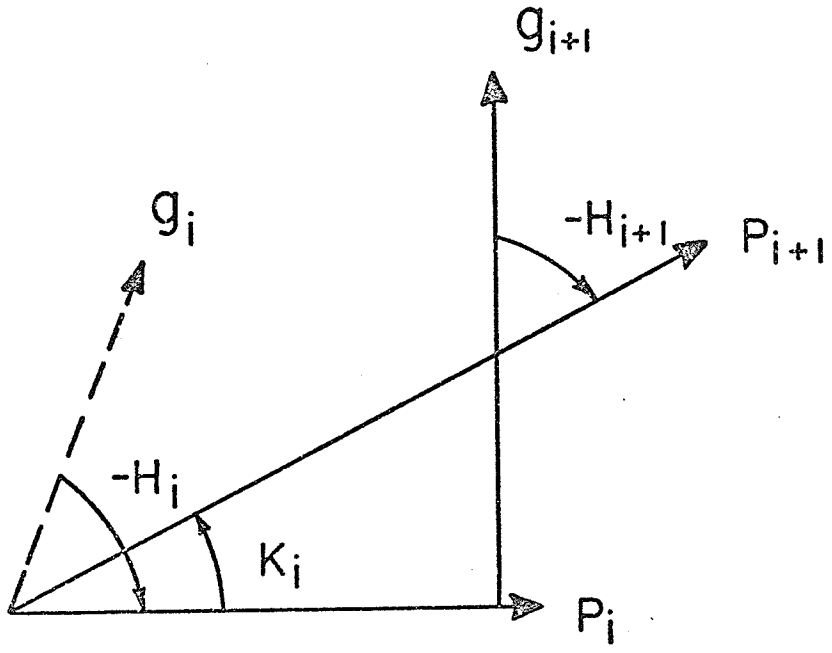


Fig. 2.6 Geometric Interpretation of the Relation Between Search Direction and Gradient vector

The following property can be deduced from the recurrence relation given by eqns. (2.50) and (2.51).

Property: The angle θ, between two successive search directions of the quadratic convergent minimization process, does not exceed 90°. The extreme case where $\theta = 90°$ corresponds to a steepest descent step. The statement is shown graphically in Fig. 2.7.

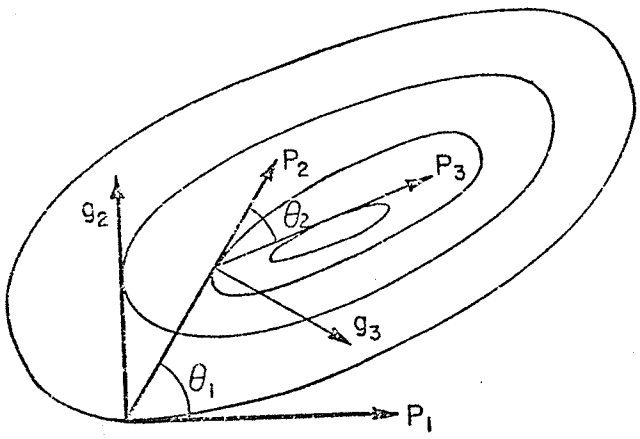


Fig. 2.7 Minimization Scheme of Algorithm 1

Proof

By definition, eqn. (2.51)

$$K_i = I - \frac{H_i \underline{y}_i \underline{y}_i^T}{\underline{y}_i^T H_i \underline{y}_i}$$

Premultiply by $\underline{y}_i^T$ and postmultiply by $\underline{y}_i$ lead to

$$\underline{y}_i^T K_i \underline{y}_i = \underline{y}_i^T \underline{y}_i - \frac{\underline{y}_i^T H_i \underline{y}_i \underline{y}_i^T \underline{y}_i}{\underline{y}_i^T H_i \underline{y}_i} = 0 \qquad (2.52)$$

Thus K_i is a semidefinite matrix, since $\underline{y}_i \neq 0$ generally. To show that K_i is positive semidefinite, postmultiplying eqn. (2.51) by H_i, i.e.,

$$K_i H_i = H_i - \frac{H_i \underline{y}_i \underline{y}_i^T H_i}{\underline{y}_i^T H_i \underline{y}_i} \qquad (2.53)$$

the quadratic form of the matrix of eqn. (2.53) is

$$\underline{x}^T K_i H_i \underline{x} = \underline{x}^T H_i \underline{x} - \frac{\underline{x}^T H_i \underline{y}_i \underline{y}_i^T H_i \underline{x}}{\underline{y}_i^T H_i \underline{y}_i} \qquad (2.54)$$

Define

$$\underline{s} = H^{\frac{1}{2}} \underline{x} \qquad \underline{t} = H^{\frac{1}{2}} \underline{y}$$

substituting into eqn. (2.54), we get

$$\underline{x}^T K_i H_i \underline{x} = \underline{s}^T \underline{s} - \frac{(\underline{s}^T \underline{t})^2}{(\underline{t}^T \underline{t})}$$

$$= \frac{(\underline{s}^T \underline{s})(\underline{t}^T \underline{t}) - (\underline{s}^T \underline{t})^2}{\underline{t}^T \underline{t}}$$

$$> 0 \qquad \text{by the Schwarz inequality} \qquad (2.55)$$

Since H_i is positive definite, we conclude that K_i is positive semidefinite from eqns. (2.52), (2.53) and (2.55).

From eqn. (2.50)

$$\underline{p}_{i+1}^T \underline{p}_i = \underline{p}_i^T K_i \underline{p}_i$$

Since K_i is proved to be positive semidefinite, we conclude that

$$\underline{p}_{i+1}^T \underline{p}_i \geq 0 \qquad (2.57)$$

2.6 Algorithm 2: VMM Without Linear Search

In algorithm 1, the multiplier α_i is taken as the value of α which minimizes $F(x_i + \alpha \underline{p}_i)$, that is, the function is minimized locally along the direction of search. This is usually done by evaluating the function and gradient for a number of different values of α and interpolating according to some strategy. Though the important properties of quadratic convergence and stability [11] are the consequences of linear search, considerable extra computing effort is required in finding α lead to an attempt to abandon the linear search so that only one evaluation of F and $\underline{g}$ per iteration is needed.

However, to guarantee nondivergence of a minimization process, it is important that the function value decreases monotonically. In other words, retention of positive definiteness in H is necessary. Moreover, the property of quadratic convergence ensures fast ultimate convergence of the process. It is therefore desirable to retain some guarantee that the H matrices tend to G^{-1}. It has been shown [10] that this requires that for quadratic functions the eigenvalues of H must tend monotonically to those of G^{-1}. The updating formulae derived in Section (2.4) satisfy

this requirement [10] and, thus, can be used in an algorithm not requiring linear search.

Stability

It has been shown from eqns. (2.15) and (2.16) that the minimization process is stable if matrices H generated from the updating formulae are positive definite. To guarantee the positive definiteness of matrix H generated by updating formulae of Section (2.4), the condition [10]

$$\underline{\delta}_i^T \underline{y}_i > 0 \tag{2.58}$$

is imposed in an algorithm where linear search is abandoned. The derivation of the above requirement is given below. It follows the same lines as the stability proof of Section (2.5) for the case with linear search.

For formula (I), from eqn. (2.33), we have

$$\underline{x}^T H_{i+1} \underline{x} = \underline{x}^T H_i \underline{x} + \frac{\underline{x}^T \underline{\delta}_i \underline{\delta}_i^T \underline{x}}{\underline{\delta}_i^T \underline{y}_i} - \frac{\underline{x}^T H_i \underline{y}_i \underline{y}_i^T H_i \underline{x}}{\underline{y}_i^T H_i \underline{y}_i} \tag{2.59}$$

Define $\underline{s} = H^{\frac{1}{2}} \underline{x}$, $\underline{t} = H^{\frac{1}{2}} \underline{y}_i$ eqn. (2.54) becomes,

$$\underline{x}^T H_{i+1} \underline{x} = \underline{s}^T \underline{s} + \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i} - \frac{(\underline{s}^T \underline{t})^2}{(\underline{t}^T \underline{t})}$$

$$= \frac{(\underline{s}^T \underline{s})(\underline{t}^T \underline{t}) - (\underline{s}^T \underline{t})^2}{\underline{t}^T \underline{t}} + \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i}$$

$$\geq \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i}$$

$$> 0 \tag{2.60}$$

if $\underline{\delta}_i^T \underline{y}_i > 0$ (2.61)

Therefore, for the process to be stable, it is necessary to update over an interval $\underline{\delta}_i$ for which $\underline{\delta}_i^T \underline{y}_i > 0$.

Next, for updating formula (II), from eqn. (2.36), we have

$$\underline{x}^T H_{i+1} \underline{x} = \underline{x}^T \left(I - \frac{\underline{\delta}_i \underline{y}_i^T}{\underline{\delta}_i^T \underline{y}_i}\right) H_i \left(I - \frac{\underline{y}_i \underline{\delta}_i^T}{\underline{\delta}_i^T \underline{y}_i}\right) \underline{x} + \frac{\underline{x}^T \underline{\delta}_i \underline{\delta}_i^T \underline{x}}{\underline{\delta}_i^T \underline{y}_i}$$

Let $B = \left(I - \frac{\underline{y}_i \underline{\delta}_i^T}{\underline{\delta}_i^T \underline{y}_i}\right)$ and $\underline{z} = B \underline{x}$ we have,

$$\underline{x}^T H_{i+1} \underline{x} = \underline{z}^T H_i \underline{z} + \frac{(\underline{x}^T \underline{\delta}_i)^2}{\underline{\delta}_i^T \underline{y}_i}$$

$$> \frac{(\underline{x}^T \underline{\delta}_i^T)^2}{\underline{\delta}_i^T \underline{y}_i}$$

since H_i is positive definite. Therefore,

$$\underline{x}^T H_{i+1} \underline{x} > 0 \tag{2.62}$$

if

$$\underline{\delta}_i^T \underline{y}_i > 0 \tag{2.63}$$

The same condition, eqns. (2.56) and (2.58) for stability is obtained for both formulae.

Both Algorithms 1 and 2 are applicable for solving load flow problems. Algorithm 2 without linear search generally has better convergence, in terms of number of function evaluations, than Algorithm 1 . The following chapter presents an application of the techniques described here for power system load flow problems.

3

solution of load flow problems

Briefly stated, a load flow study is the determination of the voltage, current, power and power factor or reactive power at various points in an electric network under existing or contemplated conditions of normal operation [29]. Mathematically, the problem involves the solution of a set of nonlinear equations satisfying the power and voltage requirements of the load and generation points. The problem can be solved by many different methods [18, 26, 28, 29]. The main methods in use today are those based on successive overrelaxation, abbreviated SOR, and the Newton-Raphson method, abbreviated NR. In this chapter, nonlinear programming formulation of the load flow is introduced. Then it is solved by techniques discussed in the previous chapter.

A section on SOR and NR load flow formulations and solutions is included for the convenience of later comparison and discussion.

3.1 System Equations

Consider a network with n busbars, Fig. 3.1. The system can be represented by a set of node equations which relates the voltages and currents in the network.

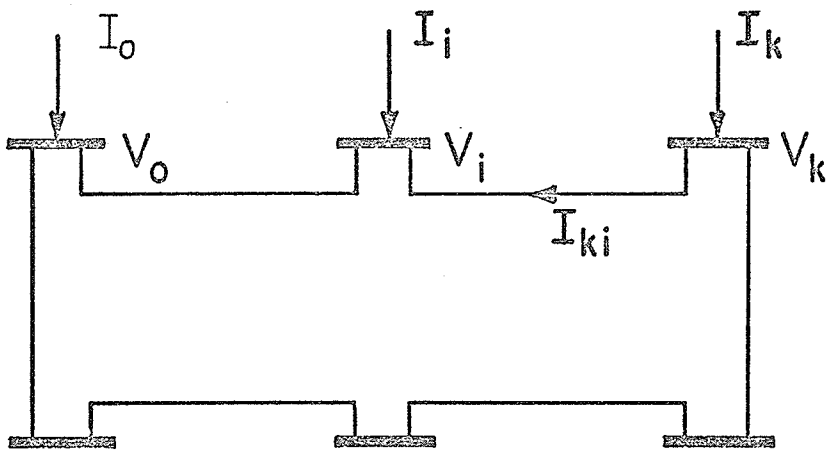


Fig. 3.1 One-line Diagram of a Power System Network

The branch current I_{ki}, flowing between nodes k and i is

$$I_{ki} = y_{ki} (V_k - V_i) \tag{3.1}$$

where y_{ki} is the admittance between nodes i and k and V_k and V_i are node voltages measured with respect to neutral. At node k, the node equation can be written as

$$I_k = \sum_{\substack{i=1 \\ i \neq k}}^{n} (V_k - V_i)\, y_{ki} \qquad k = 2, 3, \ldots, n$$

$$= \sum_{i=1}^{n} Y_{ki} V_i \qquad (3.2)$$

where

$$Y_{kk} = \sum_{i \neq k} y_{ki} \qquad (3.3)$$

and $$Y_{ki} = - y_{ki} \qquad (3.4)$$

The voltages , V, currents , I , and admittances , Y , in the equations are complex numbers and are represented as follows:

$$I_k = a_k + j\, b_k \qquad (3.5a)$$

$$V_k = e_k + j\, f_k \qquad (3.5b)$$

$$Y_{ki} = G_{ki} - j\, B_{ki} \qquad (3.5c)$$

The power at node k is

$$S_k = V_k\, I_k^{*} \qquad (3.6)$$

where * denotes complex conjugate

$$S_k = P_k + j\, Q_k \qquad (3.7)$$

and $I_k^{*} = a_k - j\, b_k$

In the load-flow study, three types of nodes are considered. At the slack node or slack bus, the voltage magnitude and phase angle are specified. The load nodes or load buses are those at which active power and reactive power are specified. The generator buses or voltage-controlled buses are those at which the real power and voltage magnitude are specified.

3.2 Conventional Load Flow Formulation and Solution Techniques

Two commonly used iterative methods, SOR and Newton-Raphson, of solving load flow problems are described. The SOR load flow solution [14, 31] is based on correcting the system voltages successively and using a small voltage tolerance to ensure satisfactory power mismatches at the nodes. The Newton-Raphson method[1] for load flow [30, 34, 35] involves, at each iteration step, the solution of a set of linear equations expressing the relationship between the changes in real power and reactive power (or voltage magnitude in the case of voltage-controlled node) and the components of node voltages.

Successive Overrelaxation Formulation and Solution

From eqns. (3.2) and (3.6), we have

$$s_k^* = V_k^* \sum_{i=1}^{n} Y_{ki} V_i \tag{3.8}$$

Let the scheduled power be $s_{ks} = P_{ks} + j Q_{ks}$. When the solution is complete, the calculated power matches the scheduled power, i.e.,

$$(P_{ks} + jQ_{ks})^* = V_k^* \sum_{i=1}^{n} Y_{ki} V_i \tag{3.9}$$

Rearranging, we get

$$V_k = \frac{1}{Y_{kk}} \left[\frac{(P_{ks} - j Q_{ks})}{V_k^*} \right] - \sum_{i \neq k} Y_{ki} V_i \tag{3.10}$$

The iterative process is initiated by assigning estimated voltages for all buses except the slack bus where the voltage is fixed. Eqn. (3.10) then gives the corrected value of voltage at the kth bus based on the scheduled power s_{ks} and the best previous voltage values for the corresponding buses.

---

1 This should not be confused with the Generalized Newton-Raphson method of unconstrained function minimization which finds the zeros of the gradient of a function.

Once the corrected voltage at each bus is found, it is used in calculating the voltage correction at the next. One iteration of the SOR scheme corresponds to the process in which the voltages at all nodes have been consecutively corrected once. The process is repeated until the change in voltages between two successive iterations at each node is smaller than a certain specified tolerance limit. The total power mismatch is then calculated. This must be zero or less than a certain preassigned tolerance at the solution point. If it is not, further iterations are necessary. Otherwise the process is terminated.

In the case of a voltage controlled node, the reactive power is obtained from the imaginary part of eqn. (3.6), i.e.,

$$Q_k = I_m \{V_k \; I_k^*\}$$

$$Q_k = I_m \{V_k \left[\sum_{i=1}^{n} Y_{ki} \; V_i\right]^*\} \tag{3.11}$$

Substituting eqn. (3.5), the reactive power is

$$Q_k = (e_k^2 + f_k^2) \; B_{kk} + \sum_{\substack{i=1 \\ i\neq k}}^{n} \{f_k(e_i G_{ki} + f_i B_{ki}) - e_k(f_i G_{ki} - e_i B_{ki})\} \tag{3.12}$$

The complex power at a voltage-controlled bus is

$$S_{ks} = P_{ks} + j\left[(e_k^2 + f_k^2) \; B_{kk} + \sum_{\substack{i=1 \\ i\neq k}}^{n} \{f_k(e_i G_{ki} + f_i B_{ki}) - e_k(f_i G_{ki} - e_i B_{ki})\}\right] \tag{3.13}$$

where e_k and f_k are the components of voltage at node k and must satisfy the relation

$$e_k^2 + f_k^2 = (|V_k|_{specified})^2 \tag{3.14}$$

where $|V_k|_{specified}$ is the magnitude of the specified voltage at node k.

Newton-Raphson Formulation and Solution

Substituting eqn. (3.2) into eqn. (3.6), we have

$$S_k^* = V_k^* \sum_{m=1}^{n} Y_{km} V_m \qquad k = 2, 3, \ldots, n \tag{3.15}$$

Bus No. 1 being the slack bus.

Substituting $S_k = P_k + j Q_k$

$V_k = e_k + j f_k$

$Y_{km} = G_{km} - j B_{km}$

into eqn. (3.15) and separating real and imaginary parts,

$$P_k = e_k \sum_{m=1}^{n} (G_{km} e_m + B_{km} f_m) + f_k \sum_{m=1}^{n} (G_{km} f_m - B_{km} e_m) \tag{3.16a}$$

and

$$Q_k = f_k \sum_{m=1}^{n} (G_{km} e_m + B_{km} f_m) - e_k \sum_{m=1}^{n} (G_{km} f_m - B_{km} e_m) \tag{3.16b}$$

If P_{ks} and Q_{ks} are the real and imaginary parts of scheduled power at node k, the set of 2(n - 1) nonlinear equations to be solved is

$$\left.\begin{aligned} r_k &= P_{ks} - P_k = 0 \\ r_{k+n} &= Q_{ks} - Q_k = 0 \end{aligned}\right\} \quad k = 2, 3, \ldots, n \tag{3.17}$$

In the case of a voltage-controlled node, the reactive power equation at that node is replaced by the equation

$$r_{k+n} = |V_k| - (e_k^2 + f_k^2)^{\frac{1}{2}} \tag{3.18}$$

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

$$\begin{bmatrix} r_2 \\ \vdots \\ r_n \\ r_{n+2} \\ \vdots \\ r_{2n} \end{bmatrix} = \begin{bmatrix} \frac{\partial r_2}{\partial e_2} & \cdots & \frac{\partial r_2}{\partial e_n} & \frac{\partial r_2}{\partial f_2} & \cdots & \frac{\partial r_2}{\partial f_n} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial r_n}{\partial e_2} & \cdots & \frac{\partial r_n}{\partial e_n} & \frac{\partial r_n}{\partial f_2} & \cdots & \frac{\partial r_n}{\partial f_n} \\ \frac{\partial r_{n+2}}{\partial e_2} & \cdots & \frac{\partial r_{n+2}}{\partial e_n} & \frac{\partial r_{n+2}}{\partial f_2} & \cdots & \frac{\partial r_{n+2}}{\partial f_n} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial r_{2n}}{\partial e_2} & \cdots & \frac{\partial r_{2n}}{\partial e_n} & \frac{\partial r_{2n}}{\partial f_2} & \cdots & \frac{\partial r_{2n}}{\partial f_n} \end{bmatrix} \begin{bmatrix} \Delta e_2 \\ \vdots \\ \Delta e_n \\ \Delta f_2 \\ \vdots \\ \Delta f_n \end{bmatrix}$$

or, in matrix form,

$$\underline{r} = J\ \underline{\Delta e} \tag{3.19}$$

where the coefficient matrix J is the matrix of first derivatives and is

called the Jacobian matrix.

The voltage correction at the mth iteration is given by

$$\underline{\Delta e}_m = J_m^{-1} \underline{r}_m \tag{3.20}$$

Hence, the new estimates for bus voltages are

$$\underline{e}_{m+1} = \underline{e}_m + J_m^{-1} \underline{r}_m \tag{3.21}$$

or

$$\underline{e}_{m+1} = \underline{e}_m + t\, J_m^{-1} \underline{r}_m \tag{3.22}$$

where t is a scalar multiplier chosen to prevent the process from diverging. The solution of the set of equations, eqn. (3.20), at each iteration involves the evaluation and inversion of the coefficient matrix. A modified approach, based on Broyden's variation of Newton's method [4, 15], is used in which no matrix inversion after the first iteration is required. The approximate inverse Jacobian matrix is computed from the function values $\underline{r}$ using the updating formula:

$$H_{m+1} = H_m + \frac{t\, \underline{p}_m \underline{p}_m^T H_m}{\underline{p}_m^T H_m \underline{y}_m} - \frac{H_m \underline{y}_m \underline{p}_m^T H_m}{\underline{p}_m^T H_m \underline{y}_m} \tag{3.23}$$

where

$$H_m = J_m^{-1}$$

$$\underline{p}_m = H_m \underline{r}_m$$

and

$$\underline{y}_m = \underline{r}_{m+1} - \underline{r}_m$$

It is noted from the expression that no additional function evaluation is required beyond those that would be needed if J is not changed, for

updating the inverse Jacobian matrix.

3.3 Nonlinear Programming Formulation of Load Flow

Since solution of the load flow problem must satisfy the condition that power mismatches at all nodes are zero, nonlinear programming can be used for solving load flow problems by defining an objective function from the set of $2(n - 1)$ nonlinear equations, eqns. (3.16) and (3.17), which describe the system. The point at which the function is minimized coincides with the solution of the equations.

Referring to the one-dimensional example, Fig. 2.1 of Section (2.2), it is clear that if a function F is constructed such that the set of $2(n - 1)$ nonlinear equations, eqn. (3.17), are the first partial derivatives of F with respect to the variables, i.e.,

$$r_k = \frac{\partial F}{\partial e_k} \qquad \text{and} \qquad r_{k+n} = \frac{\partial F}{\partial f_k} \tag{3.24}$$

the Generalized Newton-Raphson method of Section (2.2) for minimizing the function F so constructed, and the Newton-Raphson method of Section (3.2) for solving the system of nonlinear equations, both give identical steps. In the load flow problem, an objective function F satisfying eqn. (3.24) is not available. A least p-th formulation is used.

The objective function is defined by

$$\begin{aligned} F &= \sum_{k=2}^{n} \left(|r_k|^p + |r_{k+n}|^p \right) \\ &= f(e_2, \ldots, e_n, f_2, \ldots, f_n) \end{aligned} \tag{3.25}$$

where p is a positive real number which governs the degree of convexity

of the function.

$$r_k = p_{ks} - [e_k \sum_{m=1}^{n} (G_{km}\, e_m + B_{km}\, f_m) + f_k \sum_{m=1}^{n} (G_{km}\, f_m - B_{km}\, e_m)]$$

and

$$r_{k+n} = Q_{ks} - [f_k \sum_{m=1}^{n} (G_{km}\, e_m + B_{km}\, f_m) - e_k \sum_{m=1}^{n} (G_{km}\, f_m - B_{km}\, e_m)]$$

The elements of the gradient vectors are

$$\frac{\partial F}{\partial e_j} = p \sum_{k=2}^{n} [|r_k|^{p-1} \frac{\partial r_k}{\partial e_j} \text{sign } (r_k)] + p \sum_{k=2}^{n} [|r_{k+n}|^{p-1} \frac{\partial r_{k+n}}{\partial e_j} \text{sign } (r_{k+n})]$$

$$= (-p) [|r_j|^{p-1} \sum_{m=1}^{n} (G_{jm} e_m - B_{jm} f_m) \text{sign}(r_j)$$

$$+ |r_{j+n}|^{p-1} \sum_{m=1}^{n} (G_{jm} f_m + B_{jm} e_m) \text{ sign } (r_{j+n})]$$

$$+ (-p) \sum_{k=2}^{n} [|r_k|^{p-1} (G_{kj} e_k - B_{kj} f_k) \text{sign}(r_k)$$

$$+ |r_{k+n}|^{p-1} (G_{kj} f_k + B_{kj} e_k) \text{ sign } (r_{k+n})] \qquad (3.26)$$

and

$$\frac{\partial F}{\partial f_j} = p \sum_{k=2}^{n} [|r_k|^{p-1} \frac{\partial r_k}{\partial f_j} \text{sign}(r_k) + p \sum_{k=2}^{n} [|r_{k+n}|^{p-1} \frac{\partial r_{k+n}}{\partial f_j} \text{sign}(r_{k+n})]$$

$$= (-p) [|r_j|^{p-1} \sum_{m=1}^{n} (G_{jm} f_m - B_{jm} e_m) \text{sign}(r_j)$$

$$+ |r_{j+n}|^{p-1} \sum_{m=1}^{n} (G_{jm} e_m + B_{jm} f_m) \text{sign}(r_{j+n})]$$

$$+ (-p) \sum_{k=2}^{n} [|r_k|^{p-1}(B_{kj}e_k+G_{kj}f_k)\text{sign}(r_k) + |r_{k+n}|^{p-1}(B_{kj}f_k-G_{kj}e_k)\text{sign}(r_{k+n})] \quad (3.27)$$

This formulation of load flow, based on power mismatches together with a solution process which seeks directly the set of voltage values satisfying the terminal conditions, is physically more meaningful than the SOR iterative method where each individual voltage is corrected assuming that all others are already correct and without any control on the mismatches.

A number of standard test systems were solved using both Algorithms 1 and 2. The results are included in the next section.

3.4 Numerical Results and Discussion

The following systems were used to test the method:

(a) a five-bus system from Stagg and El-Abiad [28];

(b) a six-bus system of Ward and Hale [35];

(c) the IEEE standard 14-bus test system [13]; and

(d) the IEEE standard 30-bus test system [13].

All cases were started with a flat voltage profile of (1 + j0) per unit. All the voltage, power, reactive power and admittance data are per unit quantities throughout the chapter. The tests were taken to a high solution precision to study the complete response of the nonlinear programming approach.

The load flow function, as defined by eqn. (3.25), is the least p-th sum of power mismatches. Tests were carried out to determine the dependence of convergence rate on p. The results of this test using

Algorithm 2 on the 5-bus system are given in Table 3.1.

Table 3.1

Convergence Rate vs. p

Exit Criterion = 10^{-10}

p	No. of Iterations	No. of Function Evaluations
1.3	41	70
1.8	25	28
2.0	13	15
2.2	20	22
2.5	41	43

It is noted from Table 3.1 that the value of p has an important effect on the rate of convergence. Numerical results presented later in the section and in the following sections are based on a least square definition of load flow function which is considered to give the optimum convergence rate of the solution process. For large p values, the function tends to become flat in the region $F < 1$, and a slower rate of convergence is expected in this region. Small p value improves the convergence rate in the region $F < 1$, but has poorer convergence in the region $F > 1$. This is shown in Fig. 3.2 for a one dimensional case. Tests had been carried out using different p values for regions $F > 1$ and $F < 1$. This does result in faster convergence but the improvement is not significant.

Tables 3.2 and 3.3 summarize the results obtained by the nonlinear programming approach using two versions of the variable metric method and the normal iterative approach using the successive overrelaxation method. All tests were run on an IBM 360/65 computer with double precision. A

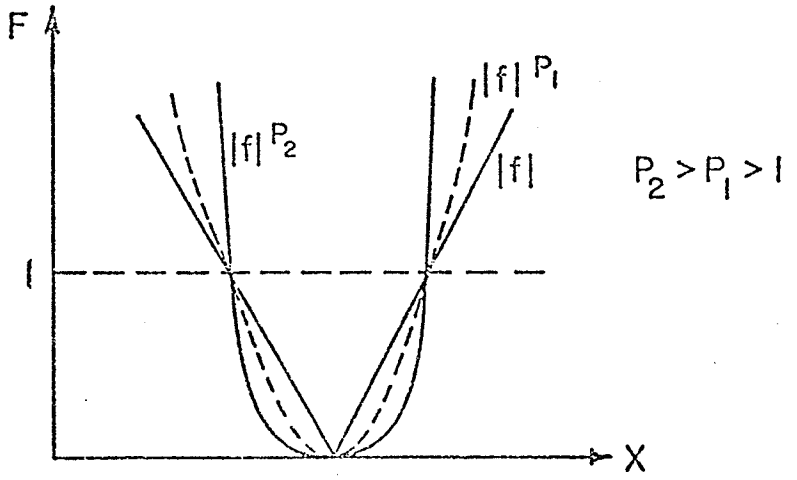


Fig. 3.2 Effect of p on the Convexity of the Objective Function

listing of the subroutines is given in Appendix A3. In the SOR process, the rate of convergence is very sensitive to the choice of the accelerating factor, poor choices could cause divergence. Whereas in the nonlinear programming approach, an acceleration factor is not required. The accuracy of the solution is measured by the degree to which the terminal condition of the problem is satisfied, that is, by the total power mismatch ε, at all nodes.

Table 3.2

Convergence Comparison of Solutions

Using Algorithm 1 and Algorithm 2

Exit Criterion, $\varepsilon^2 = 10^{-10}$

	Algorithm 1		Algorithm 2	
System Size	No. of Iterations	No. of Function Evaluations	No. of Iterations	No. of Function Evaluations
5	11	24	13	15
6	15	38	20	22
14	31	70	36	41
30	66	152	74	79

Table 3.3

Comparison of Solutions Using

Algorithm 2 and SOR

Exit Criterion, $\varepsilon^2 = 10^{-10}$

	Algorithm 2			SOR			
System Size	No. of Iterations	Computation Time (sec.)	Mismatch ε^2	No. of Iterations	ω_{opt}	Computation Time (sec.)	Mismatch ε^2
5	13	0.199	$.34\text{x}10^{-10}$	19	1.4	0.055	$.90\text{x}10^{-11}$
6	20	0.410	$.39\text{x}10^{-11}$	27	1.5	0.177	$.28\text{x}10^{-11}$
14	36	5.690	$.33\text{x}10^{-12}$	54	1.6	1.190	$.71\text{x}10^{-11}$
30	74	56.390	$.76\text{x}10^{-13}$	87	1.8	9.220	$.72\text{x}10^{-11}$

Table 3.4 presents data showing the order of the minimum total power mismatch squared ε^2 that can be obtained by SOR. The corresponding number of iterations required by VMM and SOR to achieve this accuracy is included. It is noted that a small increase in accuracy requires much more computation.

Table 3.4

Comparison of Number of Iterations Required

for Solution with Highest Accuracy, ε^2, Obtainable by SOR

System Size	ε^2	Algorithm 2 Iterations	SOR Iterations
5	$.75 \times 10^{-13}$	13	35
6	$.11 \times 10^{-12}$	20	46
14	$.15 \times 10^{-12}$	36	120

Table 3.5 summarizes the results obtained by the Newton-Raphson method without implementing the optimally ordered elimination scheme [30].

Table 3.5

Results of the Newton-Raphson Method

System Size	No. of Iterations	Computation Time (sec.)	Mismatch ε^2
5	3	0.095	0.50×10^{-9}
6	4	0.160	0.14×10^{-10}
14	4	1.570	0.20×10^{-9}
30	4	13.300	0.70×10^{-9}

It has been reported [30] that the Newton-Raphson method of load flow with the implementation of the optimally ordered elimination scheme requires less computer time than the SOR method. Further, the normal Newton-Raphson method involves the formation and solution of a system of linear equations or inversion of the coefficient matrix, the Jacobian, at each iteration. The modification, based on Broyden's variation of Newton's method, can be made such that the Jacobian matrix is formed and is then inverted only at the first iteration. The inverse Jacobian is then updated at subsequent iterations using eqn. (3.23) which involves only a few matrix-vector multiplications. It is estimated that computation time is approximately half of that required by the normal method. It can be seen from Tables 3.3 and 3.5 that in both the SOR method and the VMM method, the number of iterations increases with system size whereas the number of iterations of the Newton-Raphson method is independent of system size. Fig. 3.3 shows that variable metric method requires fewer iteration steps than SOR to attain a solution with the same accuracy. However, as the number of operations performed at each iteration step of the VMM algorithm is larger than that required by one SOR iteration, the

time per iteration of VMM is greater than that of SOR.

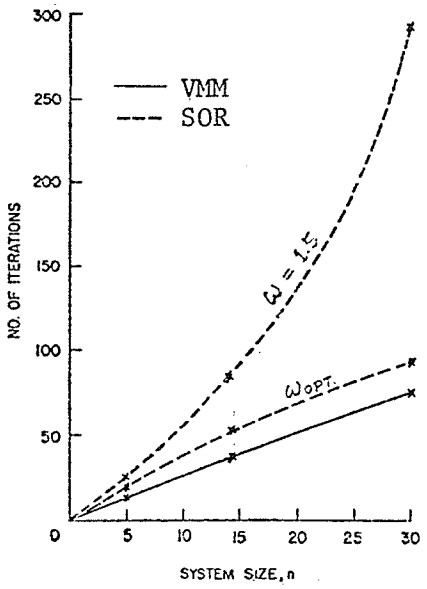


Fig. 3.3 Number of Iterations vs. System Size

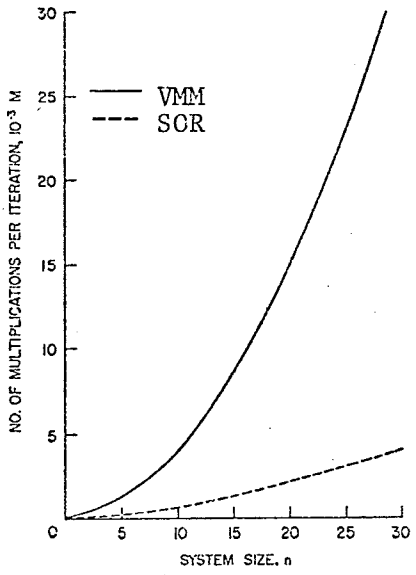


Fig. 3.4 Number of Multiplications per Iteration M vs. System Size n

$M = 36(n + 1/3)^2 - 4$ for VMM (solid line)
$M = 4(n + 2)^2 - 16$ for SOR (dashed line)

The computation time required to solve the problem depends to a certain extent on the programming techniques. The data for the overall computation time given in Tables 3.3 and 3.5 is included to give a picture of the relative amount of time required by the three methods which is after all our main interest. Appendix A2 analyses and compares the amount of computations per iteration, in terms of the number of multiplications involved, of the SOR and VMM methods for solving load flow problems. The rate of increase of arithmetic operations, and hence computation time, per iteration with system size for the VMM and the SOR methods is shown in Fig. 3.4 Fig. 3.5 shows the breakeven curve

$$K = \frac{36(n + \frac{1}{3})^2 - 4}{4(n + 2)^2 - 16}$$

where n is the system size. The condition for VMM computation time to be less than SOR is that the ratio of SOR iterations N_s, to VMM iterations N_v, has a value greater than K, i.e.,

$$\frac{N_s}{N_v} > K$$

Wallach [33] claims that the reformulation of load flow problem as an optimization problem may reduce the required computer time. It is concluded from the experimental data obtained that computation time is unlikely to be reduced using the VMM approach.

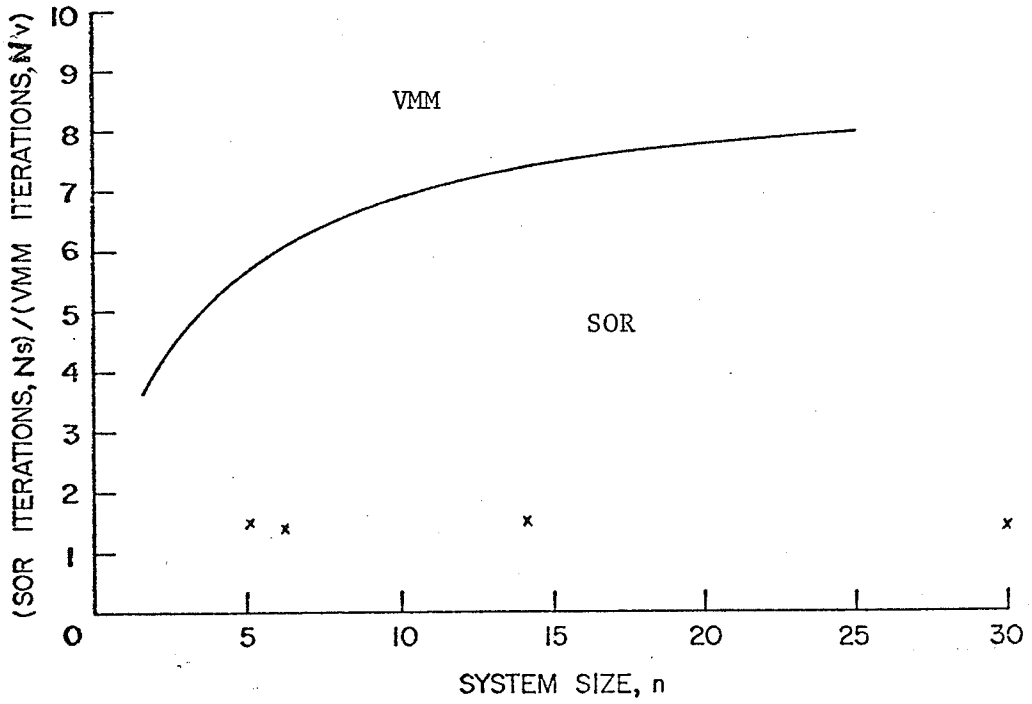


Fig. 3.5 The Breakeven Curve.
Points indicated are the ratios of number of iterations of ω_{opt} to VMM results of Fig. 3.3.

The convergence characteristics of the two methods, VMM and SOR, are given in Figs. 3.6 and 3.7, respectively. The SOR convergence characteristics as shown in Fig. 3.7 for the 5-bus system is different from the nonlinear programming approach and is based on the variation of voltages between

two successive iterations. In both the NR and the VMM methods, voltages are varied in order to reduce power mismatches.

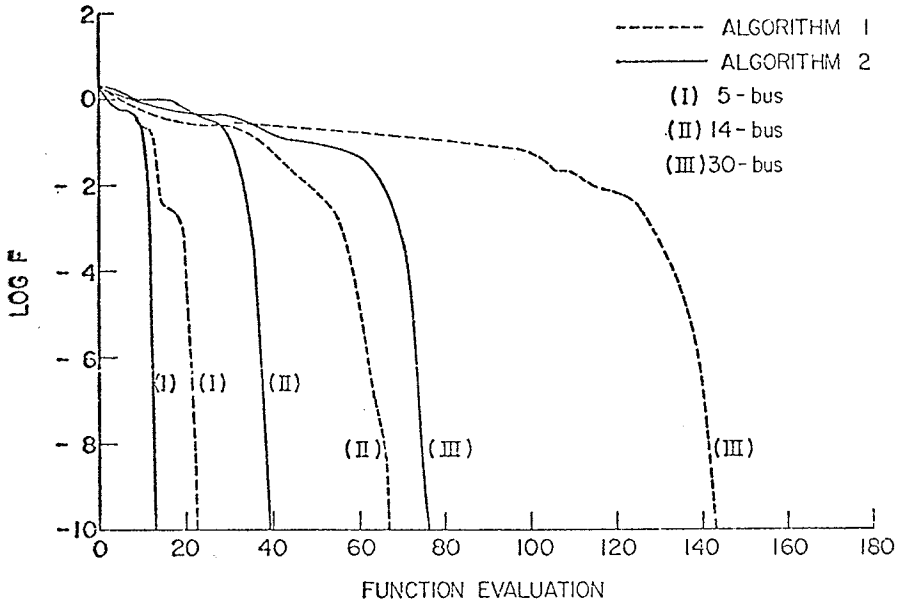


Fig. 3.6 VMM Convergence Characteristics

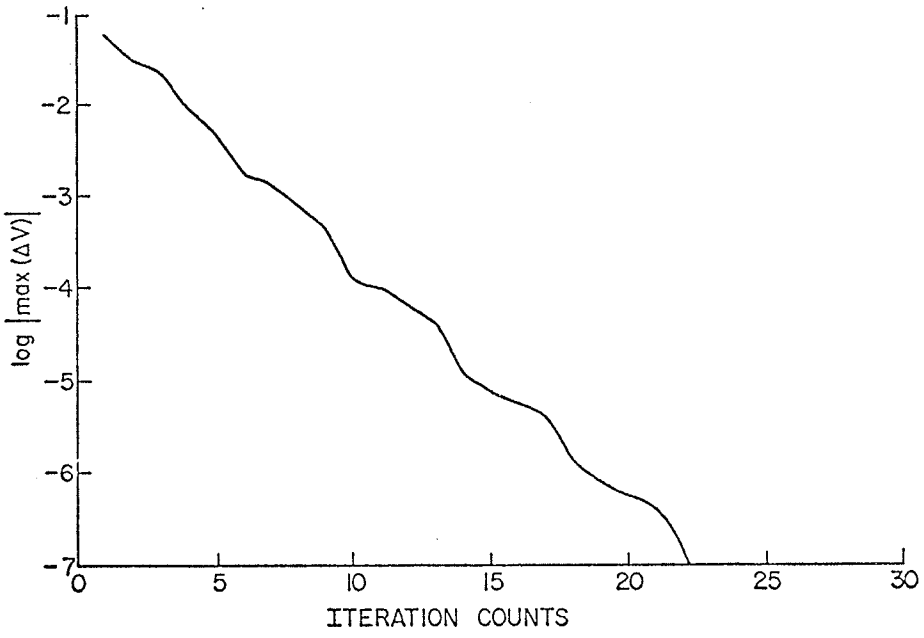


Fig. 3.7 SOR Convergence Characteristics

It is shown from the curves of Fig. 3.6, the convergence characteristic of VMM, that convergence is fast for the first few steps, followed by a very slow region and finally a steep curve indicating fast rate of convergence as the solution is approached. A large proportion of the computation time is spent in this middle section. This indicates that with the formulation of the load flow function given in section (3.3), there is a region remote from the solution which is nearly stationary, thus causing small steps to be taken for a number of iterations. Inspite of the above mentioned somewhat unfavorable feature, the negative slopes of the curves in Fig. 3.6 reveal the important property of the monotonic convergence of the process. In other words, the method provides corrections which always produce better values for the variables.

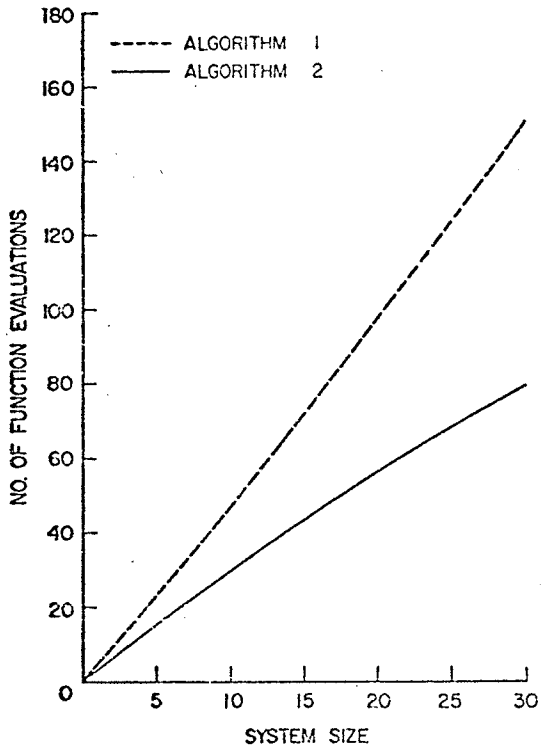


Fig. 3.8 Number of Function evaluations vs. system size

Both algorithms of VMM were tested. Algorithm 2 without linear search converges to the solution requiring about 60% of the total number of function evaluations of that required by algorithm 1 with linear search [27] as shown in Fig. 3.8.

A final comment on the experimental results is that a second solution can be detected by VMM, whereas this is not shown by the SOR method. The Newton-Raphson method diverges for a poor

starting point. The nonlinear programming approach always converges irrespective of the starting value, although it may not converge to the physical solution. As an example, using the starting voltage value of (0.5 + j0) instead of (1 + j0), the following busbar voltages were obtained as the solution to the 5-bus system.

Bus No.	Voltages
1	1.060000 + j0
2	0.567100 - j0.049820
3	0.107033 - j0.078465
4	0.019248 - j0.060000
5	0.189300 -j0.1442670

Although the voltage values given above are not feasible operating points, the result gives an indication that a second feasible solution near the first one can exist. This causes system instability. Such a test is, therefore, important at the system design stage to ensure that the system is stable.

The systems chosen as examples in this section to test the method are more or less well-behaved systems. The most desirable characteristic that the method always converges for systems of all kinds is shown in the next section where a 13-bus system with negative transfer reactance branches with which the SOR method diverges is solved.

3.5 SOR Divergent Case

In this section, a 13-bus system with negative reactance branches, solved by A. Brameller using matrix projection method, is solved by the SOR and the VMM methods. It is shown that the system diverges with the SOR method. However, solution was obtained by VMM with no difficulty.

The system is connected by 13 lines and is supplied by 5 generators. The network diagram is shown in Fig. 3.9 with the specified terminal conditions and line impedance data given in Tables 3.6 and 3.7, respectively.

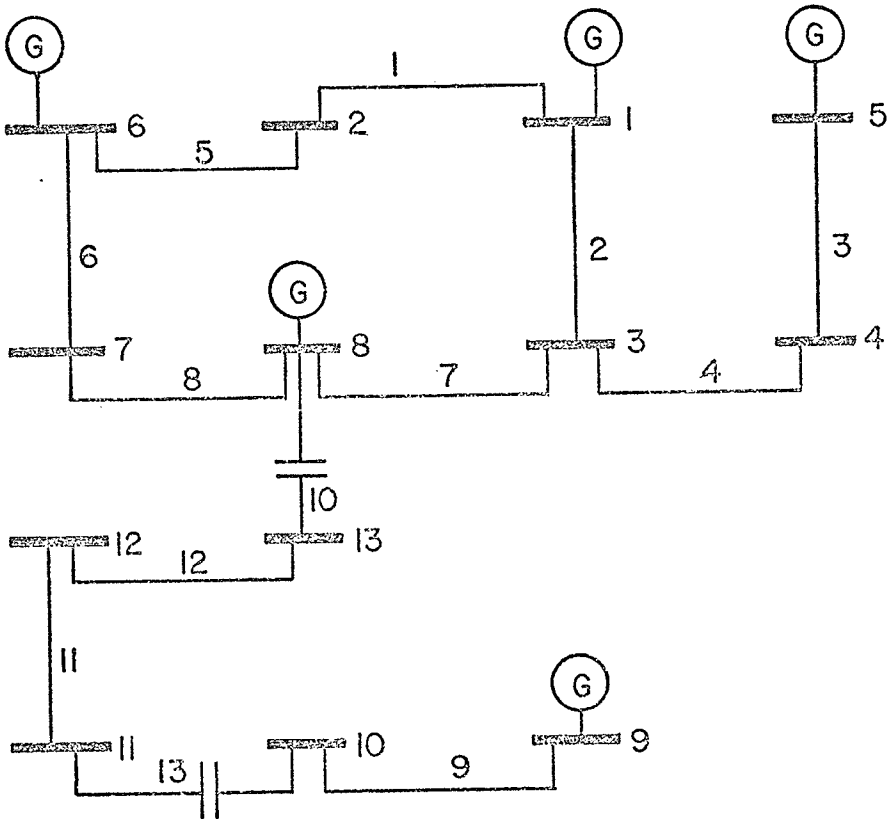


Fig. 3.9 Network Diagram

Both Algorithms 1 and 2 of VMM are used to solve the problem. Solution with a total power mismatch of order 10^{-10} is obtained after 74 function evaluations using Algorithm 1, whereas Algorithm 2 requires only 40 function evaluations. The convergence characteristics for both cases are given in Fig. 3.10. It is shown in Table 3.8 that the SOR process

diverges in this case. The terminal conditions of the solution obtained by VMM are included in Table. 3.9.

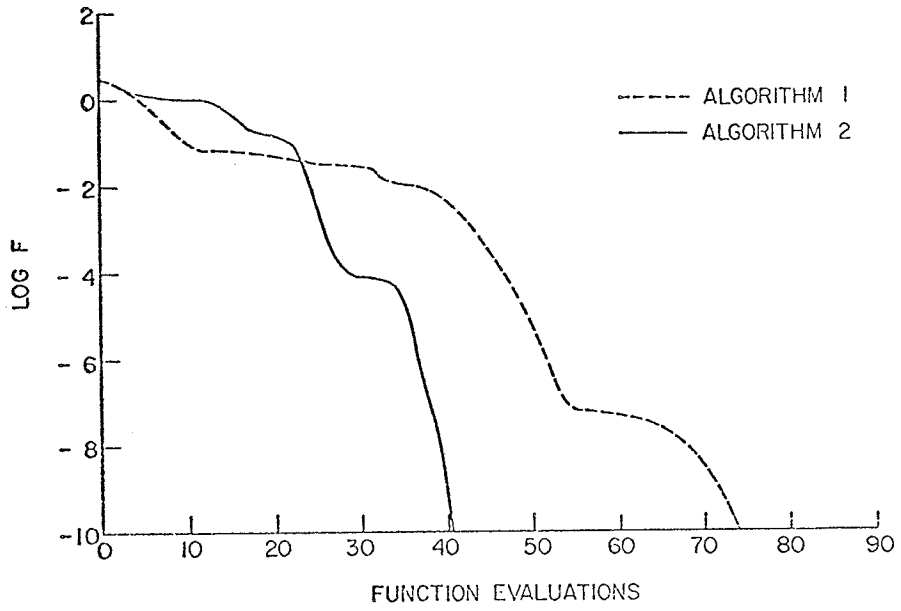


Fig. 3.10 VMM Convergence Characteristic (13-bus system)

Table 3.6

Busbar Data for 13-node System

Bus. No.	Volt. Mag.	Generation	Load	
		MW	MW	MVAR
1	1.000	-	1650	560
2	-	-	0	0
3	-	-	0	0
4	-	-	0	0
5	1.000	0	0	0
6	1.037	450	50	30
7	-	-	0	0
8	1.100	0	0	0
9	0.943	50	0	0
10	1.100	0	0	0
11	-	-	50	30
12	-	-	50	32
13	-	-	0	0

Bus No. 1 is the slack node

Table 3.7

Line Impedance Data

(1000 MVA Base)

Branch No.	R	X	B
1	.0040	.0850	.0
2	.0040	.0947	.0
3	.0040	.0947	.0
4	.0074	.1430	.436
5	.0481	.4590	.246
6	.0090	.1080	.016
7	.0121	.2330	.712
8	.0	.1500	.0
9	.0105	.2020	.620
10	.0	-.1500	.0
11	.0086	.1660	.508
12	.0075	.1460	.448
13	.0	-.1500	.0

Table 3.8

Convergence Comparison

Iteration No.	Algorithm 2 of VMM (Power Mismatch Squared)	SOR (Max. Vol. Disp.)
0	2.78378	6.703451
5	1.16962	0.7477635×10^{8}
10	1.03761	0.7816136×10^{15}
15	0.288843	0.7494661×10^{22}
20	0.761436×10^{-1}	
25	0.215440×10^{-3}	
30	0.736803×10^{-4}	
35	0.518713×10^{-6}	
40	0.886626×10^{-12}	

Table 3.9

Solution for Terminal Conditions

Bus No.	Voltage	P (MW)	Q (MVAR)
1	1.000 + j0	-338	-1059
2	1.022 + j0.015	0	0
3	1.077 + j0.011	0	0
4	1.057 + j0.012	0	0
5	0.999 + j0.013	0	- 600.9
6	1.032 + j0.100	400	- 172.6
7	1.023 + j0.078	0	0
8	1.011 + j0.047	0	-2006.4
9	0.999 + j0.048	50	- 657
10	1.006 + j0.038	0	-1395.6
11	0.886 + j0.041	- 50	- 30
12	0.949 + j0.040	- 50	- 32
13	0.878 + j0.049	0	0

4

conclusions and suggestions for future work

A reformulation of the load flow problem as an optimization problem was presented. This was achieved by the fact that at solution, the sum of the power flows at each bus had to equal zero.

In the SOR method, the voltage at each busbar is expressed explicitly in terms of the voltages at other busbars. The iterative process is based on the voltage convergence. The Newton-Raphson method of load flow solution involves a direct solution of the system of power flow equations which are nonlinear and one equation is written for each bus. Hence, better power balance is obtained using the NR method than the SOR method. In the optimization approach, the load flow function is defined as the sum of squares of the mismatches, the minimum of which corresponds to the solution when the minimum is zero.

This thesis has also presented the application of the variable metric method without linear search to solve the load flow problem. The main emphasis of this work is placed on the investigation of the use of the minimization approach as a tool in load flow study. The following conclusions are drawn.

(1) The rapid final convergence characteristic of the minimization process process for the load flow function gives a solution with a total mismatch of the order of 10^{-15} with only a few additional iterations beyond that required by a solution with less accuracy. This means that high solution accuracy is obtained at almost no extra cost. Solution with this high degree of accuracy is essential in power loss studies which involve the subtraction of power flows that are almost equal and also in the optimum power scheduling problem which is concerned with the minimization of power losses which are small quantities. Though the SOR has given fairly satisfactory power mismatches for the systems tested, it is the experience of the Manitoba Hydro that power loss studies often cannot be done by SOR which is based on voltage convergence criteria.

(2) An encouraging feature of the nonlinear programming approach is that the process is nondivergent for systems of all kinds. The SOR method fails to cater for systems with negative transfer reactance branches. The process diverges for such cases. In load flow studies using the SOR method, the series capacitive reactance in the line is usually netted out by the inductive reactance. In those cases where there is a net capacitive reactance, methods such as the NR method and the nonlinear programming approach which converge for the negative

reactance case are nevertheless essential when information at any intermediate points along the line is required.

(3) Another characteristic of the nonlinear programming approach is that a second solution can be detected, whereas this is not shown by the normal iterative methods. A load flow study using the nonlinear programming approach is, therefore, beneficial at the system design stage to ensure that the system is not, perhaps, bistable. Usually the "extra" solution is far from the real one and so would not be permitted by system controls. But one cannot be sure that this need always be so. One day a system having two operating points that are close to each other could be constructed and found unable to be put into operation.

(4) Wallach [33] claims that the reformulation of the load flow problem as an optimization problem may reduce the computer time. From the analysis of the methods and the experimental data obtained, it is concluded that with the optimization methods available so far, a nonlinear programming approach to load flow requires greater computation time than normal iterative methods. Algorithm 2, the VMM without linear search, has better convergence than the algorithm with linear search. The number of function evaluations of the former is about 60% that of the latter. It has been reported by Tinney [30] that the computation time of the NR method with the implementation of the optimally ordered elimination scheme is less than that required by the best accelerated SOR method. Estimated from the data given, the computation time for a 30-bus system is approximately 3 seconds.

From Table 3.3 of chapter 3, the computation time required by SOR and Algorithm 2 of VMM are approximately 9 and 56 seconds respectively on an IBM 360/65 computer. In other words, the ratios of the computation time for the three methods, the NR, SOR and VMM are approximately 0.3 : 1 : 6. The figures given do not include input data time which is common to all methods. Estimated on the basis that the charge for one hour of cpu time is $450.00, the computation costs involved in the solution of the 30-bus system using NR, SOR and VMM are approximately $0.40, $1.20 and $7.00 respectively.

(5) As is indicated in (4), the optimization approach involves more computation. But the stability of the solution process is guaranteed. It is proposed that power system programs should use the NR method, which is the fastest, and switch to VMM if in trouble. In this way, the manhours wasted in trying to locate the cause of trouble, which may be due to the instability of the method or the nonexistence of a solution, could be reduced. Although the VMM requires higher computation costs, it is not expensive with respect to engineers time.

(6) Convergence characteristics of Fig. 3.4 indicates that a large proportion of the total number of iterations is spent on the flat region of the curve at the beginning of the solution process. It is suggested that further work should be done to study the behaviour of the load flow function as well as to develop new minimization algorithms capable of catering for this situation and giving a steep convergence throughout. The rapid convergence rate of the optimization approach may find its usefulness in on-line control in which small perturbations to a

solved system could be solved rapidly. Also, one can envisage the application of optimization methods in the solution of system transients by predictor-corrector methods.

(7) The author warns that a solution of a nonlinear problem, such as the load flow problem, may not have a solution for a specified schedule of loads - even with a slack bus included. Also, local nonzero minima may exist that may be discovered by the optimization or Newton-Raphson schemes. The prior will locate the nonzero minimum, whereas, the latter may oscillate about it. It is recommended that the total power mismatch be printed out so that, if it is nonzero, the user will not confuse such a point with a solution. Also, as pointed out earlier, multiple solutions may exist. In such cases, one should use a number of different starting points.

REFERENCES

[1] Adachi, N., "On variable metric algorithms", Journal of Optimization Theory and Applications", Vol. 7, p. 391, June 1971.

[2] Bandler, J.W., "Optimization methods for computer-aided design", IEEE Transaction, Vol. MTT-17, p. 533, August 1969.

[3] Box, M.J., "A comparison of several current optimization methods and the use of transformation in constrained problems", Computer Journal, Vol. 9, p. 67, January 1966.

[4] Broyden, C.G., "A class of methods for solving nonlinear simultaneous equations", Mathematics of Computation, Vol. 19, p. 577, 1965.

[5] Broyden, C.G.. "Quasi-Newton methods and their applications to function minimization", Mathematics of Computation, Vol. 21, p. 368, 1967.

[6] Broyden, C.G., "The convergence of a class of double-rank minimization algorithms, part 1, general considerations", J. Inst. Maths. Applications, Vol. 6, p. 76, March 1970.

[7] Broyden, C.G., "The convergence of a class of double-rank minimization algorithms, part 2, the new algorithm", J. Inst. Maths. Applications, Vol. 6, p. 222, September 1970.

[8] Davidon, W.C., "Variable-metric method for minimization", A.E.C. Research and Development Report, ANL-5990, 1959.

[9] Fletcher, R., "Function minimization without evaluating derivatives-a review", Computer Journal, Vol. 8, p. 33, 1965.

[10] Fletcher, R., "A new approach to variable-metric algorithms", Computer Journal, Vol. 13, p. 317, August 1970.

[11] Fletcher, R. and Powell, M.J.D., "A rapidly convergent descent method for minimization", Computer Journal, Vol. 6, p. 163, 1963.

[12] Fletcher, R. and Reeves, C.M., "Function minimization by conjugate gradients", Computer Journal, Vol. 7, p. 149, July 1964.

[13] Freris, L.L. and Sasson, A.M., "Investigation of the load-flow problem", Proc. IEE, Vol. 115, p. 1459, October 1968.

[14] Glimn, A.F. and Stagg, G.W., "Automatic calculation of load flows", AIEE Transaction, Vol. 76, pt. III, p. 817, October 1957.

[15] Hardaway, R.H., "An algorithm for finding a solution of simultaneous nonlinear equations", AFIP Conference Proceedings, Vol. 33, p. 105, 1968.

[16] Hildebrand, F.B., "Methods of applied mathematics", Chapter, 1, Prentice-Hall, 1965.

[17] Hooke, R. and Jeeves, T.A., "A direct search solution of numerical and statistical problems", J. ACM, Vol. 8, p. 212, April 1961.

[18] Laughton, M.A. and Humphrey Davies, M.W., "Numerical techniques in solution of power system load flow problems", Proc. IEE, Vol. 111, p. 1575, September 1964.

[19] Murtagh, B.A. and Sargent, R.W.H., "Computational experience with quadratically convergent minimization methods", Computer Journal, Vol. 13, p. 185, May 1970.

[20] Nelder, J.A. and Mead, R., "A simplex method for function minimization", Computer Journal, Vol. 7, p. 308, January 1965.

[21] Powell, M.J.D., "An efficient method of finding the minimum of a function of several variables without calculating derivatives", Computer Journal, Vol. 7, p. 155, July 1964.

[22] Powell, M.J.D., "A method for minimizing a sum of squares of nonlinear functions without calculating derivatives", Computer Journal, Vol. 7, p. 303, January 1965.

[23] Powell, M.J.D., "Minimization of functions of several variables", in Numerical Analysis: An Introduction, J. Walsh, Ed. Washington, D.C.: Thompson, 1967

[24] Powell, M.J.D., "On the convergence of the variable metric algorithm", J. Inst. Maths. Applications, Vol. 7, p. 21, February 1971.

[25] Rosenbrock, H.H., "An automatic method for finding the greatest or least value of a function", Computer Journal, Vol. 3, p. 175, October 1960.

[26] Sasson, A.M. and Jaimes, F.J., "Digital methods applied to power flow studies", IEEE Transactions, Vol. PAS-86, p. 860, July 1967.

[27] Sasson, A.M., "Nonlinear programming solutions for load-flow, minimum-loss, and economic dispatching problems", IEEE Transactions, Vol. PAS-88, p. 399, April 1969.

[28] Stagg, G.W. and El-Abiad, A.H., "Computer Methods in Power System Analysis", Chapter 8, McGraw-Hill, 1968.

[29] Stevenson, W.D., "Elements of Power System Analysis", Chapter 10, McGraw-Hill, 1962.

[30] Tinney, W.F., "Power flow solution by Newton's method", IEEE Transaction, Vol. PAS-86, p. 1449, November 1967.

[31] Treece, J.A., "Bootstrap gauss-seidel load flow", Proc. IEEE, Vol. 116, p. 866, May 1969.

[32] Van Ness, J.E. and Griffin, J.H., "Elimination methods for load flow studies", AIEE Transactions,Vol. 80, pt. III, p. 299, June 1961.

[33] Wallach, Y., "Gradient methods for load flow problems", IEEE Transaction, Vol. PAS-87, p. 1314, May 1968.

[34] Wallach, Y. and Even, R.K., "Application of Newton's method to load flow calculations", Proc. IEE, Vol. 114, p. 372, March 1967.

[35] Ward, J.B. and Hale, H.W., "Digital computer solution of power flow problems", AIEE Transactions,Vol. 75, p. 398, June 1956.

[36] Wilde, D.J. and Beightler, C.S., "Foundations of optimization", Prentice-Hall, 1967.

APPENDICES

A1 Derivation of Recurrence Formula for Successive Search Direction VMM Without Linear Search

From eqn. (2.48), the search direction at the (i + 1)th iterations is

$$\underline{p}_{i+1} = - H_{i+1}\, \underline{g}_{i+1} \qquad \text{(A1)}$$

Substituting eqn. (2.33) into eqn. (A1) gives

$$\underline{p}_{i+1} = - \left(H_i + \frac{\underline{\delta}_i\, \underline{\delta}_i^T}{\underline{\delta}_i^T\, \underline{y}_i} - \frac{H_i\, \underline{y}_i\, \underline{y}_i^T\, H_i}{\underline{y}_i^T\, H_i\, \underline{y}_i}\right) \underline{g}_{i+1}$$

$$= - H_i\, \underline{g}_{i+1} + \frac{H_i\, \underline{y}_i\, \underline{y}_i^T\, H_i}{\underline{y}_i^T\, H_i\, \underline{y}_i}\, \underline{g}_{i+1} \qquad \text{from eqns. (2.39) and (2.40)}$$

$$= - H_i\, \underline{g}_{i+1} + H_i\, \underline{y}_i + \frac{H_i\, \underline{y}_i\, \underline{y}_i^T\, H_i}{\underline{y}_i^T\, H_i\, \underline{y}_i}\, \underline{g}_i \qquad \text{from eqn. (2.20)}$$

$$= - H_i\, \underline{g}_i + \frac{H_i\, \underline{y}_i\, \underline{y}_i^T\, H_i\, \underline{g}_i}{\underline{y}_i^T\, H_i\, \underline{y}_i} \qquad \text{using eqn. (2.20) again}$$

$$= \left(I - \frac{H_i\, \underline{y}_i\, \underline{y}_i^T}{\underline{y}_i^T\, H_i\, \underline{y}_i}\right) \underline{p}_i \qquad \text{as } \underline{p}_i = - H_i\, \underline{g}_i$$

Q.E.D.

A2 Analysis and Comparison of the Amount of Computation Required by SOR and VMM

SOR

(I) Formulation:

$$V_i^{(m+1)} = V_i^{(m)} + \omega \left[\frac{\frac{P_i^*}{V_i^{*(m)}} - \sum_{k=1}^{i-1} Y_{ik} V_k^{(m+1)} - \sum_{k=i+1}^{n} Y_{ik} V_k^{(m)}}{Y_{ii}} - V_i^{(m)} \right]$$

$$i = 2, 3, \ldots, n$$

where bracketted superscript denotes iteration count. All quantities are complex.

(II) Estimated number of multiplications per iteration, M_s

$$M_s = 4(n+2)^2 - 16$$

VMM

(I) Formulation

(1) function evaluation

$$F = \sum_{i=1}^{2(n-1)} |r_i|^P$$

where

$$r_i = PS_i - \left[e_i \sum_{k=0}^{n} (g_{ik}e_k + b_{ik}f_k) + f_i \sum_{k=0}^{n} (g_{ik}f_k - b_{ik}e_k) \right]$$

$$r_{i+n} = QS_i - \left[f_i \sum_{k=0}^{n} (g_{ik}e_k + b_{ik}f_k) - e_i \sum_{k=0}^{n} (g_{ik}f_k - b_{ik}e_k) \right]$$

$$1 \leq i \leq n$$

(2) gradient calculation

$$g_i = \frac{\partial F}{\partial e_i} = 2 \sum_{k=1}^{2n} \left(r_k \frac{\partial r_k}{\partial e_i}\right)$$

(3) voltage correction

$$\underline{e}^{(m+1)} = \underline{e}^{(m)} + \alpha \, \Delta \underline{e}^{(m)}$$

where $\Delta \underline{e}^{(m)} = - H^{(m)} \, \underline{g}^{(m)}$

$$\alpha^{(m)} = \frac{2(F_{min} - F(e^{(m)}))}{\Delta \underline{e}^{(m)T} \, \underline{g}}$$

(4) updating H

$$H = H + \frac{\alpha \Delta \underline{e} \alpha \Delta \underline{e}^T}{\alpha \Delta \underline{e}^T \, \underline{y}} - \frac{H \, \underline{y} \, \underline{y}^T \, H}{\underline{y}^T \, H \, \underline{y}}$$

where superscripts are omitted for clarity.

(II) Estimated Number of Multiplications Per Iteration, M_V

(1)	function evaluation	$8n^2 + 12n$
(2)	gradient calculation	$12n^2$
(3)	correction	$4n^2 + 4n$
(4)	updating H	$12n^2 + 8n$

Therefore, $M_V = 36(n + \frac{1}{3})^2 - 4$.

A3 Program Description and Listings

The program consists of three parts:

(1) the main program which reads in all the data required. Input data consists of initial voltage values, specified terminal conditions of the network, admittance data and all appropriate parameters required by the minimization subroutines.

(2) subroutine FUNCT which computes the load flow function to be minimized and the gradient vector. It is of the form FUNCT (NN, EN, U, GRAD). For each NN-dimensional argument EN, the function value and gradient vector are computed and on return, stored in U and GRAD, respectively.

(3) a minimization subroutine which performs the calculation of an unconstrained minimum of a function of several variables. Subroutine FMFP locates the minimum of a function using variable metric method with linear search, (Algorithm 1 of Chapter 2). Subroutine VMM01 uses variable metric method without linear search (Algorithm 2 of Chapter 2).

```
FORTRAN IV G LEVEL  20.1                FUNCT              DATE = 72109        22/43/41

0001                  SUBROUTINE FUNCT(NN,EN,U,GRAD)
                C     LOAD FLOW FUNCTION SUBROUTINE WHICH EVALUATES THE FUNCTION VALUE
                C     AND GRADIENT VECTOR
                C
                C     VARIABLE DESCRIPTIONS:
                C     N  -- NUMBER OF BUSBARS
                C     NN -- NUMBER OF VARIABLES OF THE FUNCTION BEING MINIMIZED
                C     E  -- VECTOR OF DIMENSION (NE=2N) WHOSE ELEMENTS ARE THE REAL AND
                C           IMAGINARY PARTS OF THE BUSBAR VOLTAGES
                C     EN -- VECTOR OF DIMENSION NN WHOSE ELEMENTS ARE THE REAL AND
                C           IMAGINARY PARTS OF THE BUSBAR VOLTAGES EXCEPT THE SLACK BUS
                C     G  -- A NXN MATRIX WHOSE ELEMENTS ARE THE REAL COMPONENTS OF THE
                C           ADMITTANCE MATRIX
                C     B  -- A NXN MATRIX WHOSE ELEMENTS ARE THE IMAGINARY COMPONENTS OF
                C           THE ADMITTANCE MATRIX
                C     PS -- A N-ORDER VECTOR OF REAL COMPONENT OF POWER AT EACH BUS
                C     QS -- A N-ORDER VECTOR IMAGINARY COMPONENT OF POWER AT EACH BUS
                C     U  -- FUNCTION VALUE
                C     GRAD -- GRADIENT VECTOR OF FUNCTION U
                C
                C     DOUBLE PRECISION E(NE),EN(NN),GRAD(NN),TANG(NE),F(NE),PS(N),QS(N),
                C    1G(N,N),B(N,N),A(N),D(N),P,U,DABS,DSQRT,VM(N)
0002                  DOUBLE PRECISION EN(24),PS(13),QS(13),TANG(26),G(13,13),B(13,13),
                     1A(13),GRAD(24),F(26),E(26),D(13),P,J,DABS,DSQRT,VM(13)
0003                  COMMON G,B,PS,QS,E,VM, P,KOUNT,N
0004                  DO 6 I=2,N
0005                  E(I)=EN(I-1)
0006                6 E(I+N)=EN(I+N-2)
0007                  IF(KOUNT.GT.0) P=2.00
0008                  DO 1 I=2,N
0009                  A(I)=0.00
0010                  D(I)=0.00
0011                  DO 1 J=1,N
0012                  K=J+N
0013                  A(I)=A(I)+E(J)*G(J,I)+E(K)*B(J,I)
0014                1 D(I)=D(I)+E(K)*G(J,I)-E(J)*B(J,I)
0015                  U=0.
0016                  DO 8 I=2,N
0017                  K=I+N
0018                  F(I)=PS(I)-(E(I)*A(I)+E(K)*D(I))
                C     GENERATOR BUS -- VOLTAGE MAGNITUDE SPECIFIED
0019                  IF(I.EQ.5.OR.I.EQ.6) GO TO 3
0020                  IF(I.EQ.8.OR.I.EQ.9) GO TO 3
0021                  IF(I.EQ.10) GO TO 3
                C     LOAD BUS -- REACTIVE POWER SPECIFIED
0022                  F(K)=QS(I)-(E(K)*A(I)-E(I)*D(I))
0023                  GO TO 8
0024                3 F(K)=VM(I)-DSQRT(E(I)**2+E(K)**2)
0025                8 U=U+DABS(F(I))**P+DABS(F(K))**P
                C     GRADIENT CALCULATION
0026                  DO 2 I=2,N
0027                  K=I+N
0028                  IF(I.EQ.5.OR.I.EQ.6) GO TO 7
0029                  IF(I.EQ.8.OR.I.EQ.9) GO TO 7
0030                  IF(I.EQ.10) GO TO 7
0031                  TANG(I)=(DABS(F(I))**(P-2)*F(I)*(A(I)+E(I)*G(I,I)-E(K)*B(I,I))+
                     1DABS(F(K))**(P-2)*F(K)*(G(I,I)*E(K)-D(I)+B(I,I)*E(I)))*(-P)
0032                  TANG(K)=(DABS(F(I))**(P-2)*F(I)*(B(I,I)*E(I)+D(I)+G(I,I)*E(K))+
                     1DABS(F(K))**(P-2)*F(K)*(A(I)+E(K)*G(I,I)-B(I,I)*E(I)))*(-P)
0033                  GO TO 9
0034                7 TANG(I)=(DABS(F(I))**(P-2)*F(I)*(A(I)+E(I)*G(I,I)-E(K)*B(I,I))
                     1+DABS(F(K))**(P-2)*F(K)*E(I)/DSQRT(E(I)**2+E(K)**2))*(-P)
0035                  TANG(K)=(DABS(F(I))**(P-2)*F(I)*(B(I,I)*E(I)+D(I)+G(I,I)*F(K))
                     1+DABS(F(K))**(P-2)*F(K)*E(K)/DSQRT(E(I)**2+E(K)**2))*(-P)
0036                9 DO 2 M=2,N
0037                  IF(M.EQ.I) GO TO 2
0038                  L=M+N
0039                  IF(M.EQ.5.OR.M.EQ.6) GO TO 10
0040                  IF(M.EQ.8.OR.M.EQ.9) GO TO 10
0041                  IF(M.EQ.10) GO TO 10
0042                  TANG(I)=TANG(I)+(DABS(F(M))**(P-2)*F(M)*(G(M,I)*E(M)-B(M,I)*E(L))+
                     1DABS(F(L))**(P-2)*F(L)*(G(M,I)*E(L)+B(M,I)*E(M)))*(-P)
0043                  TANG(K)=TANG(K)+(DABS(F(M))**(P-2)*F(M)*(B(M,I)*E(M)+G(M,I)*E(L))+
                     1DABS(F(L))**(P-2)*F(L)*(B(M,I)*E(L)-G(M,I)*E(M)))*(-P)
0044                  GO TO 2
0045               10 TANG(I)=TANG(I)+(DABS(F(M))**(P-2)*F(M)*(G(M,I)*E(M)-B(M,I)*E(L)))
                     1*(-P)
0046                  TANG(K)=TANG(K)+(DABS(F(M))**(P-2)*F(M)*(B(M,I)*E(M)+G(M,I)*E(L)))
                     1*(-P)
0047                2 CONTINUE
0048                  DO 4 I=1,NN
0049                  IF(I.GE.N) GO TO 5
0050                  GRAD(I)=TANG(I+1)
0051                  EN(I)=E(I+1)
0052                  GO TO 4
0053                5 GRAD(I)=TANG(I+2)
0054                  EN(I)=E(I+2)
0055                4 CONTINUE
0056                  RETURN
0057                  END
```

```
      SUBROUTINE VMM01(N,X,F,G,H,UNITH,FEST,EPS,MAXFN,IPRINT,IEXIT)
C     FUNCTION MINIMIZATION SUBROUTINE OF ALGORITHM 2
C
C     VARIABLE DESCRIPTIONS:
C     N      -- NUMBER OF VARIABLES
C     X(N)   -- CURRENT APPROXIMATION TO VARIABLES - AN INITIAL
C               APPROXIMATION SHOULD BE PROVIDED ON ENTRY, WHICH WILL
C               BE REPLACED BY THE BEST ESTIMATE OBTAINED ON EXIT
C     F      -- ESTIMATE OF MINIMUM VALUE OF FUNCTION IS LEFT ON EXIT
C     G(N)   -- ESTIMATE OF GRADIENT AT MINIMUM IS LEFT ON EXIT
C     UNITH  -- (LOGICAL) IF .TRUE. THEN VMM01 SETS H TO A UNIT MATRIX.
C     FEST   -- A LOWER BOUND ON THE VALUE OF F
C     EPS(N) -- ACCURACY REQUIRED IN EACH ELEMENT OF X
C     MAXFN  -- MAXIMUM NUMBER OF TIMES FUNCT CAN BE CALLED
C     IPRINT -- PRINTING WILL TAKE PLACE ON ENTRY, AT EVERY IPRINT
C               ITERATIONS, AND ON EXIT, GIVING <ITERATION NO., NO. OF
C               CALLS OF FUNCT, CURRENT F,X,G> IN THAT ORDER.
C               NO PRINTING AT ALL IF IPRINT=0
C     IEXIT  -- 1  NORMAL EXIT
C               2  G DX  0 NOT POSSIBLE WITHOUT ROUNDING ERROR
C               3  FUNCT CALLED MAXFN TIMES
C               4  AN ULTRA-QUADRATIC VARIATION IN F DETECTED AFTER THE
C                  NTH ITERATION
C               5 FUNCTION VALUE LESS THAN FEST DETECTED
C     H(N*(N+7)/2)  -- WORKING SPACE VECTOR
C
      DOUBLE PRECISION X(24),G(24),H(372),EPS(24),F,STEP,GDX,Z,W,FEST,
     1FP,GPDX,DGDX,DSQRT,DGHDG,DABS
      LOGICAL CONV,UNITH
      IF (IPRINT.NE.0) PRINT 102
  102 FORMAT(2X,'ENTRY TO VMM01')
      ITN=0
      KK=0
      CALL FUNCT(N,X,F,G)
      IF(F.LT.FEST) GO TO 98
      NFNS=1
      STEP=1.D0
      IDX=N
      IDG=N+N
      IH=IDG+N
      IF( .NOT.UNITH) GO TO 1
  104 IJ=IH+1
      DO 2 I=1,N
      DO 2 J=I,N
      H(IJ)=0.D0
      IF(I.EQ.J) H(IJ)=1.D0
    2 IJ=IJ+1
    1 ITN=ITN+1
      CONV=.TRUE.
      GDX=0.D0
      DO 3 I=1,N
      Z=0.D0
      IJ=IH+I
      IF(I.EQ.1) GO TO 21
      II=I-1
      DO 4 J=1,II
      Z=Z-H(IJ)*G(J)
    4 IJ=IJ+N-J
   21 DO 5 J=I,N
      Z=Z-H(IJ)*G(J)
    5 IJ=IJ+1
      IF(DABS(Z).GT.EPS(I)) CONV=.FALSE.
      H(IDX+I)=Z
    3 GDX=GDX+G(I)*Z
      IF(IPRINT.EQ.0) GO TO 6
      IF(MOD(ITN,IPRINT).NE.0) GO TO 6
    6 IEXIT=1
      IF(CONV) GO TO 99
      IEXIT=2
      IF(GDX.GE.0.D0) GO TO 99
      Z=1.D0
      IF(ITN.LT.N.AND.UNITH) Z=STEP
      W=2.D0*(FEST-F)/GDX
      IF(W.LT.Z) Z=W
      STEP=Z
   13 GDX=GDX*Z
      DO 7 I=1,N
      H(IDX+I)=H(IDX+I)*Z
    7 X(I)=X(I)+H(IDX+I)
      CALL FUNCT(N,X,FP,H)
      IF(FP.LT.FEST) GO TO 98
      NFNS=NFNS+1
      IEXIT=3
      IF(NFNS.EQ.MAXFN) GO TO 99
      GPDX=0.D0
      DO 8 I=1,N
      H(IDG+I)=H(I)-G(I)
    8 GPDX=GPDX+H(I)*H(IDX+I)
      DGDX=GPDX-GDX
      IF(F.GT.FP-0.1D-3*GDX) GO TO 9
      IEXIT=4
      IF(GPDX.LT.0.D0.AND.ITN.GT.N) GO TO 99
      Z=3.D0*(F-FP)+GPDX+GDX
      W=DSQRT(1.D0-GDX/Z*GPDX/Z)*DABS(Z)
      Z=1.D0-(GPDX+W-Z)/(DGDX+2.D0*W)
      IF(Z.LT.0.1D0) Z=0.1D0
      DO 10 I=1,N
   10 X(I)=X(I)-H(IDX+I)
      GO TO 11
    9 F=FP
      DO 12 I=1,N
   12 G(I)=H(I)
      IF(DGDX.GT.0.D0) GO TO 14
      GDX=GPDX
      Z=4.D0
   11 STEP=Z*STEP
```

```
C081              GO TO 13
0082           14 IF(GPDX.LT.0.500*GDX) STEP=2.00*STEP
0083          107 DGMDG=0.00
0084              DO 15 I=1,N
0085              Z=0.DO
0086              IJ=IM+I
0087              IF(I.EQ.1) GO TO22
0088              II=I-1
0089              DO 16 J=1,II
0090              Z=Z+H(IJ)*H(IDG+J)
0091           16 IJ=IJ+N-J
0092           22 DO 17 J=I,N
0093              Z=Z+H(IJ)*H(IDG+J)
0094           17 IJ=IJ+1
0095              DGMDG=DGMDG+Z*H(IDG+I)
0096           15 H(I)=Z
0097              IF(DGMDG.LT.0.D0) DGMDG=DGDX*0.1D-1
0098              IF(DGDX.LT.DGMDG) GO TO18
0099              W=1.DO+DGMDG/DGDX
0100              DO 19 I=1,N
0101           19 H(IDX+I)=W*H(IDX+I)-H(I)
0102              DGDX=DGDX+DGMDG
0103              DGMDG=DGDX
0104           18 IJ=IM
0105              DO 20 I=1,N
0106              W=H(IDX+I)/DGDX
0107              Z=H(I)/DGMDG
0108              DO 20 J=I,N
0109              IJ=IJ+1
0110           20 H(IJ)=H(IJ)+W*H(IDX+J)-Z*H(J)
0111              WRITE(6,100) ITN,NFNS,F
0112          100 FORMAT(' ITN=',2I5,4X,'F=',D14.6,4X,'DLOG F =',D14.6)
0113              GO TO 1
0114           98 IEXIT=5
0115           99 IF(IPRINT.EQ.0) RETURN
0116              RETURN
0117              END
```