

LOAD FLOW SOLUTION BY APPLYING HYBRID ALGORITHM TO  
THE NEWTON-RAPHSON METHOD

by

Chaipant Tappayuthpijarn

Submitted in Partial Fulfillment of the Requirements

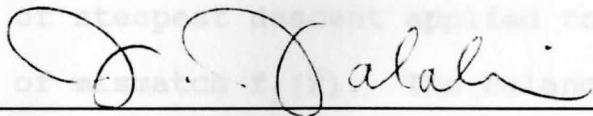
for the Degree of

Masters of Science

in the

Electrical Engineering

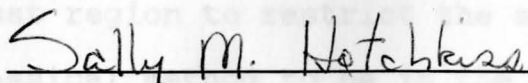
Program



06/06/90

Advisor

Date



June 8, 1990

Dean of the Graduate School

Date

YOUNGSTOWN STATE UNIVERSITY

June, 1990

**ABSTRACT**

**LOAD FLOW SOLUTION BY APPLYING HYBRID ALGORITHM TO  
THE NEWTON-RAPHSON METHOD**

Chaipant Tappayuthpijarn

Masters of Science in Engineering

Youngstown State University, 1990

The purpose of the HYBRID algorithm, discussed in this thesis, is to improve the efficiency of the convergence of the existing NEWTON-RAPHSON method in solving the system of nonlinear power flow equations, when its close initial estimates are not available. The algorithm is based on the interpolation between the fast convergence standard NEWTON-RAPHSON iteration and the method of steepest descent applied to the sum of the square of mismatch  $f_i(\bar{x})$ . The balance between these two methods is governed by introducing the concept of the trust region to restrict the step predicted by the classical method to be in the quadratic region and to switch to the steepest decent method that is better when the initial values are far from the solution.

Digital computer results and their comparisons of the 10 bus test system, with different initial values, by the proposed algorithm and by the standard method are also discussed in this thesis.

## ACKNOWLEDGMENTS

First I wish to thank my thesis advisor, Dr. J. Jalali, for having given me the opportunity to work on this thesis project and for his constant support, patience and the numberless guidance hours.

I thank Dr. Salvatore R. Pansino, Chairman, and Dr. Matthew Siman for reviewing this thesis and being on my committee. I must also acknowledge the assistance of Mrs. Anna Mae Serrecchio who is always willing to help students without hesitation.

Finally, I dedicate this thesis to my parents, Dr. Tam and Janjarus Tappayuthpijarn, whose love, support and encouragement enabled me to complete my studies at Youngstown State University.

## TABLE OF CONTENTS

	PAGE
ABSTRACT. . . . .	ii
ACKNOWLEDGMENTS . . . . .	iii
TABLE OF CONTENTS . . . . .	iv
LIST OF SYMBOLS . . . . .	vi
LIST OF FIGURES . . . . .	viii
LIST OF TABLES . . . . .	x
CHAPTER	
I. INTRODUCTION . . . . .	1
1.1 Background and Objective . . . . .	1
1.2 Overview. . . . .	3
II. LOAD FLOW STUDIES. . . . .	4
2.1 Introduction and background . . . . .	4
2.2 Data for Load Flow Studies. . . . .	6
2.3 Load Flow calculation by the Newton-Raphson method . . . . .	11
2.4 Information obtained in Load Flow Studies . . . . .	19
III. ERROR ANALYSIS IN NUMERICAL COMPUTATION.	
3.1 Introduction. . . . .	22
3.2 Sources of error in numerical computation . . . . .	22
3.3 Convergence analysis for the Newton-Raphson method . . . . .	24
VI. HYBRID ALGORITHM . . . . .	26
4.1 Introduction. . . . .	26
4.2 Background. . . . .	26

4.3	The method of Steepest Descent. . .	29
4.4	Trust region and Switching policies	34
4.5	Hybrid algorithm for Newton-Raphson Load Flow . . . . .	47
4.6	Computer subroutines. . . . .	50
V.	EXAMPLE AND NUMERICAL RESULTS	
5.1	Introduction. . . . .	57
5.2	Example : 10 bus test system. . . .	57
5.3	Results . . . . .	57
IV.	CONCLUSION . . . . .	70
6.1	Summary . . . . .	70
6.2	Recommendations for Future Work . .	71
APPENDIX A.	BASIC Program For The HYBRID Algorithm Applied To The Standard NEWTON-RAPHSON LOADFLOW. . . . .	72
APPENDIX B.	System Data For 10 Bus System. .	95
APPENDIX C.	Computer Results For 10 Bus System . . . . .	96
REFERENCES	. . . . .	99

$\tau$	transpose of matrix (superscript)
$\  \cdot \ $	euclidean norm
$ \cdot $	absolute value
$P_i$	$i$ th bus real power mismatch
$Q_i$	$i$ th bus reactive power mismatch
$P_i(\text{cal})$	$i$ th bus calculated real power
$Q_i(\text{cal})$	$i$ th bus calculated reactive power
$P_i(\text{spec})$	$i$ th bus scheduled real power
$Q_i(\text{spec})$	$i$ th bus scheduled reactive power

## LIST OF SYMBOLS

SYMBOL	DEFINITION
R	trust radius
k	iteration count (superscript)
$\bar{f}(\bar{x})$	a nonlinear vector function
$F(\bar{x})$	the sum of the square of mismatch $f(\bar{x})$
$\bar{g}(\bar{x})$	vector of gradient of $F(\bar{x})$
$\bar{x}$	vector of unknown variables
$\bar{x}^0$	vector of initial values
$\bar{x}^k$	vector of predicted values at iteration k
[ $\bar{J}$ ]	Jacobian matrix
[ $\bar{H}$ ]	Hessian matrix
$\bar{\delta}$	vector of NEWTON-RAPHSON step
$\bar{dx}$	vector of predicted step
$u^*$	positive scalar of steepest descent step predicted at solutions
T	transpose of matrix (superscript)
.	euclidean norm
.	absolute value
$dP_i$	$i^{\text{th}}$ bus real power mismatch
$dQ_i$	$i^{\text{th}}$ bus reactive power mismatch
$P_i(\text{cal})$	$i^{\text{th}}$ bus calculated real power
$Q_i(\text{cal})$	$i^{\text{th}}$ bus calculated reactive power
$P_i(\text{spec})$	$i^{\text{th}}$ bus scheduled real power
$Q_i(\text{spec})$	$i^{\text{th}}$ bus scheduled reactive power

## SYMBOL

## DEFINITION

SYMBOL	DEFINITION	PAGE
$ V_i ,  V_j $	voltage magnitude at bus i and j	9
$\theta_i, \theta_j$	voltage phase angle at bus i and j	9
$Y_{ij}$ <span style="border: 1px solid black; padding: 2px;"><math>\beta_{ij}</math></span>	element of bus admittance matrix in polar form	20
4.1	A straight line in 2-space illustrating vector search direction $S^k$ and scalar	30
4.2	Line searches on a quadratic function	34
4.3	The view of quadratic function with 2 variables when the trust region with radius R is applied	34
4.4	The view of the switching policies when the NEWTON point (NP) is inside the trust region	37
4.5	The view of the switching policies when the NEWTON point (NP) and the CAUCHY point (CP) are both outside the trust region	39
4.6	The view of the switching policies when the NEWTON point (NP) is outside the trust region but the CAUCHY point (CP) is	40
4.7	Computer flow diagram of the HYBRID algorithm applied to the NEWTON-RAPHSON Load Flow	43
5.1	One-line diagram for 10 bus tested system	47
5.2	The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when $ V^0  = 1.0$ p.u. and $\theta^0 = 0.0$ rad.	51
5.3	The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when $ V^0  = 1.5$ p.u. and $\theta^0 = 0.5$ rad.	54
5.4	The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when $ V^0  = 1.5$ p.u. and $\theta^0 = 1.0$ rad.	61

## LIST OF FIGURES

FIGURE		PAGE
2.1	One-Line Connection Diagram . . . . .	9
2.2	Computer flow diagram of the NEWTON-RAPHSON method applied to Load Flow solutions . .	20
4.1	A straight line in 2-space illustrating vector search direction $S^k$ and scalar $U^k$ .	30
4.2	Line searches on a quadratic function . .	34
4.3	The view of quadratic function with 2 variables when the trust region with radius $R$ is applied . . . . .	36
4.4	The view of the switching policies when the NEWTON point (NP) is inside the trust region . . . . .	37
4.5	The view of the switching policies when the NEWTON point (NP) and the CAUCHY point (CP) are both outside the trust region. .	39
4.6	The view of the switching policies when the NEWTON point (NP) is outside the trust region but the CAUCHY point (CP) is inside	40
4.7	Computer flow diagram of the HYBRID algorithm applied to the NEWTON-RAPHSON Load Flow . . . . .	51
5.1	One-line diagram for 10 bus tested system	57
5.2	The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when $ v^0 =1.0$ p.u and $\theta^0=0.0$ rad. . . . .	61
5.3	The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when $ v^0 =0.8$ p.u and $\theta^0=-0.5$ rad. . . . .	62
5.4	The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when $ v^0 =1.3$ p.u and $\theta^0=-1.0$ rad. . . . .	63



## LIST OF TABLES

FIGURE		PAGE
5.5	The characteristic of quadratic factor $r$ for different values of $R^0$ when $ V^0  = 1.0$ p.u and $\theta^0 = 0.0$ rad. . . . .	64
5.6	The characteristic of quadratic factor $r$ for different values of $R^0$ when $ V^0  = 0.8$ p.u and $\theta^0 = -0.5$ rad. . . . .	65
5.7	The characteristic of quadratic factor $r$ for different values of $R^0$ when $ V^0  = 1.3$ p.u and $\theta^0 = -1.0$ rad. . . . .	66
5.1	Switching status for the system when $ V^0  = 1.0$ p.u and $\theta^0 = 0.0$ rad. . . . .	67
5.2	Switching status for the system when $ V^0  = 0.8$ p.u and $\theta^0 = -0.5$ rad. . . . .	68
5.3	Switching status for the system when $ V^0  = 1.3$ p.u and $\theta^0 = -1.0$ rad. . . . .	69
B-1	Line admittance data for the 10 bus tested system. . . . .	70
B-2	Operating condition for the 10 bus tested system. . . . .	71
C-1	The solution of all unknowns for the 10 bus tested system. . . . .	72
C-2	The list of line flows and line losses for the 10 bus tested system. . . . .	73

## LIST OF TABLES

TABLE	DESCRIPTION	PAGE
2.1	Bus classification in Load Flow Studies. . . . .	7
3.1	Machine Precision . . . . .	23
4.1	The advantageous and disadvantageous comparisons between the NEWTON-RAPHSON method and the method of steepest descent . . . . .	28
4.2	List of computer subroutines for the HYBRID algorithm applied to the NEWTON-RAPHSON load flow . . . . .	55
5.1	Switching status for the HYBRID algorithm when $ V^0 =1.0$ p.u and $\theta^0=0.0$ rad. . . . .	67
5.2	Switching status for the HYBRID algorithm when $ V^0 =0.8$ p.u and $\theta^0=-0.5$ rad. . . . .	68
5.3	Switching status for the HYBRID algorithm when $ V^0 =1.3$ p.u and $\theta^0=-1.0$ rad. . . . .	69
B-1	Line admittance data for the 10 bus tested system. . . . .	95
B-2	Operating condition for the 10 bus tested system . . . . .	95
C-1	The solution of all unknowns for the 10 bus tested system. . . . .	97
C-2	The list of line flows and line losses for the 10 bus tested system . . . . .	98

## Chapter I

### INTRODUCTION

#### 1.1 Background and Objective

The NEWTON-RAPHSON method is the most widely used approach for the nonlinear power flow solution in power system planning largely because of its quadratic convergence characteristics. This fast convergence property yields the solutions of the nonlinear system converged in just a few iterations. However, the possible range of the initial values by the standard Newton-Raphson is required to be close to the roots of the system. An increase of error from neglected higher order terms in the indefinite Taylor's series can cause system divergence when the initial values are far from the roots. The convergence analysis of the standard NEWTON-RAPHSON method given in this thesis shows that the error on the current iteration is the function of the square of the error on a previous iteration. Therefore, the defined error could make the classical method unreliable on any iteration.

In order to overcome these problems, the method of steepest descent is introduced. This method has an important advantage over the Newton-Raphson method because it is not as sensitive to the initial values. The method of steepest descent is generally used in nonlinear optimization problems. It can be expediently used for

making solutions more accurate in cases when the Newton-Raphson method diverges when the initial values are far from its roots. Despite the advantageous property of this method, a large quantity of calculations is required to get the solutions of the system converged. This leads to the slow convergence of the method of steepest descent.

To obtain the useful properties of both the Newton-Raphson method and the method of steepest descent, the HYBRID algorithm has been developed. The purpose is to improve the convergence of the existing standard Newton-Raphson iteration when its initial estimates are far from the roots. The idea is to start with the steepest descent iteration when necessary, then switch to the standard Newton-Raphson method for fast convergence when the predicted values are close to the roots. The compromise between these two methods is governed by the concept of the trust region, defined in terms of the trust radius, and the switching policies. The purpose of the trust region is to restrict the step predicted by the standard Newton-Raphson method to be in the region where a quadratic is available. By the switching policies, if the predicted step is inside the "trust radius," then the correction is the full Newton-Raphson step; otherwise, the classical step is biased toward the steepest-descent direction. Moreover, the automatic revision of the trust radius is also provided by the proposed algorithm to

adjust the appropriate quadratic region for the next iteration at the end of every iteration.

## 1.2 Overview

In this thesis, a review of Load Flow Studies, along with a description of Load Flow calculation by the Newton-Raphson method, is contained in Chapter II. Chapter III discusses some sources of errors that can arise in numerical computation. It also includes a discussion of the effect of these errors on the convergence of the standard Newton-Raphson method in solving the system of nonlinear algebraic equations. In Chapter IV, the Hybrid algorithm is introduced, including a description of the method of steepest descent, the concept of the trust region and the switching policies between the Newton-Raphson and steepest descent iteration. At the end of the chapter, the application of the Hybrid algorithm to the standard Newton-Raphson Loadflow is provided. In Chapter V, numerical results and comparisons of the 10 bus test system, with different initial values, using the proposed algorithm and the standard Newton-Raphson method, are discussed. Finally, conclusions and recommendations for future research are provided in Chapter VI.

## Chapter II

### LOAD FLOW STUDIES

#### 2.1 INTRODUCTION AND BACKGROUND

A Load Flow Study [10,11,12,13] is the determination of the voltage, current, power factor, real power and reactive power at various points in an electrical network under existing or contemplated conditions of normal operation. It is essential in planning the future development of the system because satisfactory operation of the system depends on knowing the effects of interconnecting with other power systems, new loads, new generating stations and new transmission lines before they are installed. The mathematical formulation of the loadflow problem results in a system of nonlinear algebraic equations. These equations can be established by using either the bus or the loop frame of reference. The coefficients of the equations depend on the selection of the independent variables, i.e, voltages or currents. Thus, either the admittance or impedance matrices can be used.

Most of the early successful digital methods were based on the Y-matrix of the Gauss-Seidel method [13]. This requires minimum computer storage and uses only a small number of iterations for a small network. Unfortunately, as the size of the network is increased,

the number of iterations required increases dramatically for large systems. In some cases, the method does not provide a solution at all. These difficulties encountered in load-flow studies led to the development of the Newton-Raphson method [10,13]. The method is based on the Newton-Raphson algorithm designed to solve the simultaneous quadratic equations of the power network. Contrary to the Gauss-Seidel algorithm, it needs a larger time per iteration, but it can get the solutions in only a few iterations independent of the network size. Therefore, most of the load-flow problems that could not be solved by the Gauss-Seidel method are solved with no difficulty by this method

However, the recent research efforts have been concentrated on the development of the decouple Newton-Raphson method [10] since system planning studies and system operations may require a multiple-case load flow solution in some situations. These methods are based on the fact that in any power transmission network operating in the steady state, the coupling between  $P-|V|$  and  $Q-\theta$  is relatively weak, contrary to the strong coupling between  $P-\theta$  and between  $Q-|V|$ . Therefore, these methods solve the load-flow problem by decoupling the  $P-\theta$  and  $Q-|V|$  problem. Thus, the solutions are obtained by applying approximations to the Newton-Raphson method.

## 2.2 DATA FOR LOAD FLOW STUDIES

The load-flow problem can be defined as the calculation of the real and reactive powers flowing in each transmission line, and the magnitude and phase angle of the voltage of each bus of a given power system network for specified generation and load conditions. The information obtained from the load-flow studies can be used to test the systems capability to transfer energy from generation to load without overloading the line and to determine the adequacy of voltage regulation by shunt capacitors, shunt reactors, tap-changing transformer and the var-supplying capability of generators [12].

### 2.2.1 TYPE OF BUSES IN LOAD-FLOW STUDIES

In general, there are three types of buses in the load-flow problem :

- a. slack (generator) bus
- b. voltage-controlled (generator) buses or P-V buses
- c. load buses or P-Q buses

Since the transmission losses in a given system are associated with the bus profile, until a solution is obtained, the total power generation requirement of a system cannot be determined. Therefore, the generator at the slack bus is used to supply the additional real and reactive power necessary owing to the transmission losses



[12]. Thus, at the slack bus, the magnitude and phase angle of the voltage are known values, and the real and reactive power generated are the quantities to be determined. In order to define the load-flow problem to be solved, it is necessary to specify the real power and the voltage magnitude at each generator bus. This is because these quantities are controllable through excitation controls [12]. Since an overexcited synchronous generator supplies current at a lagging power factor, the reactive power of a generator is not required to be specified. The load buses are also known as the P-Q buses. This is due to the fact that the real and reactive power are specified at given load buses. Table 2.1 gives the bus types in load flow studies with corresponding known and unknown variables.

BUS TYPE	KNOWN QUAN.	UNKNOWN QUAN.
SLACK	$ V , \theta=0$	P, Q
GENERATOR (P-V)	P, $ V $	Q, $\theta$
LOAD (P-Q)	P, Q	$ V , \theta$

TABLE 2.1 Bus classification in Load Flow Studies

## 2.2.2 POWER SYSTEM EQUATIONS

### a. Network performance equations

In load-flow studies in normal operation, the basic assumption is that the given power system is a balanced three-phase system operating in its steady state. Therefore, the system can be represented by a one-line diagram of its single phase positive-sequence network [14], and the load-flow problem can be solved either by using the bus admittance matrix ( $Y_{bus}$ ) or the bus impedance matrix ( $Z_{bus}$ ) representation of the given network. By using the nodal analysis approach, the network equation in bus admittance form is

$$\bar{I}_{bus} = Y_{bus}\bar{E}_{bus} \quad (2.2.1)$$

or in bus impedance form is

$$\bar{E}_{bus} = Z_{bus}\bar{I}_{bus} \quad (2.2.2)$$

### b. Bus equations

Each bus of a network has four variable quantities associated with it: the real and reactive power, the voltage magnitude and the voltage phase angle. Any two of the four will be the independent variables and are specified, whereas the other two remain to be determined.

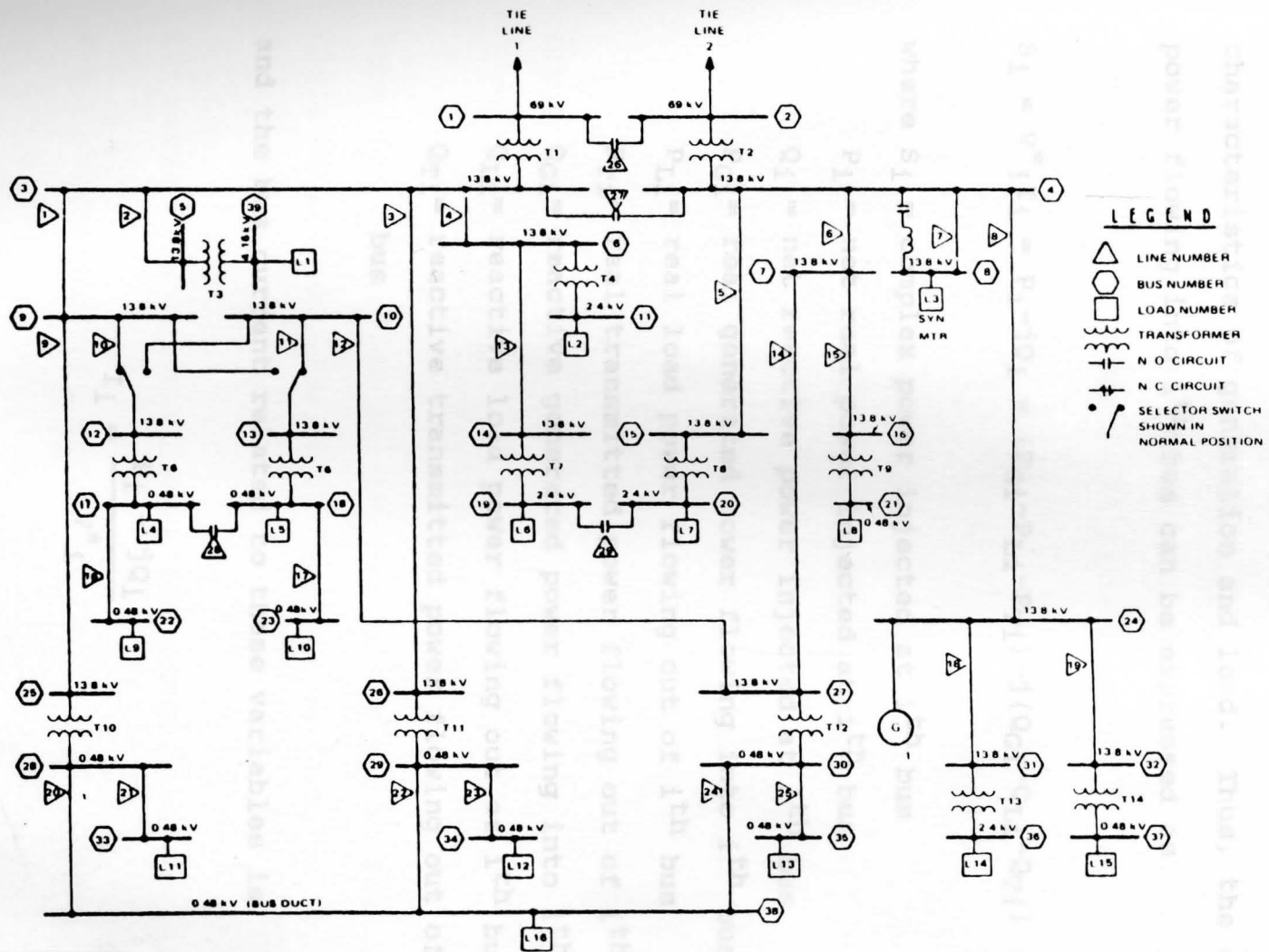


FIG 2.1 One line connection diagram

The electrical conditions at each bus are defined in terms of active and reactive power because of the physical characteristics of generation and load. Thus, the complex power flowing into  $i^{\text{th}}$  bus can be expressed as

$$S_i = V_i^* I_i = P_i - jQ_i = (P_{Gi} - P_{Li} - P_{Ti}) - j(Q_{Gi} - Q_{Li} - Q_{Ti}) \quad (2.2.3)$$

where  $S_i$  = complex power injected at  $i^{\text{th}}$  bus

$P_i$  = net real power injected at  $i^{\text{th}}$  bus

$Q_i$  = net reactive power injected at  $i^{\text{th}}$  bus

and  $P_{Gi}$  = real generated power flowing into  $i^{\text{th}}$  bus

$P_{Li}$  = real load power flowing out of  $i^{\text{th}}$  bus

$P_{Ti}$  = real transmitted power flowing out of  $i^{\text{th}}$  bus

similarly  $Q_{Gi}$  = reactive generated power flowing into  $i^{\text{th}}$  bus

$Q_{Li}$  = reactive load power flowing out of  $i^{\text{th}}$  bus

$Q_{Ti}$  = reactive transmitted power flowing out of  $i^{\text{th}}$

bus

and the bus current related to these variables is

$$I_i = \frac{P_i - jQ_i}{V_i^*} \quad (2.2.4)$$

## 2.3 LOAD FLOW CALCULATION BY THE NEWTON-RAPHSON METHOD

### 2.3.1 NEWTON-RAPHSON METHOD

The Newton-Raphson method [10, 11] is the most powerful iterative method for solving the set of nonlinear algebraic equations because of the fact

### c. Line flow equations

Line flow can be calculated only after the solution to the bus voltages is completed. The current at bus  $i$  in the line connecting bus  $i$  to bus  $j$  is

$$I_{ij} = (V_i - V_j)y_{ij} + \frac{V_i y'_{ij}}{2} \quad (2.2.5)$$

where  $y_{ij}$  = line admittance

and the  $y'_{ij}$  = total line charging admittance

and the line power flow from bus  $i$  to bus  $j$  is

$$P_{ij} - jQ_{ij} = V_i^* I_{ij} \quad (2.2.6)$$

Similarly, at bus  $j$ , the power flow from bus  $j$  to bus  $i$  is

$$P_{ji} - jQ_{ji} = V_j^* I_{ji} \quad (2.2.7)$$

Thus, the power loss in line  $i$ - $j$  is the sum of the power flows determined from eq.(2.2.6) and eq.(2.2.7)

## 2.3 LOAD FLOW CALCULATION BY THE NEWTON-RAPHSON METHOD

### 2.3.1 NEWTON-RAPHSON METHOD

The Newton-Raphson method [10,13] is the most powerful iterative method for solving the system of nonlinear algebraic equations because of its fast

convergence. The method is based on approximating a nonlinear function to the Taylor's series expansion [5]. By giving a set of nonlinear equations.

$$\begin{aligned}
 f_1(x_1, x_2, \dots, x_n) &= K_1 \\
 f_2(x_1, x_2, \dots, x_n) &= K_2 \\
 \dots\dots\dots &= \dots \\
 f_n(x_1, x_2, \dots, x_n) &= K_n
 \end{aligned}
 \tag{2.3.1}$$

and the initial estimates for the solution vector are

$$x^0_1, x^0_2, \dots, x^0_n \tag{2.3.2}$$

Assume  $dx_1, dx_2, \dots, dx_n$  are the corrections required for  $x^0_1, x^0_2, \dots, x^0_n$ , respectively, so that the eq. (2.3.1) are solved. Thus

$$\begin{aligned}
 K_1 &= f_1(x^0_1+dx_1, x^0_2+dx_2, \dots, x^0_n+dx_n) \\
 K_2 &= f_2(x^0_1+dx_1, x^0_2+dx_2, \dots, x^0_n+dx_n) \\
 \dots &= \dots\dots\dots \\
 K_n &= f_n(x^0_1+dx_1, x^0_2+dx_2, \dots, x^0_n+dx_n)
 \end{aligned}
 \tag{2.3.3}$$

According to Taylor's theorem for a function of two or more variables, the right hand side of each eq.(2.3.3) can be expanded to



or

$$\begin{bmatrix} \overline{df} \end{bmatrix} = \begin{bmatrix} \overline{J} \end{bmatrix} \begin{bmatrix} \overline{dx} \end{bmatrix} \quad (2.3.6)$$

where  $\overline{df}$  = the vector of the mismatch functions

$\overline{J}$  = Jacobian matrix

$\overline{dx}$  = vector of the corrections

The elements of the matrix  $[\overline{df}]$  and  $[\overline{J}]$  are evaluated by substituting the current values of  $x_i$ . Hence, a solution for the  $dx_i$  can be obtained by solving a system of linear equations. That is

$$\begin{bmatrix} \overline{dx} \end{bmatrix} = \begin{bmatrix} \overline{J} \end{bmatrix}^{-1} \begin{bmatrix} \overline{df} \end{bmatrix} \quad (2.3.7)$$

and the new values for  $x_i$  are evaluated from

$$x_i^{k+1} = x_i^k + dx_i \quad (2.3.8)$$

The process is repeated until the mismatch function  $[\overline{df}]$  is less than the specified tolerance, and then the solution of nonlinear system can be obtained.

### 2.3.2 APPLICATION OF THE NEWTON-RAPHSON METHOD TO LOAD-FLOW EQUATIONS IN POLAR COORDINATES USING $Y_{bus}$

The Newton-Raphson method is popularly used in solving loadflow problems because it is reliable and extremely fast in convergence. The rate of convergence of



the method is relatively independent of the size of the nonlinear system.

To apply the Newton-Raphson method to loadflow problems, the slack bus, at which the magnitude and phase angle of the voltage are specified, is not included in the iteration process. Therefore, the equation of the complex power at bus  $i$  in  $N$ -bus system can be expressed as

$$S_i = P_i - jQ_i = V_i^* I_i = \sum_{j=1}^N |V_i V_j Y_{ij}| \angle \beta_{ij} + \theta_j - \theta_i \quad (2.3.9)$$

Therefore, it can be expressed as

$$\begin{aligned} P_i &= \sum_{j=1}^N |V_i V_j Y_{ij}| \cos(\beta_{ij} + \theta_j - \theta_i) \\ Q_i &= -\sum_{j=1}^N |V_i V_j Y_{ij}| \sin(\beta_{ij} + \theta_j - \theta_i) \end{aligned} \quad (2.3.10)$$

This formulation results in a set of nonlinear simultaneous equations, two for each P-Q bus and one for each P-V bus. The known values for P-Q buses are real and reactive load bus powers while the known values for P-V buses are bus voltage magnitudes and real generated bus power. If the slack bus is set to be bus #1, the calculation of the nonlinear loadflow problem will be started at bus #2, where bus #2 to  $g$  ( $g$  = total number of generators in system) are P-V buses and bus # $g+1$  to  $n$  are P-Q buses. Thus, there are  $2N-g-1$  equations to be solved

for a loadflow solution.

The Newton-Raphson method requires that a set of linear equations be formed expressing the relationship between the changes in real and reactive powers and the components of bus voltages as follows :

The elements of the Jacobian matrix from eq. (2.3.11) can be calculated by the following procedure

$dP_2$ $\cdot$ $\cdot$ $dP_n$	=	$\frac{dP_2}{d\theta_2} \dots \frac{dP_2}{d\theta_n}$ $\dots \dots \dots$ $\frac{dP_n}{d\theta_2} \dots \frac{dP_n}{d\theta_n}$	$\frac{dP_2}{d V_{g+1} } \dots \frac{dP_2}{d V_n }$ $\dots \dots \dots$ $\frac{dP_n}{d V_{g+1} } \dots \frac{dP_n}{d V_n }$	$d\theta_2$ $\cdot$ $\cdot$ $d\theta_n$	(2.3.11)
$dQ_{g+1}$ $\cdot$ $\cdot$ $dQ_n$		$\frac{dQ_{g+1}}{d\theta_2} \dots \frac{dQ_{g+1}}{d\theta_n}$ $\dots \dots \dots$ $\frac{dQ_n}{d\theta_2} \dots \frac{dQ_n}{d\theta_n}$	$\frac{dQ_{g+1}}{d V_{g+1} } \dots \frac{dQ_{g+1}}{d V_n }$ $\dots \dots \dots$ $\frac{dQ_n}{d V_{g+1} } \dots \frac{dQ_n}{d V_n }$	$d V_{g+1} $ $\cdot$ $\cdot$ $d V_n $	

where the coefficient matrix is the Jacobian matrix and the 1<sup>st</sup> bus is the slack bus. The matrix form of eq. (2.3.11) can be written as

$$\begin{array}{c} \bar{dP} \\ \hline \bar{dQ} \end{array} = \begin{array}{|c|c|} \hline J_1 & J_2 \\ \hline J_3 & J_4 \\ \hline \end{array} \begin{array}{c} \bar{d\theta} \\ \hline \bar{d|V|} \end{array} \quad (2.3.12)$$

The elements of the Jacobian matrix from eq.(2.3.12) can be calculated by the following equations :

for  $J_1$  :

$$\frac{dP_i}{d\theta_j} = -|V_i V_j Y_{ij}| \sin(\beta_{ij} + \theta_j - \theta_i) \quad ; \text{ for } i \neq j \quad (2.3.13)$$

$$\frac{dP_i}{d\theta_i} = \sum_{\substack{j=1 \\ j \neq i}}^N |V_i V_j Y_{ij}| \sin(\beta_{ij} + \theta_j - \theta_i) \quad ; \text{ for } i=j$$

for  $J_2$  :

$$\frac{dP_i}{d|V_j|} = |V_i Y_{ij}| \cos(\beta_{ij} + \theta_j - \theta_i) \quad ; \text{ for } i \neq j \quad (2.3.14)$$

$$\frac{dP_i}{d|V_i|} = 2|V_i| Y_{ii} \cos(\beta_{ii}) + \sum_{\substack{j=1 \\ j \neq i}}^N |V_j| Y_{ij} \cos(\beta_{ij} + \theta_j - \theta_i) ; \text{ for } i=j$$

for  $J_3$  :

$$\frac{dQ_i}{d\theta_j} = -|V_i||V_j|Y_{ij}\cos(\beta_{ij}+\theta_j-\theta_i) \quad ; \text{ for } i \neq j \quad (2.3.15)$$

$$\frac{dQ_i}{d\theta_i} = \sum_{\substack{j=1 \\ j \neq i}}^N |V_iV_jY_{ij}|\cos(\beta_{ij}+\theta_j-\theta_i) \quad ; \text{ for } i=j$$

for  $J_4$  :

$$\frac{dQ_i}{d|V_j|} = -|V_i|Y_{ij}\sin(\beta_{ij}+\theta_j-\theta_i) \quad ; \text{ for } i \neq j \quad (2.3.16)$$

$$\frac{dQ_i}{d|V_i|} = -2|V_i|Y_{ii}\sin(\beta_{ii}) - \sum_{\substack{j=1 \\ j \neq i}}^N |V_i|Y_{ij}\sin(\beta_{ij}+\theta_j-\theta_i); \text{ for } i=j$$

Given an initial set of bus voltages, the real and reactive power can be calculated by eq.(2.3.10). The changes in power are the differences between the specified and the calculated values.

$$\begin{aligned} dp^k_i &= P_i(\text{sch}) - P^k_i(\text{cal}) \\ dQ^k_i &= Q_i(\text{sch}) - Q^k_i(\text{cal}) \end{aligned} \quad (2.3.17)$$

The estimated bus voltage magnitudes and bus angles are used to evaluate the elements of the Jacobian and power mismatch functions. The linear set of eq.(2.3.11) can be solved for  $dV_i$  and  $d\theta_i$  by a direct or iterative method. Then, the new estimates of bus voltage magnitude and angles are

$$\begin{aligned} V_i^{k+1} &= V_i^k + dV_i \\ \theta_i^{k+1} &= \theta_i^k + d\theta_i \end{aligned} \quad (2.3.18)$$

The process is repeated until  $P_i^k$  and  $Q_i^k$  for all buses are within a specified tolerance. The sequence of steps for the load flow solution by The Newton-Raphson method is shown in Fig. 2.2

#### 2.4 INFORMATION OBTAINED IN LOAD FLOW STUDIES

A printout of the load-flow problem results consists of a number of tabulations. The most important information to be considered first is the table that lists each bus numbers, bus voltage magnitude in per unit and phase angle in degrees, generation and load at each bus in megawatts and megavars, and line charging. Accompanying the bus and line information are the power flow from that bus over each transmission line connected to the bus and the power losses in the transmission line itself in megawatts and megavars.

In the operation of power systems, any appreciable drop in voltage on the primary of a transformer caused by a change of load may make it desirable to change the tap setting on transformers provided with adjustable taps in order to maintain proper voltage at the load. Where a tap-changing transformer has been specified to keep the voltage at a bus within designated tolerance limits, the

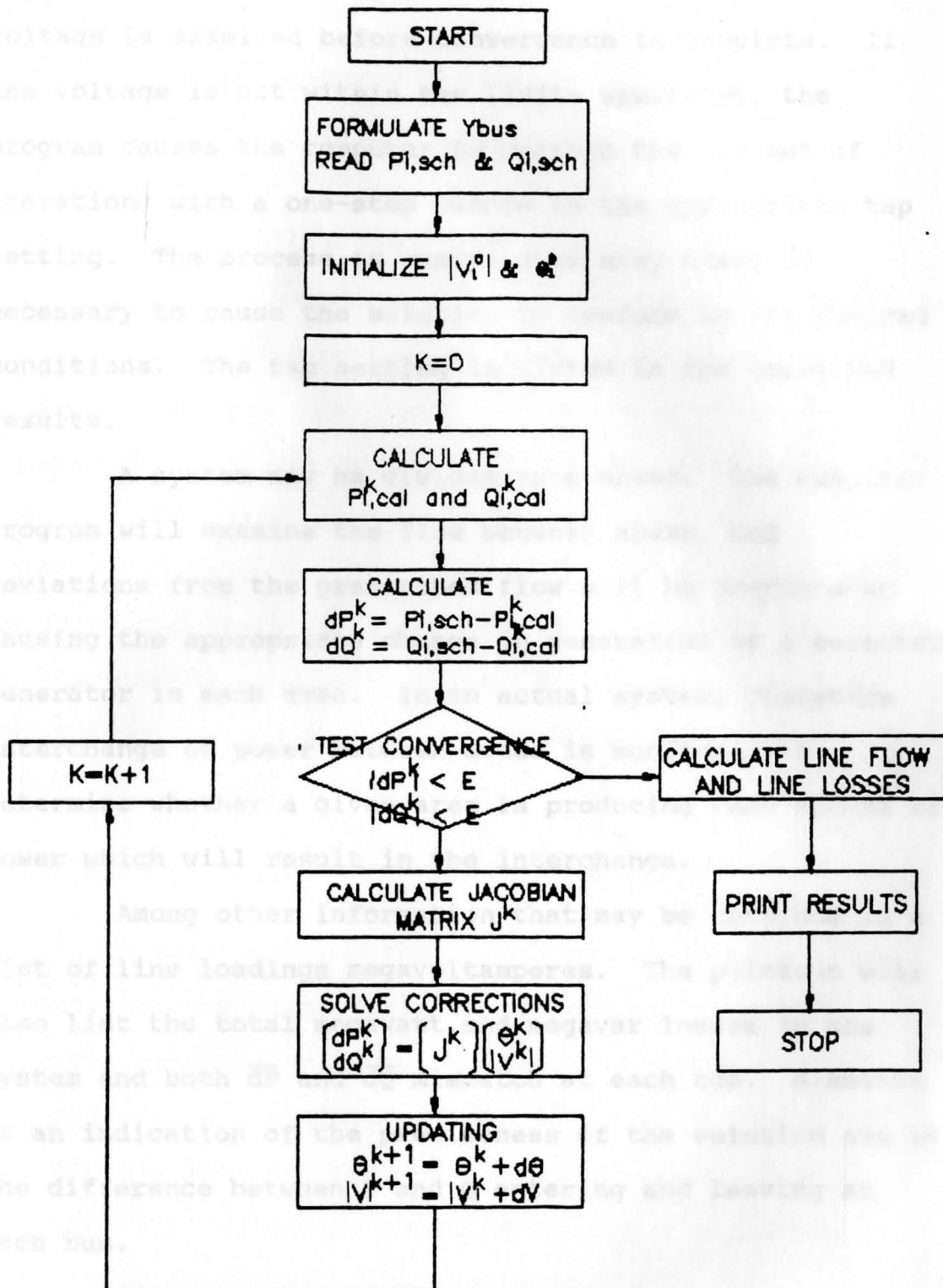


FIG. 2.2 Computer flow diagram of the NEWTON-RAPHSON method for loadflow solutions

## Chapter III

voltage is examined before convergence is complete. If the voltage is not within the limits specified, the program causes the computer to perform the new set of iterations with a one-step change in the appropriate tap setting. The process is repeated as many times as necessary to cause the solution to conform to the desired conditions. The tap setting is listed in the tabulated results.

A system may be divided into areas. The computer program will examine the flow between areas, and deviations from the prescribed flow will be overcome by causing the appropriate change in generation of a selected generator in each area. In an actual system, operation interchange of power between areas is monitored to determine whether a given area is producing that amount of power which will result in the interchange.

Among other information that may be obtained is a list of line loadings megavoltamperes. The printout will also list the total megawatt and megavar losses in the system and both  $\bar{d}P$  and  $\bar{d}Q$  mismatch at each bus. Mismatch is an indication of the preciseness of the solution and is the difference between P and Q entering and leaving at each bus.

encountered in numerical computation.

## Chapter III

### ERROR ANALYSIS FOR NUMERICAL COMPUTATION

#### 3.1 INTRODUCTION

In solving the system of nonlinear algebraic equations by the iterative method (or numerical method), the method sometimes might not converge to the solution. The reason is that some errors can arise to cause an inaccuracy in the computation during the iterative process. This makes the predicted values by the iterative method unreliable at any iteration. Finally, divergence can occur and the solution of the system may not be found. In this chapter, the main sources of errors in numerical computation and convergence analysis of the Newton-Raphson method are discussed in detail.

#### 3.2 SOURCES OF ERRORS IN NUMERICAL COMPUTATION

In this section, the major errors that can arise in numerical computation are introduced [6,7]. Some errors, such as human error, computer hardware error or some failure in a software system will not be discussed here because they are supposed to be reliable. The following list of errors contains the major errors usually encountered in numerical computation.



### a. Computer Rounding Error

This error can arise when the calculating devices, such as computers, cannot handle numbers that have more digits than its finite word length (machine precision). This makes the product of two or more numbers inaccurate in subsequent calculations. Thus, the product of the numbers must be rounded off. The effect of such a computer rounding error can be significant in an extensive calculation, or in a calculation in which the least significant digits of the number become significant. Table 3.2 shows the machine precision  $e_m$ .

computer	condition	$e_m$ in base 10
IBM 370	short precision	9.5 E-7
IBM PC	Basic DEFSNG	5.96E-8
IBM PC	Basic DEFDBL	1.39E-17
HP 85	HP Basic	3.46E-1

Table 3.1 Machine Precision

### b. Truncation Errors

These errors are the errors occur when a limiting process is truncated (broken off) before one has come to the limiting value. In the Newton-Raphson method,

value for the next iteration be  
truncation errors occur when the terms that are the order greater than one of the indefinite Taylor series are neglected to approximate the nonlinear function with the linear function. The resulting defined error is significant and causes the Newton-Raphson method divergence whenever the initial estimate value is not close enough to the roots of the nonlinear equation system.

### 3.3 CONVERGENCE ANALYSIS OF THE NEWTON-RAPHSON METHOD

To investigate the effect of the truncation errors [6], regardless of the effect of the computer rounding error, on the convergence of the Newton-Raphson method, a system of a nonlinear function with 1 variable  $f(x)=0$  is expanded to the second-order terms of indefinite Taylor's series. That is

$$0 = f(x^*) = f(x^k) + (x^* - x^k) f'(x^k) + \frac{1}{2} (x^* - x^k)^2 f''(x^k) \quad (3.3.1)$$

where  $x^*$  is the root of the nonlinear system.

After dividing eq.(3.3.1) by  $f'(x^k)$

$$x^* - (x^k - \frac{f(x^k)}{f'(x^k)}) = -\frac{1}{2} \cdot (x^* - x^k)^2 \frac{f''(x^k)}{f'(x^k)} \quad (3.3.2)$$

let the error in the current iteration and the predicted

value for the next iteration be

$$e^k = x^k - x^*$$

and

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)} \quad (3.3.3)$$

where  $\frac{f(x^k)}{f'(x^k)}$  is the Newton-Raphson step.

Thus eq.(3.3.2) can be written as

$$e^{k+1} = \frac{1}{2} \cdot e^2(k) \cdot \frac{f''(x^k)}{f'(x^k)} \quad (3.3.4)$$

According to eq.(3.3.4), the error on the current iteration is the function of the square of the error on the previous iteration. From this relation, it follows that the Newton-Raphson method will converge to the solution of the nonlinear equation system if and only if sufficiently good initial estimate values can be provided; otherwise, an increase of error can make the predicted values on the next iteration be worse than the previous iteration. This leads to the divergence of the Newton-Raphson method.

## Chapter IV

### THE HYBRID ALGORITHM

#### 4.1 INTRODUCTION

The error analysis that has been discussed in Chapter III shows that the initial values of all unknowns by the Newton-Raphson method have to be carefully selected to be close enough to the roots of the system of nonlinear algebraic equations. Regardless of the effect of the computer rounding error, an increase of the defined truncation error can arise and cause the method to diverge when the initial values are far from the roots. To solve this problem, the Hybrid algorithm is introduced in this chapter. The purpose of the algorithm is to improve the efficiency in convergence of the existing Newton-Raphson method to be able to converge to the solutions with a wide range of initial values. At the end of this chapter, the proposed algorithm is applied to the Newton-Raphson method in solving nonlinear power flow problems.

#### 4.2 BACKGROUND

The Hybrid algorithm [1] was developed by M.J.D Powell (1970). It is based on the idea of the Lavenberge/Marquart method [2,3] that is generally used in solving nonlinear least square problems. The idea of the proposed algorithm is to introduce the method of steepest

descent [2,3,9] to the existing Newton-Raphson method. The important advantage of the method of steepest descent is that the method is not as sensitive to initial values. This makes the method of steepest descent able to converge to the solutions of the system of nonlinear equations with a wide range of initial estimates. By using a gradient technique [2] in searching for the solutions along curvature of the nonlinear functions, the defined truncation error can be reduced and the method can be used for making solutions of the system of nonlinear equations more accurate for those cases where the Newton-Raphson method diverges, when its initial values are far from the roots. Despite the advantageous property of this method, a larger number of iterations is required, which leads to a slow convergence of this method. The comparison of advantages and disadvantages between the Newton-Raphson method and the method of steepest descent is shown on Table 4.1

To obtain the advantages of both Newton-Raphson method and the method of steepest descent, the concept of a trust region [2,3] is introduced. The purpose of the trust region is to restrict the step predicted by the Newton-Raphson method to be inside the appropriate region, in which the defined truncation error will not affect and cause a divergence, and to establish the switching policies between these two methods. By using the proposed

### NEWTON-RAPHSON METHOD

#### Advantages

- The method has a fast convergence characteristic and can converge to the solutions of the system of nonlinear equations in just a few iterations.
- The rate of convergence by the method is independent of the size of the nonlinear system.

#### Disadvantages

- The initial values of unknowns required by this method must be close to the roots of the system of nonlinear equations to avoid the effect of defined truncation error.

### THE METHOD OF STEEPEST DESCENT

#### Advantages

- The method is able to converge to the solutions of the system of nonlinear equations with a wide range of initial values.

#### Disadvantages

- The method has a zigzagging and slow converging characteristic.

TABLE 4.1 The comparison of the advantages and disadvantages between the Newton-Raphson method and the method of steepest descent

algorithm, when the initial values are far from the roots, the iteration obtains the predicted step in the steepest descent direction, and then switches to the Newton-Raphson iteration to obtain the Newton-Raphson step for a fast convergence when they approach the neighborhood of the solutions.

#### 4.3 THE METHOD OF STEEPEST DESCENT

A typical iteration of a line search algorithm [2] for optimization, subject to nonlinear constraints, calculates the predicted step and updates the predicted value for the next iteration by the line function

$$\bar{x}^{k+1} = \bar{x}^k + u^k \bar{s}^k \quad (4.3.1)$$

where  $\bar{s}^k$  = descent direction

$u^k$  = positive scalar

The relationship in eq.(4.3.1) is shown in Figure 4.1. There are several concepts associated with the line function. First, it is a vector function of a scalar, namely,  $u$ . Notationally,  $\bar{x}^* = \bar{x}^*(u)$ . Assuming the objective function of the optimization problem is  $F(\bar{x})$ , the scalar function along the line is a function of only  $u$ , when given a starting point  $\bar{x}$  and direction  $\bar{s}$ , that is,  $F(\bar{x}^*) = F(u)$ . A line search is the process of finding

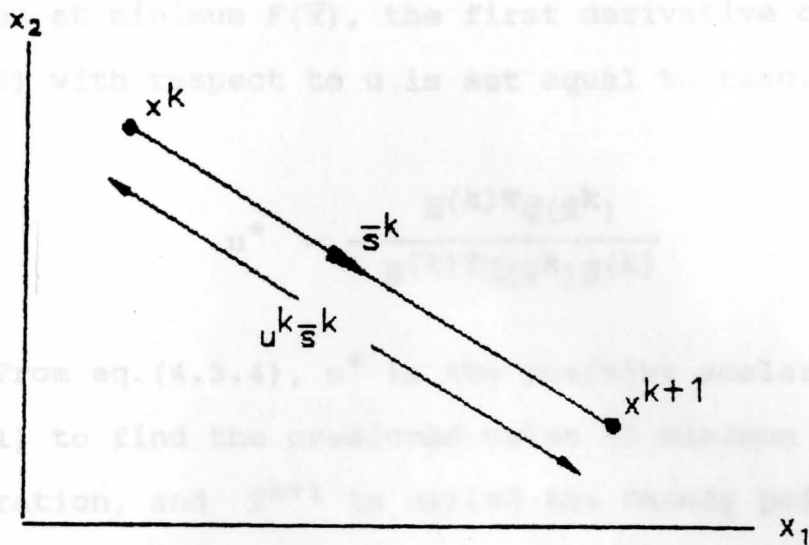


FIG 4.1 A straight line in 2-space illustrating vector search direction  $s^k$  and scalar  $u^k$

some  $u$ , say  $u^*(k)$  where  $u^*(k)$  is a positive scalar chosen at iteration  $k$  to minimize  $F(u)$ . In order to find  $u^*$ , the general quadratic function [2] in vector notation is recalled. That is

$$F(\bar{x}) = c + \bar{g}^T(\bar{x}) \begin{bmatrix} \bar{x} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \bar{x} \end{bmatrix}^T \bar{H}(\bar{x}) \begin{bmatrix} \bar{x} \end{bmatrix} \quad (4.3.2)$$

where  $\bar{g}(\bar{x}) =$  gradient of the objective function  $F(\bar{x})$

$\bar{H}(\bar{x}) =$  Hessian matrix [2]

Substituting eq.(4.3.1) in eq.(4.3.2) yields

$$F(\bar{x}^{k+1}) = c + \bar{g}^T(\bar{x}^k) \begin{bmatrix} \bar{x}^k + u^k \bar{s}^k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \bar{x}^k + u^k \bar{s}^k \end{bmatrix}^T \bar{H}(\bar{x}^k) \begin{bmatrix} \bar{x}^k + u^k \bar{s}^k \end{bmatrix} \quad (4.3.3)$$



To find  $u^*$  at minimum  $F(\bar{x})$ , the first derivative of eq.(4.3.3) with respect to  $u$  is set equal to zero. That yields

$$u^* = - \frac{\bar{s}(k)^T \bar{g}(\bar{x}^k)}{\bar{s}(k)^T \bar{H}(\bar{x}^k) \bar{s}(k)} \quad (4.3.4)$$

From eq.(4.3.4),  $u^*$  is the positive scalar used in eq.(4.3.1) to find the predicted value at minimum for the next iteration, and  $\bar{x}^{k+1}$  is called the **Cauchy point**.

To apply the line search technique for solving the system of nonlinear algebraic equations, the vector of a nonlinear system with  $N$  equations and  $N$  unknowns is given by

$$\bar{f}_i(\bar{x}) = 0 \quad ; i=1,2,\dots,N \quad (4.3.5)$$

and the objective function of eq.(4.3.5) is set to be

$$\text{minimize} \quad F(\bar{x}) = \sum_{i=1}^N f_i^2(\bar{x}) \quad (4.3.6)$$

In this instance,  $F(\bar{x})$  takes on the minimum value zero at all solutions of the nonlinear equation system.

The descent direction  $\bar{s}^k$  is the vector of the first derivatives of eq.(4.3.6). That is

$$\text{gradient } \bar{g}(\bar{x}) = \left[ \begin{array}{c} \frac{dF(\bar{x})}{dx_1}, \frac{dF(\bar{x})}{dx_2}, \dots, \frac{dF(\bar{x})}{dx_n} \end{array} \right]^T$$

$$= 2 \left[ \sum_{i=1}^N f_i(\bar{x}) \cdot \frac{df_i(\bar{x})}{dx_1}, \dots, \sum_{i=1}^N f_i(\bar{x}) \cdot \frac{df_i(\bar{x})}{dx_n} \right]^T$$

$$= 2 \cdot \bar{J}^T(\bar{x}) \cdot \bar{f}(\bar{x}) \quad (4.3.7)$$

It is obvious that the direction  $-\bar{g}(\bar{x})$  is the direction in which  $F(\bar{x})$  decreases most rapidly and the descent direction  $\bar{s}^k$  of the line search is in the direction of steepest-descent. From eq.(4.3.1), the predicted value for the next iteration in the steepest-descent direction is given by

$$\bar{x}^{k+1} = \bar{x}^k - u \cdot \bar{g}^k(\bar{x}) \quad (4.3.8)$$

where  $\bar{g}(\bar{x}^k) =$  gradient of  $F(\bar{x})$  calculated in eq.(4.3.7)  
 $u =$  a positive scalar that is chosen in order to reduce  $F(\bar{x}^{k+1}) < F(\bar{x}^k)$

To find the scalar  $u^*$  at the predicted minimum or at the solutions (cauchy point), the vector direction  $\bar{s}^k = -\bar{g}^k(\bar{x})$  is substituted into eq.(4.3.4). That is

$$u^* = \frac{\bar{g}^T(\bar{x}^k) \cdot \bar{g}(\bar{x}^k)}{\bar{g}^T(\bar{x}^k) \cdot \bar{H}(\bar{x}^k) \cdot \bar{g}(\bar{x}^k)} \quad (4.3.9)$$

From the Gauss-Newton formula [2], The Hessian matrix  $\bar{H}(\bar{x}^k)$  in eq.(4.3.9) can be approximated by

$$\bar{H}(\bar{x}^k) = \bar{J}^T(\bar{x}^k) \cdot \bar{J}(\bar{x}^k) \quad (4.3.10)$$

By substituting eq. (4.3.10) into eq. (4.3.8), the scalar  $u^*$  of the line search in the direction of steepest-descent can be calculated by

$$u^* = \frac{\bar{g}^T(\bar{x}^k) \cdot \bar{g}(\bar{x}^k)}{\bar{g}^T(\bar{x}^k) \cdot \bar{J}^T(\bar{x}^k) \cdot \bar{J}(\bar{x}^k) \cdot \bar{g}(\bar{x}^k)} \quad (4.3.11)$$

That yields

$$u^* = \frac{\| \bar{g}(\bar{x}^k) \|^2}{\| \bar{J}(\bar{x}^k) \cdot \bar{g}(\bar{x}^k) \|^2} \quad (4.3.12)$$

From eq.(4.3.8), the predicted point in the steepest descent direction, at the solution called the "Cauchy point", can be found by the following equation

$$\bar{x}^{k+1} = \bar{x}^k - u^* \bar{g}(\bar{x}^k) \quad (4.3.13)$$

where  $u^*$  and  $\bar{g}(\bar{x}^k)$  can be calculated by eq.(4.3.12) and eq.(4.3.7), respectively.

Figure 4.2 shows a view of zigzagging and a slow convergence of the steepest descent method.

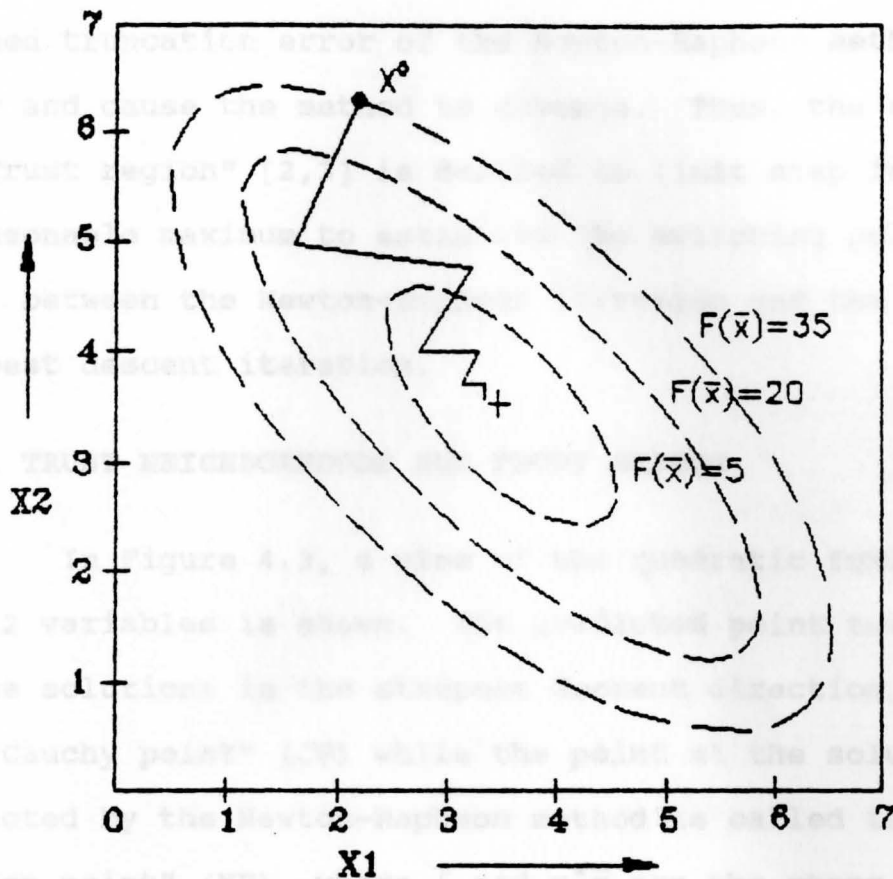


FIG 4.2. Line searches on a quadratic function. The steepest descent direction usually causes zigzagging and slow convergence.

#### 4.4 TRUST REGION AND SWITCHING POLICIES

The standard Newton-Raphson method converges to the solutions of the nonlinear equation system at a quadratic rate, but without restrictions on its predicted step size, it is often unreliable on any iteration. On the other hand, when the starting point is well removed from the solutions, the quadratic of the nonlinear

functions becomes no longer valid. An increase of the defined truncation error of the Newton-Raphson method can occur and cause the method to diverge. Thus, the concept of "Trust region" [2,3] is defined to limit step length to a reasonable maximum to establish the switching policies [1,2] between the Newton-Raphson iteration and the steepest descent iteration.

#### 4.4.1 TRUST NEIGHBORHOODS AND TRUST RADIUS

In Figure 4.3, a view of the quadratic function with 2 variables is shown. The predicted point terminates at the solutions in the steepest descent direction, called the "Cauchy point" (CP) while the point at the solutions predicted by the Newton-Raphson method is called the "Newton point" (NP), where  $\bar{\delta}$  and  $u^*\bar{g}$  are the steps predicted by the Newton-Raphson method and the method of the steepest descent, respectively. A circular neighborhood of radius  $R$  about  $x^0$ , called "trust region", has been added to limit the step predicted by the Newton-Raphson method to be inside on every iteration. For example, consider the results of centering that neighborhood of radius  $R$  at every turning point,  $\bar{x}^k$ : the zigzagging characteristic of steepest descent (Figure 4.2) is avoided well before arriving in the neighborhood of the solutions, and the rapid convergence of the Newton-Raphson iteration is likely to prevail. In between these two

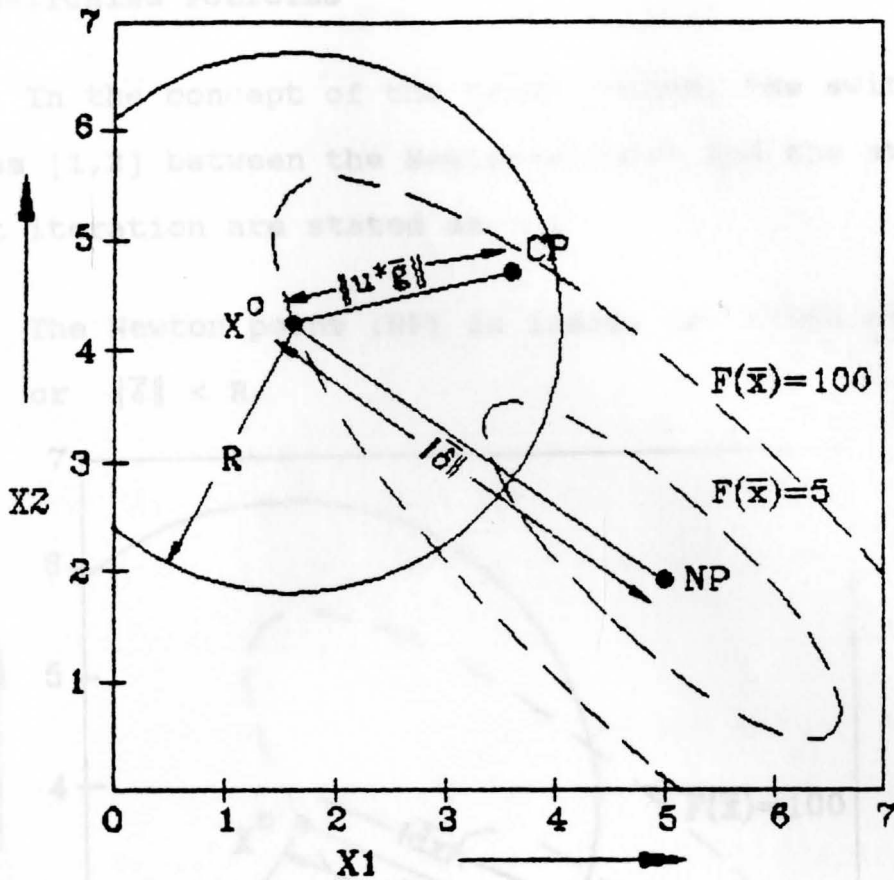


FIG. 4.3 The view of quadratic function with 2 variables when the trust region with radius  $R$  is applied

states, the Newton-Raphson step is limited to be less than the length of the trust radius  $R$ , presumably providing a reasonable rate of progress. In the case when the Newton-Raphson step is greater than  $R$ , the iteration will bias the predicted step into the steepest descent direction.

In this case, which is illustrated in Figure 4.3, the point at the solution predicted by the Newton-Raphson (NP) is assumed to be inside the neighborhood of a

#### 4.4.2 SWITCHING POLICIES

In the concept of the trust region, the switching policies [1,2] between the Newton-Raphson and the steepest descent iteration are stated as

**case 1** The Newton point (NP) is inside the trust region, or  $\|\bar{\delta}\| < R$ .

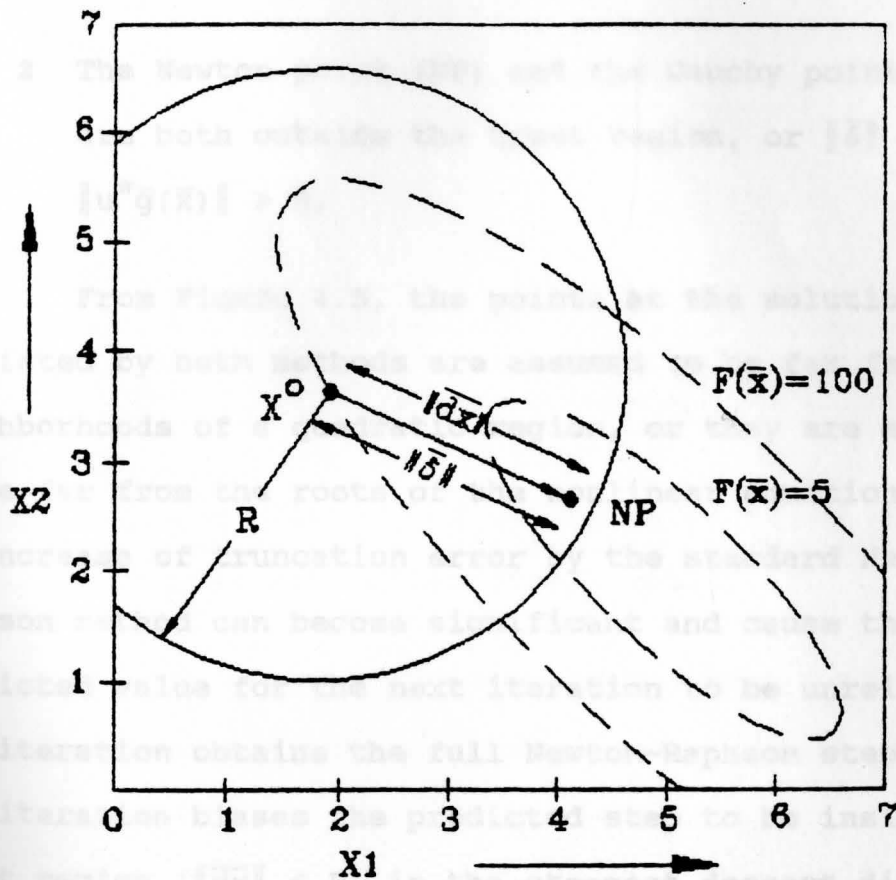


Fig 4.4 The view of switching policies when the Newton point (NP) is inside the trust region

In this case, which is illustrated in Figure 4.4, the point at the solution predicted by the Newton-Raphson (NP) is assumed to be inside the neighborhoods of a

quadratic region, or it is approximately close to the roots of the nonlinear equation system. Thus, the iteration obtains the full Newton-Raphson step ( $\delta$ ) for fast convergence. That is

$$\bar{dx} = \bar{\delta}^k \quad (4.4.1)$$

where  $\bar{x}^{k+1} = \bar{x}^k + \bar{dx}$  is the predicted value for the next iteration.

**case 2** The Newton point (NP) and the Cauchy point (CP) are both outside the trust region, or  $\|\bar{\delta}\| > R$  and  $\|u^* \bar{g}(\bar{x})\| > R$ .

From Figure 4.5, the points at the solutions predicted by both methods are assumed to be far from the neighborhoods of a quadratic region, or they are assumed to be far from the roots of the nonlinear equation system. An increase of truncation error by the standard Newton-Raphson method can become significant and cause the predicted value for the next iteration to be unreliable if the iteration obtains the full Newton-Raphson step. Thus, the iteration biases the predicted step to be inside the trust region ( $\|\bar{dx}\| \leq R$ ) in the steepest descent direction to reduce the effect of the defined truncation error. To obtain the predicted step  $dx$  to be inside the trust region, the length of the predicted step is set to be equal to the trust radius  $R$  for a maximum length. From



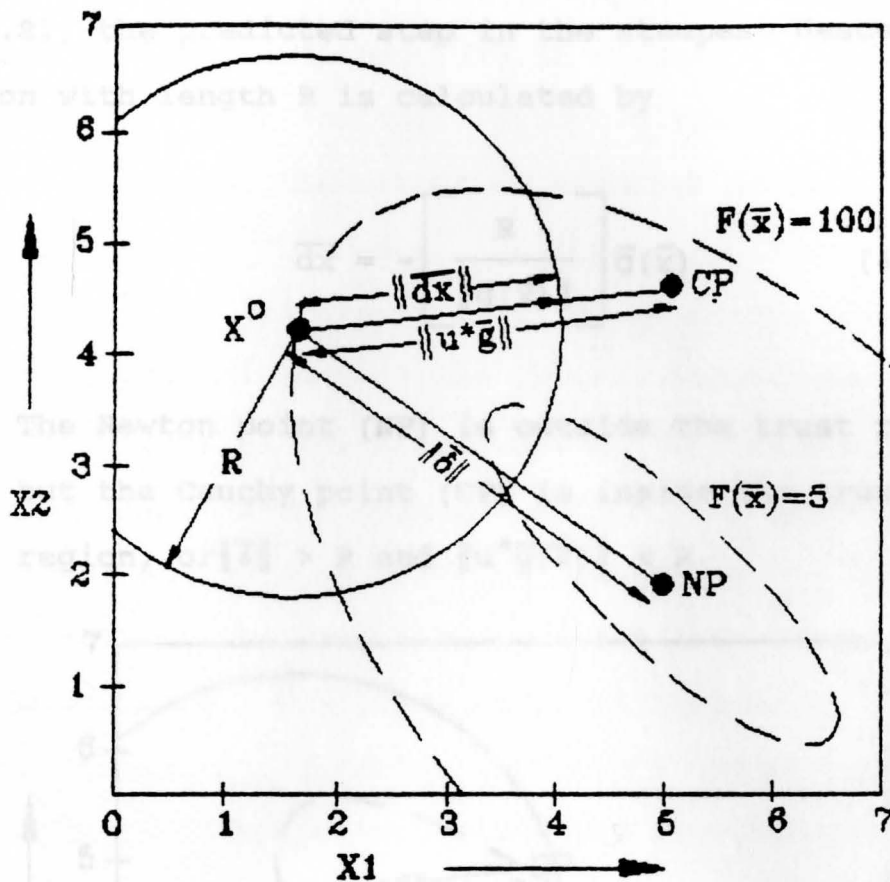


FIG. 4.5 The view of the switching policies when the Newton point (NP) and the Cauchy point (CP) are both outside trust region.

eq.(4.3.8), the length of the predicted step in the steepest descent direction is bound equal to be

$$\|u\bar{g}(\bar{x})\| = R \quad (4.4.2)$$

That yields

$$u = \frac{R}{\|\bar{g}(\bar{x})\|} \quad (4.4.3)$$

By substituting the positive scalar  $u$  from eq.(4.4.3) to

eq.(4.3.8), the predicted step in the steepest descent direction with length  $R$  is calculated by

$$\bar{dx} = - \left[ \frac{R}{\|\bar{g}(\bar{x})\|} \right] \bar{g}(\bar{x}) \quad (4.4.4)$$

**case 3** The Newton point (NP) is outside the trust region, but the Cauchy point (CP) is inside the trust region, or  $\|\bar{\delta}\| > R$  and  $\|u^*\bar{g}(\bar{x})\| < R$ .

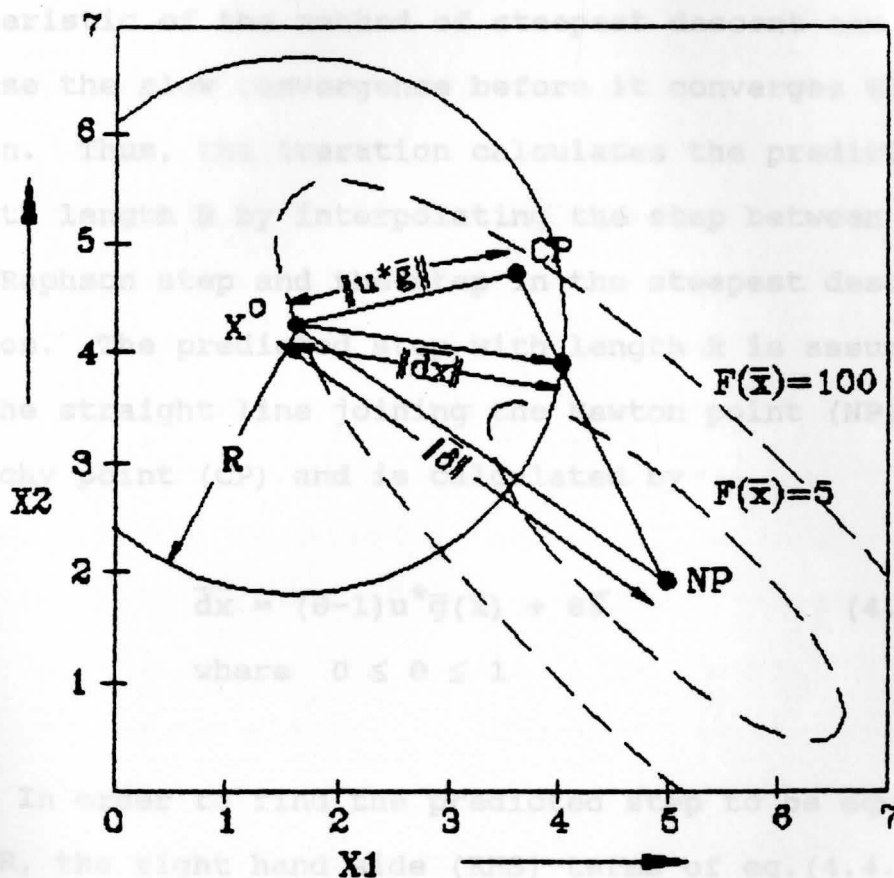


FIG. 4.6 The view of the switching policies when the Newton point (NP) is outside the trust region, but the Cauchy point (CP) is inside.

In Figure 4.6, the point at the solution predicted by the Newton-Raphson method (NP) is assumed to be far from the roots of the nonlinear equation system, but the Cauchy point (CP) by the method of steepest descent shows that it is close to the roots. In this case, the iteration does not preferably bias the predicted step in only the steepest descent direction even though the Newton point (NP) is outside the trust region. The reason for this is that, inside the neighborhoods of the solution of the nonlinear equation system, the zigzagging characteristic of the method of steepest descent can arise and cause the slow convergence before it converges to the solution. Thus, the iteration calculates the predicted step with length  $R$  by interpolating the step between the Newton-Raphson step and the step in the steepest descent direction. The predicted step with length  $R$  is assumed to be on the straight line joining the Newton point (NP) and the Cauchy point (CP) and is calculated by

$$\bar{dx} = (\theta-1)u^*g(\bar{x}) + \theta\bar{\delta} \quad (4.4.5)$$

$$\text{where } 0 \leq \theta \leq 1$$

In order to find the predicted step to be equal to length  $R$ , the right hand side (RHS) terms of eq.(4.4.5) are bound to be

$$\|(\theta-1)u^*g(\bar{x}) + \theta\bar{\delta}\| = R \quad (4.4.6)$$

By straightforward algebra,  $\theta$  from eq.(4.4.6) can be calculated by the following equations.

$$\theta = \frac{b + \sqrt{b^2 - ac}}{a} \quad (4.4.7)$$

where

$$\begin{aligned} a &= u^{*2} \|\bar{g}(\bar{x})\|^2 + 2u^* \sum_i g_i(\bar{x}) \delta_i + \|\bar{\delta}\|^2 \\ b &= u^* \sum_i g_i(\bar{x}) \delta_i + u^{*2} \|\bar{g}(\bar{x})\|^2 \\ c &= u^{*2} \|\bar{g}(\bar{x})\|^2 - R^2 \end{aligned} \quad (4.4.8)$$

#### 4.4.3 THE METHOD FOR REVISING THE TRUST RADIUS R

The trust radius  $R$  can be revised for every iteration or even during the same iteration [1]. Usually, the trust radius  $R$  is preferably adjusted so that it is as large as possible to decrease the sum of the square of the mismatch  $F(\bar{x})$  for every iteration. This depends on a good prediction of the mismatch difference  $f_i(\bar{x} + \bar{d}\bar{x}) - f_i(\bar{x})$ ;  $i=1,2,\dots,n$ , without taking an extra small step. However, the trust radius  $R$  can also be reduced if the length  $R^k$ , at the turning point;  $\bar{x}^k$ , is so big that the iteration can not decrease the sum of the square of the mismatch  $F(\bar{x} + \bar{d}\bar{x})$  to be less than the old one.

To revise the trust radius  $R$ , at the end of the iteration, the test in eq.(4.4.9) is made.

$$F(\bar{x}^k + \bar{d}\bar{x}) < F(\bar{x}^k) \quad (4.4.9)$$

If the condition in eq.(4.4.9) fails, the iteration number is not increased to  $k+1$  and the trust radius  $R$  is reduced to be

$$R^k = \frac{1}{2}R^k \quad (4.4.10)$$

The iteration is repeated to calculate the predicted step  $\bar{dx}$  until the condition in eq.(4.4.9) holds and then prepare to increase the trust radius  $R$ .

The increase of the trust radius  $R$  can be provided due to the following factors

1. A good prediction of the mismatch difference  $\{f_i(\bar{x}+\bar{dx})-f_i(\bar{x})\}$  ;  $i=1,2,\dots,N$
2. The linearity of the nonlinear function  $f_i(\bar{x})$  ; between the turning point  $\bar{x}^k$  and  $\bar{x}^k+\bar{dx}$ . That is approximated to be

$$\phi_i = f_i(\bar{x}+\bar{dx}) - f_i(\bar{x}) = f_i(\bar{x}) + \sum_j J_{ij} dx_j \approx f_i(\bar{x}+\bar{dx}) \quad (4.4.11)$$

$$\text{and } \phi = \sum_i \phi_i^2 \approx F(\bar{x}+\bar{dx})$$

The basis of the method for increasing the trust radius  $R$  is that the mismatch difference  $f_i(\bar{x}+\bar{dx}) - \phi_i$  is attributed to terms that are of the order of  $R^2$ . If the trust radius  $R$  is multiplied by the factor  $\Omega$ , then the mismatch difference is also expected to be multiplied by about  $\Omega^2$ . Guided by this assumption, the multiplier  $\Omega$  can be calculated by bounding

$$\sum_i |f_i(\bar{x} + d\bar{x})| + (\Omega^2 - 1) |f_i(\bar{x} + d\bar{x}) - \phi_i| = 0.9F(\bar{x}) - [F(\bar{x} + d\bar{x}) - 0.1\phi] \quad (4.4.12)$$

This yields

$$\Omega^2 = 1 + \frac{aa}{bb + \sqrt{bb^2 + aa cc}} \quad (4.4.13)$$

where  $aa = 0.9F(\bar{x}) - [F(\bar{x} + d\bar{x}) - 0.1\phi]$

$$bb = \sum_i |f_i(\bar{x} + d\bar{x}) [f_i(\bar{x} + d\bar{x}) - \phi_i]| \quad (4.4.14)$$

$$cc = \sum_i (f_i(\bar{x} + d\bar{x}) - \phi_i)^2$$

To avoid an oscillating value of the trust radius  $R$ , it is suggested not to scale  $R$  by  $\Omega$  directly whenever  $R$  is calculated. The reason for this is that, in cases the trust radius  $R$  is reduced in a previous iteration, multiplying the reduced  $R$  by  $\Omega$  would restore the trust radius  $R$  to about its original value. When two values of  $\Omega$  have been calculated, they must both have been obtained since the last reduction in  $R$ . The factor by which  $R$  is multiplied is set equal to the lesser calculated value of  $\Omega$ . Moreover, the factor  $\Omega$  is limited being not greater than 2, and " $d_{\max}$ " is the upper bound of the trust radius  $R$ . To apply this strategy, a parameter  $\tau$  ( $\tau^0=1$ ) is introduced and set to the value one both before the first iteration and also whenever the trust radius is reduced. Thus, the trust radius  $R$  can be increased by the following equations:

$$R^{k+1} = \min(\beta^k R^k, d_{\max})$$

$$\text{where } \beta^k = \min(2, \Omega^k, \tau^k) \quad (4.4.15)$$

$$\tau^k = \frac{\Omega^{k-1}}{\beta^{k-1}}$$

In addition, if consecutive iterations obtain the full Newton-Raphson step, then the trust radius  $R$  is revised to be equal to the value of  $\|\bar{\delta}\|$ . The reason is that consecutive successful Newton-Raphson iterations tend to decrease in the length of predicted steps (due to the quadratic properties).

#### 4.4.4 SOME INDICATORS USED IN DETERMINING THE TRUST RADIUS $R$

In general, the nonlinear surfaces of nonlinear functions are approximately quadratic only in the immediate vicinity of the solution. The trust radius  $R$  has to be chosen with some thought of making the algorithm converge to the solution without taking extra iterations. An indicator, used to judge the appropriateness of the trust radius  $R$ , should be provided and is used to report major decisions.

Thus, the indicator, which is used to determine if the predicted point is close to a quadratic region of the nonlinear functions, is introduced. The idea is to compare the actual reduction of the sum of the square of the mismatch  $F(\bar{x})$  obtained with each step  $\bar{dx}$  to that which

is available from the same step in an ideal quadratic model (based on data from where the step began). The reduction of the sum of the mismatch is recalled to be  $F(\bar{x}^k) - F(\bar{x}')$ , where  $\bar{x}^k$  is the predicted value on every turning point and  $\bar{x}'$  is the solution point of the quadratic functions based on the gradient  $\bar{g}(\bar{x}^k)$  and the Newton-Raphson step ( $\bar{\delta}$ ). To find the quadratic factor, the quadratic form [2] is recalled to be

$$F(\bar{x}^k) - F(\bar{x}') = \frac{1}{2} \left[ \bar{g}(\bar{x}^k) \right]^T \left[ \bar{H}(\bar{x}^k) \right]^{-1} \left[ \bar{g}(\bar{x}^k) \right] \quad (4.4.16)$$

Substituting the Hessian matrix  $\bar{H} = \bar{J}^T \bar{J}$  into eq.(4.4.16), yields

$$F(\bar{x}^k) - F(\bar{x}') = \frac{1}{2} \left[ \bar{g}(\bar{x}^k) \right]^T \left[ \bar{\delta} \right] \quad (4.4.17)$$

Then, the quadratic factor  $r$  is defined to be the ratio

$$r = \frac{F(\bar{x}^k) - F(\bar{x}^k + \bar{\delta})}{F(\bar{x}^k) - F(\bar{x}')} \quad (4.4.18)$$

where the denominator of eq.(4.4.18) is calculated by eq.(4.4.17).

From eq.(4.4.18), the quadratic factor  $r \rightarrow 1$ , when the predicted values approach the solutions of the system



nonlinear equations. Thus, the decision of choosing the trust radius  $R$  can be determined by observing the behavior of the quadratic factor  $r$  from iteration to iteration.

#### **4.5 HYBRID ALGORITHM FOR NEWTON-RAPHSON LOADFLOW**

The proposed Hybrid algorithm can be easily incorporated into the existing Newton-Raphson power flow program. Some additional computer subroutines for the Hybrid algorithm, such as subroutine for calculating the predicted step in the steepest descent direction, subroutine that is used to perform the switching policies between the existing Newton-Raphson iteration and the steepest descent iteration, or even subroutine for revising the trust radius  $R$ , can be also easily provided to improve an efficiency in convergence of the existing Newton-Raphson power flow program. By wisely selecting the initial value of the trust radius  $R$ , extra iterations in the steepest descent direction can be avoided and the fast convergence of the standard Newton-Raphson method can be obtained by the proposed algorithm.

##### **4.5.1 TRUST RADIUS INITIALIZATION FOR THE NONLINEAR POWER FLOW PROBLEM**

The length of the trust radius  $R$  varies according to the size of the nonlinear equation system. If  $\bar{x}$  is the

vector of unknowns of the system of nonlinear algebraic equations with  $N$  equations, then the initial value of the trust radius  $R$  can be approximated by

$$\begin{aligned} R^0 &= \|\bar{dx}_{\max}\| & (4.5.1) \\ &= (dx^2_{1,\max} + dx^2_{2,\max} + \dots + dx^2_{n,\max})^{\frac{1}{2}} \end{aligned}$$

where  $dx_{i,\max}$  = maximum Newton-Raphson step allowed for unknown  $x_i$ ;  $i=1,2,\dots,N$  for the first iteration.

For the system of the nonlinear power flow equations, there are two types of unknowns. One is the voltage magnitudes ( $|v|$ ) on every load bus, and the other one is the voltage phase angles ( $\theta$ ) on every bus in the system (except the slack bus). From eq.(4.5.1), the trust radius  $R^0$  for the nonlinear power flow equations can be initialized by

$$R^0 = (m(dv_{\max})^2 + n(d\theta_{\max})^2)^{\frac{1}{2}} \quad (4.5.2)$$

where  $dv_{\max}$  = maximum Newton-Raphson step allowed for voltage magnitude at load buses for the first iteration

$d\theta_{\max}$  = maximum Newton-Raphson step allowed for voltage phase angles at all buses (except the slack bus) in the system for

the first iteration

$m$  = total number of load buses

$n$  = total number of system buses (except the  
slack bus)

#### 4.5.2 Digital steps of Hybrid algorithm for nonlinear power flow problem

**step 1** : Read the information required for a power flow solutions, such as Y-bus elements, generation and load data, etc.

**step 2** : Formulate the system of nonlinear equations for the real and reactive bus power mismatch.

**step 3** : Initialize all variables, such as unknown bus voltage magnitudes ( $|v^0|$ ), unknown bus voltage phase angles ( $\theta^0$ ) and trust radius  $R^0$ , etc.

**step 4** : Perform the standard Newton-Raphson iteration, calculate the Newton-Raphson step and obey the switching policies .

- if  $\|\bar{\delta}\| < R$ , then the iteration obtains the full Newton-Raphson step.

- if  $\|\bar{\delta}\| > R$  and  $u^*\bar{g}(\bar{x}) > R$ , then the iteration calculates the predicted step with length  $R$  in the steepest descent direction; otherwise, the predicted step is the step interpolated between

the Newton-Raphson step and the step in the steepest descent direction.

**step 5 :** Calculate  $F(\bar{x}+d\bar{x})$  and try the test  $F(\bar{x}+d\bar{x}) < F(\bar{x})$

- If  $F(\bar{x}+d\bar{x}) > F(\bar{x})$ , then reduce the trust radius  $R$  and go to step 4.

- If  $F(\bar{x}+d\bar{x}) < F(\bar{x})$ , then increase the trust radius  $R$  and prepare for the next iteration.

**step 6 :** If the mismatch  $F(\bar{x}+d\bar{x})$  is less than the prespecified tolerance, then stop the iteration and calculate all bus powers and line flows; otherwise, count to the next iteration and go to step 4.

The iteration might be stopped in the case when  $\|\bar{g}(\bar{x}^k)\| = 0$  but  $F(\bar{x}^k) \neq 0$ . The reason is that the iteration has approached a local minimum (not a global minimum) [2]. This point is not the solution. Therefore, a new set of initial values is tried. FIG 4.7 shows the computer flow diagram of the Hybrid algorithm.

#### 4.6 COMPUTER SUBROUTINE FOR THE HYBRID ALGORITHM APPLIED TO THE NEWTON-RAPHSON LOADFLOW

The listing of the computer program, HYBRID, is contained in APPENDIX A. The program is written in BASIC language by using the Qbasic compiler. The machine

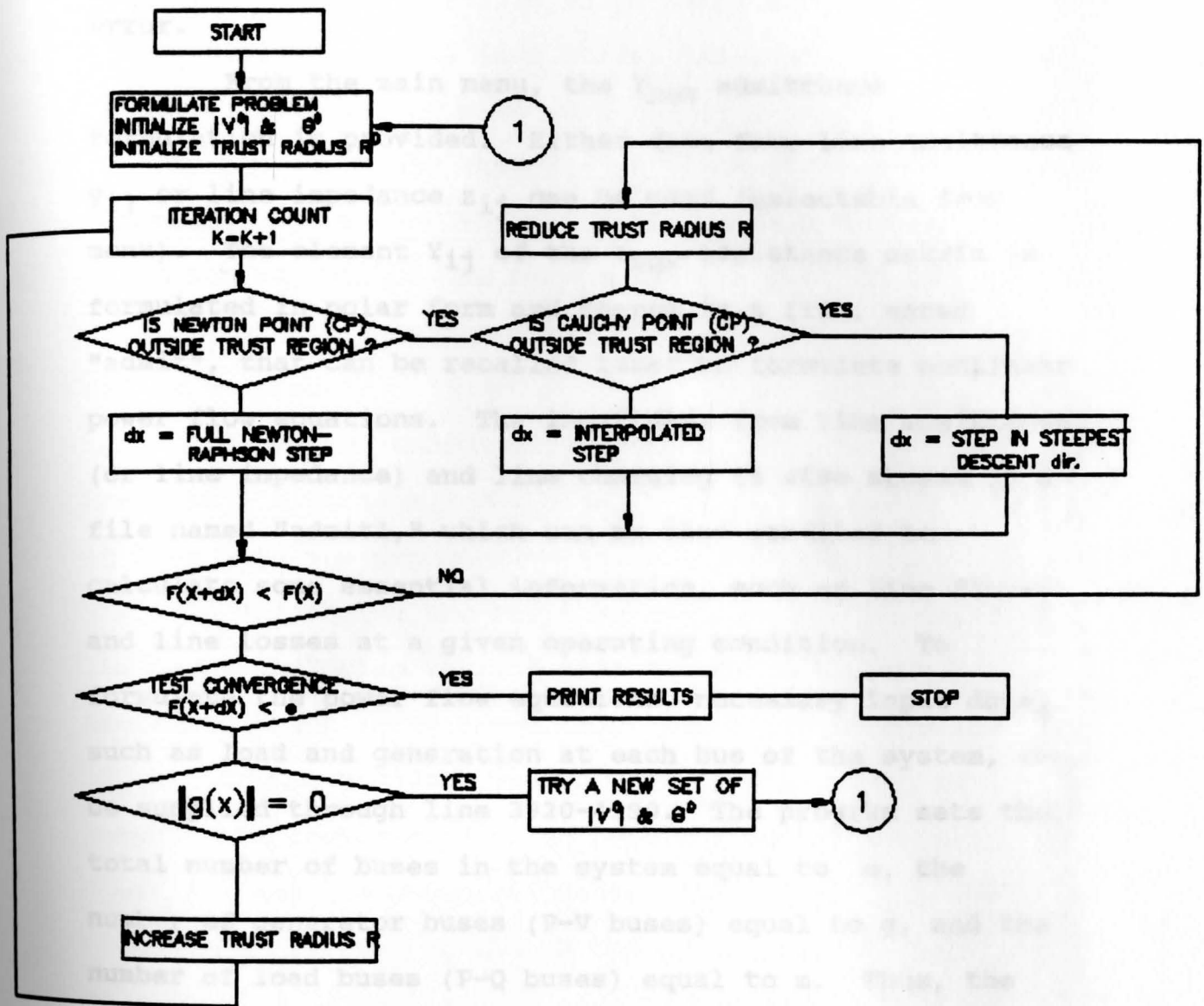


FIG 4.7 Computer flow diagram of the HYBRID algorithm applied to the NEWTON-RAPHSON loadflow

precision is set to be into double precision mode by the program to reduce the effect of the computer rounding error.

From the main menu, the  $Y_{bus}$  admittance formulation is provided. Either data from line admittance  $y_{ij}$  or line impedance  $z_{ij}$  can be used (selectable from menu). The element  $Y_{ij}$  of the  $Y_{bus}$  admittance matrix is formulated in polar form and stored in a file, named "admit", that can be recalled later to formulate nonlinear power flow equations. The input data from line admittance (or line impedance) and line charging is also stored in a file named "admit1," which can be also recalled to calculate some essential information, such as line flows and line losses at a given operating condition. To formulate the power flow equations, necessary input data, such as load and generation at each bus of the system, can be supplied through line 3920-4190. The program sets the total number of buses in the system equal to  $n$ , the number of generator buses (P-V buses) equal to  $g$ , and the number of load buses (P-Q buses) equal to  $m$ . Thus, the  $2n-g-1$  nonlinear power flow equations are formulated by the program (2 equations for each P-Q bus, and 1 equation for each P-V bus). The program requires the first bus to be the slack bus, from bus 2 through bus  $g$  to be P-V buses, and from bus  $g+1$  to bus  $m$  to be P-Q buses, respectively. The input data required by the program are

voltage magnitude and the real power generation for all generator buses (except the slack bus), and real and reactive loads for all load buses. However, the real and reactive loads can be also supplied at generator buses. In order to start the iteration, the initial values of all unknowns, such as bus voltage magnitude and bus voltage phase angles, can be set to desired values from line 4420-4310. (default values are set to be equal to  $|V^0| = 1$  p.u and  $\theta^0 = 0$  rad.)

Line 6000 through line 7500 is written to perform the Hybrid algorithm. The trust radius  $R$  is initialized in line 6200 and can be changed to a desired value. The variable  $F$  contains the value of the sum of the square of the power mismatch while a variable  $F1$  contains the preceding value. The elements of the Jacobian matrix  $A(i,j)$  and the gradient vector  $GD(i)$  are provided to compute the predicted step in both the Newton-Raphson iteration and the steepest descent iteration. Line 6660 checks the location of the Newton point (NP) at each iteration, by comparing the norm of the Newton-Raphson step to the length of the trust radius  $R$ , while line 6830 checks the location of the Cauchy point (CP), by comparing the value  $\|u^* \bar{g}\|$ , calculated from line 6730-6830, to the length of the trust radius  $R$ . The subroutine for revising the trust radius  $R$  is provided by the program from line 7160-7460. The trust radius  $R$  will be reduced if the

condition  $F > F_1$  in line holds; otherwise, it will be increased. Line 7480 checks the convergence. If the sum of the square of the power mismatch  $F$  is less than the prespecified tolerance  $E$ , then the program stops the iteration. Line 11000-11650 is provided to calculate the results, such as line flows, line losses, and some essential information for load flow studies. The list of subroutines for the program HYBRID is shown in TABLE 4.2.

Compute NEWTON-RAPHSON step ( $\delta$ ) and its norm $ \delta $	6000-6050
Compute step biased in steepest direction and its norm	6100-6150
Revise trust radius $R$	6200-6250
Test convergence	6300-6350
Compute the sum of the square of the power mismatch	6400-6450
Compute power mismatch	6500-6550
Compute gradient $\nabla(F)$ and its norm $ \nabla(F) $	6600-6650
Compute Jacobian matrix	6700-6750
Compute inverse matrix	6800-6850
Compute line flows, line losses and line current	11000-11650
Print out results	14000-15100

TABLE 4.2 List of computer subroutines for the hybrid algorithm applied to NEWTON-RAPHSON algorithm



NAME	LINE
Formulate $Y_{bus}$ admittance matrix	1000-2780
Input data for loadflow calculation	3780-4230
Initialize all unknown $ v^0 $ and $\theta^0$	4240-4310
Initialize all variables for HYBRID algorithm, such as Trust radius $R^0$ and tolerance limit $E$	6170-6220
Compute NEWTON-RAPHSON step ( $\bar{\delta}$ ) and its norm $\ \bar{\delta}\ $	6430-6600
Compute step biased in steepest direction and its norm	6720-7150
Revise Trust radius $R$	7160-7460
Test convergence	7470-7490
Compute the sum of the square of the power mismatch	7510-7660
Compute power mismatch	8000-8190
Compute gradient $\bar{g}(\bar{x})$ and its norm $\ \bar{g}(\bar{x})\ $	6320-6410
Compute Jacobian matrix	9000-9590
Compute inverse matrix	10000-10640
Compute line flows, line losses and line current	11000-11650
Print out results	14000-15140

TABLE 4.2 List of computer subroutines for the HYBRID algorithm applied to NEWTON-RAPHSON loadflow

## Chapter V

### EXAMPLE AND NUMERICAL RESULTS

#### 5.1 INTRODUCTION

In this chapter, the 10 bus power system is chosen to be investigated. Digital computer results of the test system, obtained by the HYBRID algorithm and the standard NEWTON-RAPHSON method, along with the effect in choosing the different values of  $R^0$  are given to be discussed.

#### 5.2 EXAMPLE : 10 BUS TEST SYSTEM

One line diagram of the 10 bus power system is shown in Fig 5.1. The system data, such as line admittance data, load data and generation data are given in APPENDIX B. The system consists of 8 load buses (P-Q buses), two generators (one of which performed as the slack bus) and 13 lines. Therefore, there are 17 nonlinear power flow equations obtained for this system. (2 equations for each P-Q bus and one for each P-V bus)

#### 5.3 RESULTS

To investigate the effect of initial values on the convergence of both the HYBRID algorithm and the standard NEWTON-RAPHSON method, 3 different sets of initial values are given to be:

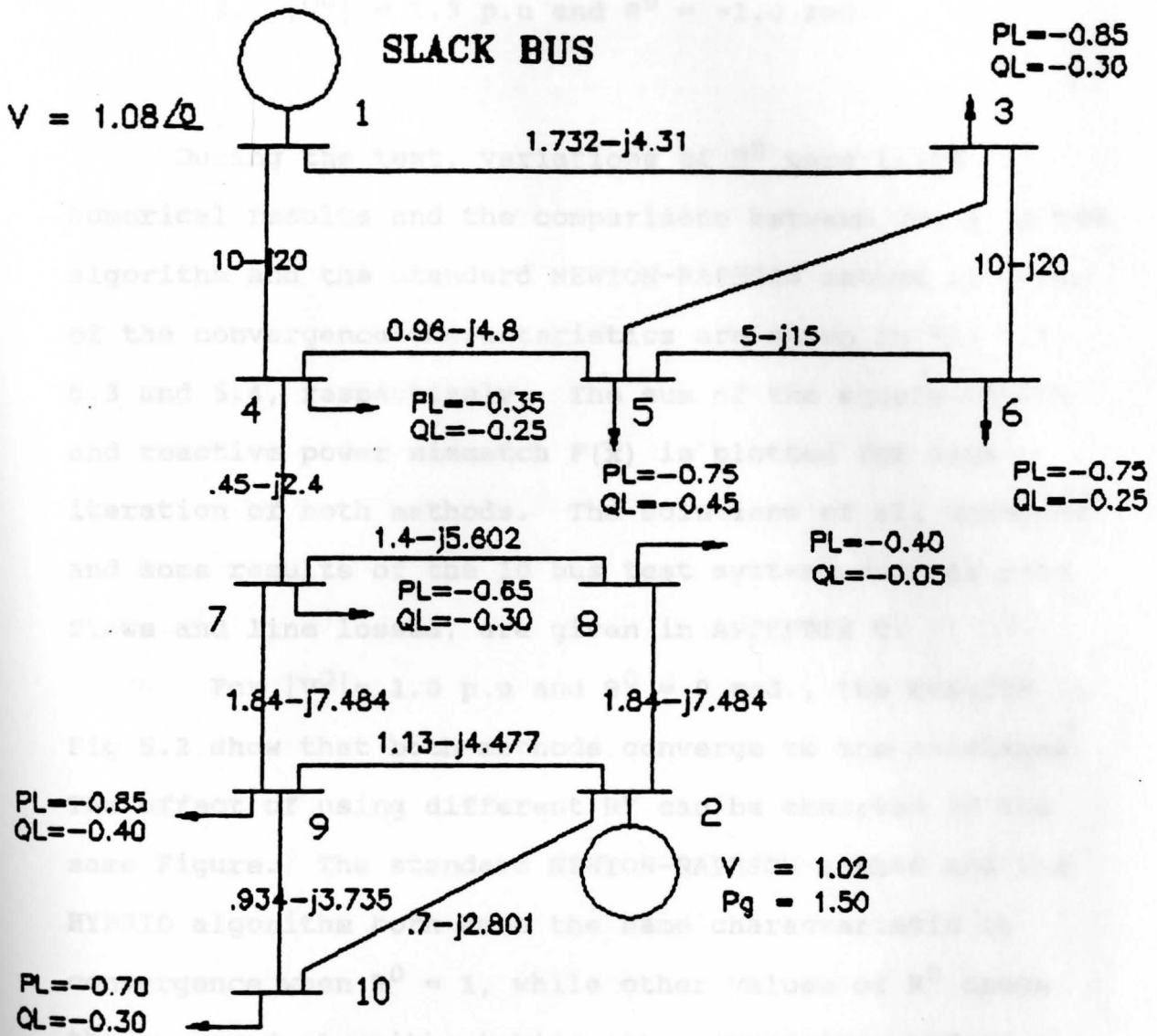


FIG 5.1 One Line Diagram of 10 bus tested system

1.  $|v^0| = 1.0$  p.u and  $\theta^0 = 0.0$  rad.
2.  $|v^0| = 0.8$  p.u and  $\theta^0 = -0.5$  rad.
3.  $|v^0| = 1.3$  p.u and  $\theta^0 = -1.0$  rad.

During the test, variations of  $R^0$  were tried.

Numerical results and the comparisons between the proposed algorithm and the standard NEWTON-RAPHSON method in terms of the convergence characteristics are shown in Fig 5.2, 5.3 and 5.4, respectively. The sum of the square of real and reactive power mismatch  $F(\bar{x})$  is plotted for each iteration of both methods. The solutions of all unknowns and some results of the 10 bus test system, such as line flows and line losses, are given in APPENDIX C.

For  $|v^0| = 1.0$  p.u and  $\theta^0 = 0$  rad., the results in Fig 5.2 show that both methods converge to the solutions. The effect of using different  $R^0$  can be observed in the same Figure. The standard NEWTON-RAPHSON method and the HYBRID algorithm both have the same characteristic in convergence when  $R^0 = 1$ , while other values of  $R^0$  cause the proposed algorithm taking extra iterations before the solutions are approached. The switching status (0=full NEWTON-RAPHSON step, 1=full steepest descent step and 2=interpolated step), shown in TABLE 5.1, indicates that the predicted NEWTON-RAPHSON step is inside the trust region on every iteration when  $R^0=1$ , while the small values of  $R^0$  ( $R^0=0.1$ ,  $R^0=0.3$  and  $R^0=0.5$ ) caused the

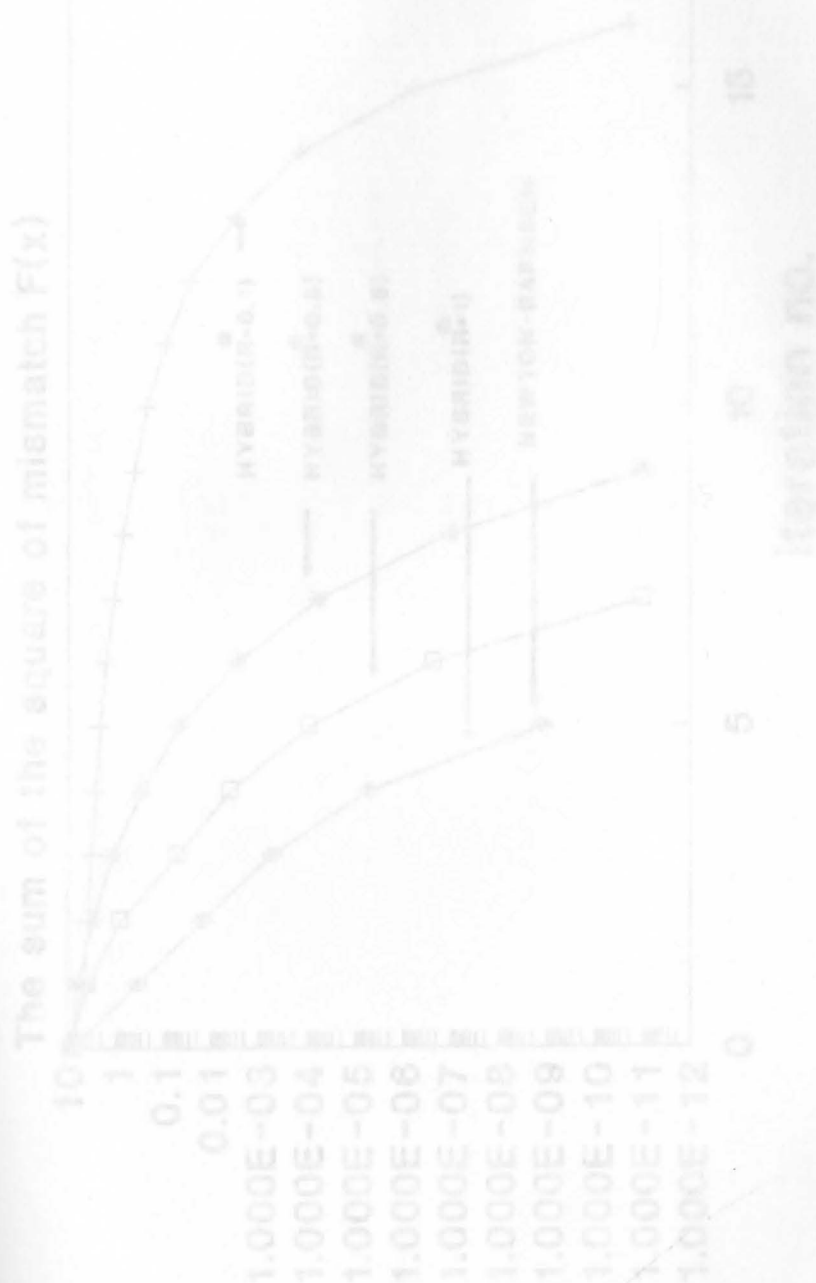
predicted step to be biased in the direction of the negative gradient  $\bar{g}(\bar{x})$  (steepest descent direction) before the full NEWTON-RAPHSON step can be obtained.

For  $|v^0| = 0.8$  p.u and  $\theta^0 = -0.5$  rad., the results in Fig 5.3 show that the proposed algorithm converges to the solutions of the system faster than those by the standard NEWTON-RAPHSON method. The reduction of the mismatch  $F(\bar{x})$  is oscillatory on the first iteration, because of the effect of truncation error before it converged to the solutions, while the predicted step on the same iteration by the HYBRID algorithm is biased to be in the steepest descent direction to reduce  $F(\bar{x} + d\bar{x}) < F(\bar{x})$  before it obtained the full NEWTON-RAPHSON step on the second iteration (shown in TABLE 5.2).

For  $|v^0| = 1.3$  p.u and  $\theta^0 = -1.0$  rad., the results in Fig. 5.4 does not show any convergence by the standard NEWTON-RAPHSON method. The characteristic of the mismatch  $F(\bar{x})$  is oscillatory in nature and it does not approach the solutions of the system. By comparison, the HYBRID algorithm, with  $R^0=2$ , successfully converged to the solutions with the fewest iterations.

In order to select the appropriate value for  $R^0$ , the behavior of the quadratic factor  $r$ , that is favorable to be  $R \rightarrow 1$ , can be observed. The results, in Fig 5.5, 5.6 and 5.7, show that the values of quadratic factor  $r$  for the optimum  $R^0$  ( $R^0=1$ ,  $R^0=1$  and  $R^0=2$ ), are all in the

neighborhood of 1 from the beginning of the iteration. Compared to those by the other values of  $R^0$ , a few more iterations are required before they approached the neighborhoods of the solutions ( $R \rightarrow 1$ ).



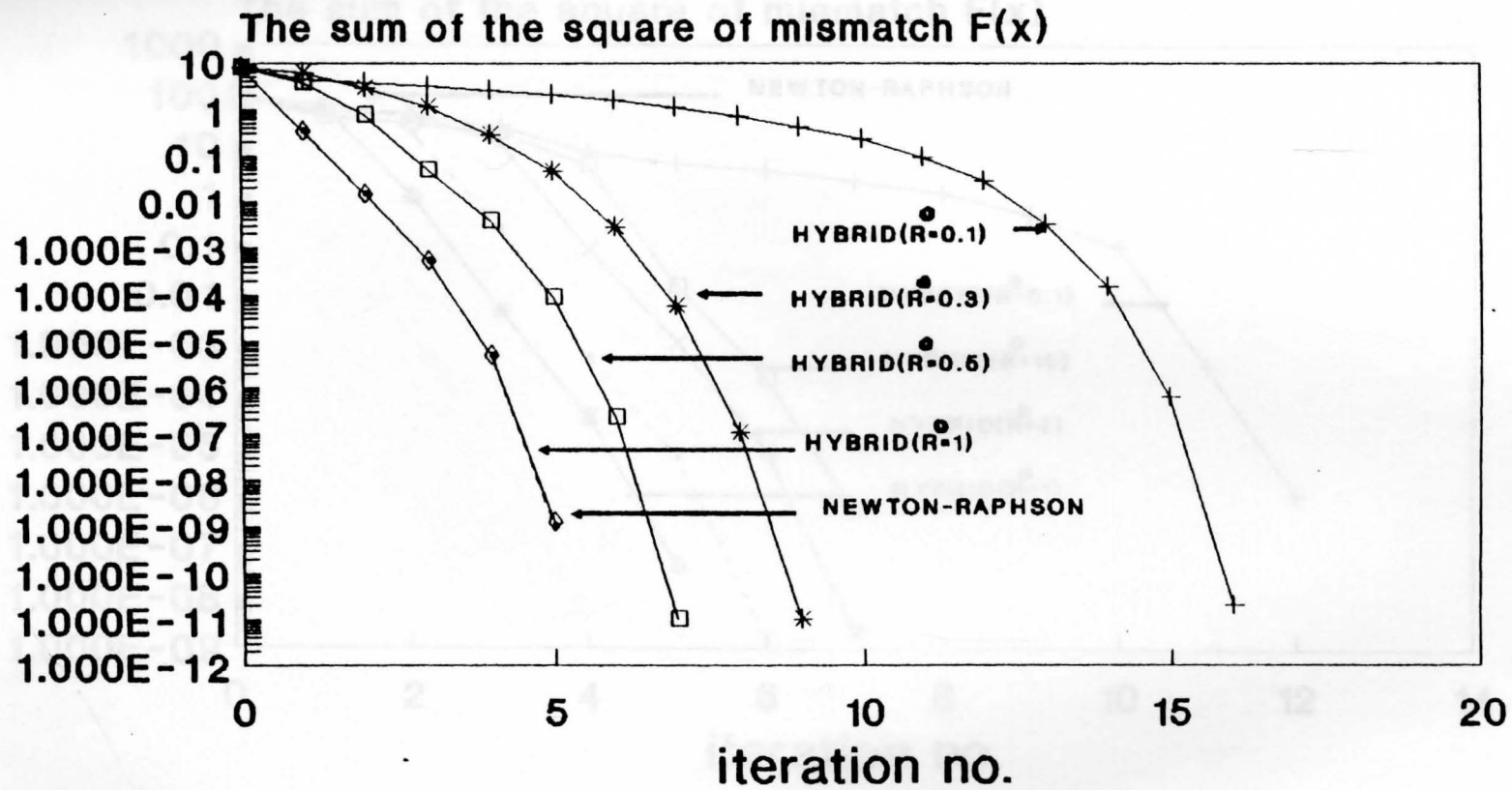


FIG 5.2 The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when  $|V^0| = 1.0$  p.u and  $\theta^0 = 0.0$  rad.

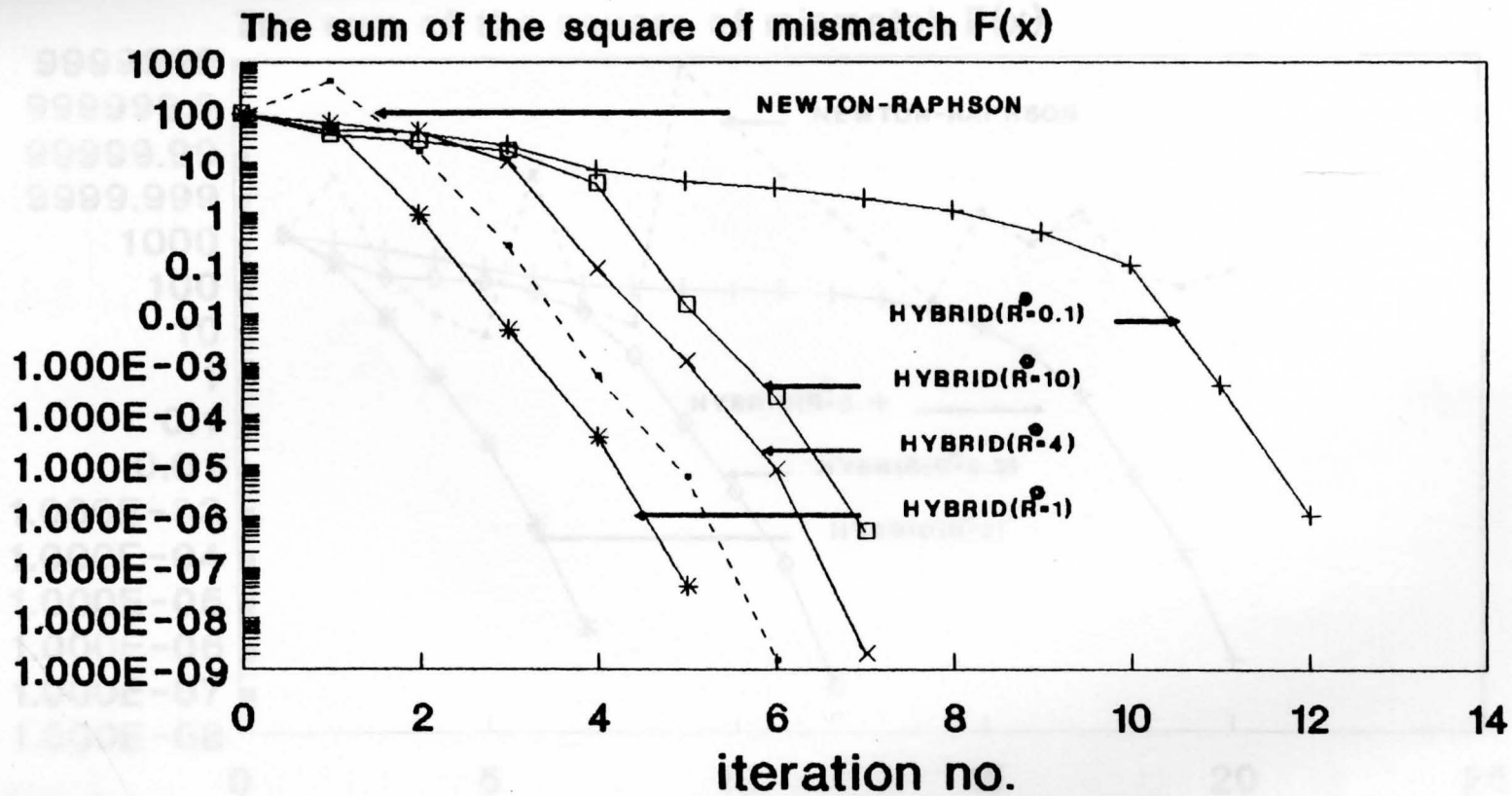


FIG 5.3 The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when  $|v^0| = 0.8$  p.u and  $\theta^0 = -0.5$  rad.



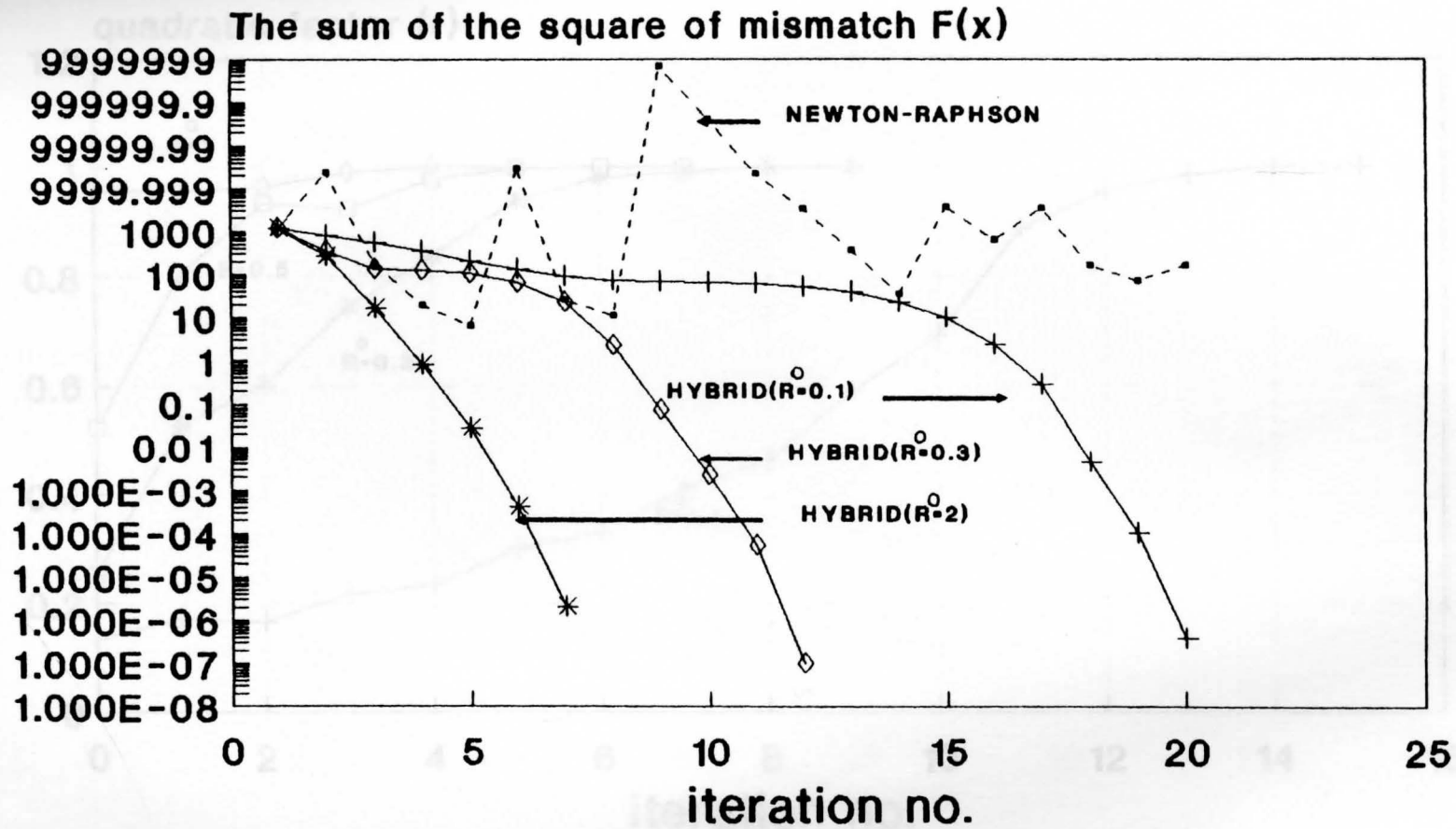


FIG 5.4 The convergence characteristic comparison between the HYBRID algorithm and the NEWTON-RAPHSON method when  $|V^0| = 1.3$  p.u and  $\theta^0 = -1.0$  rad.

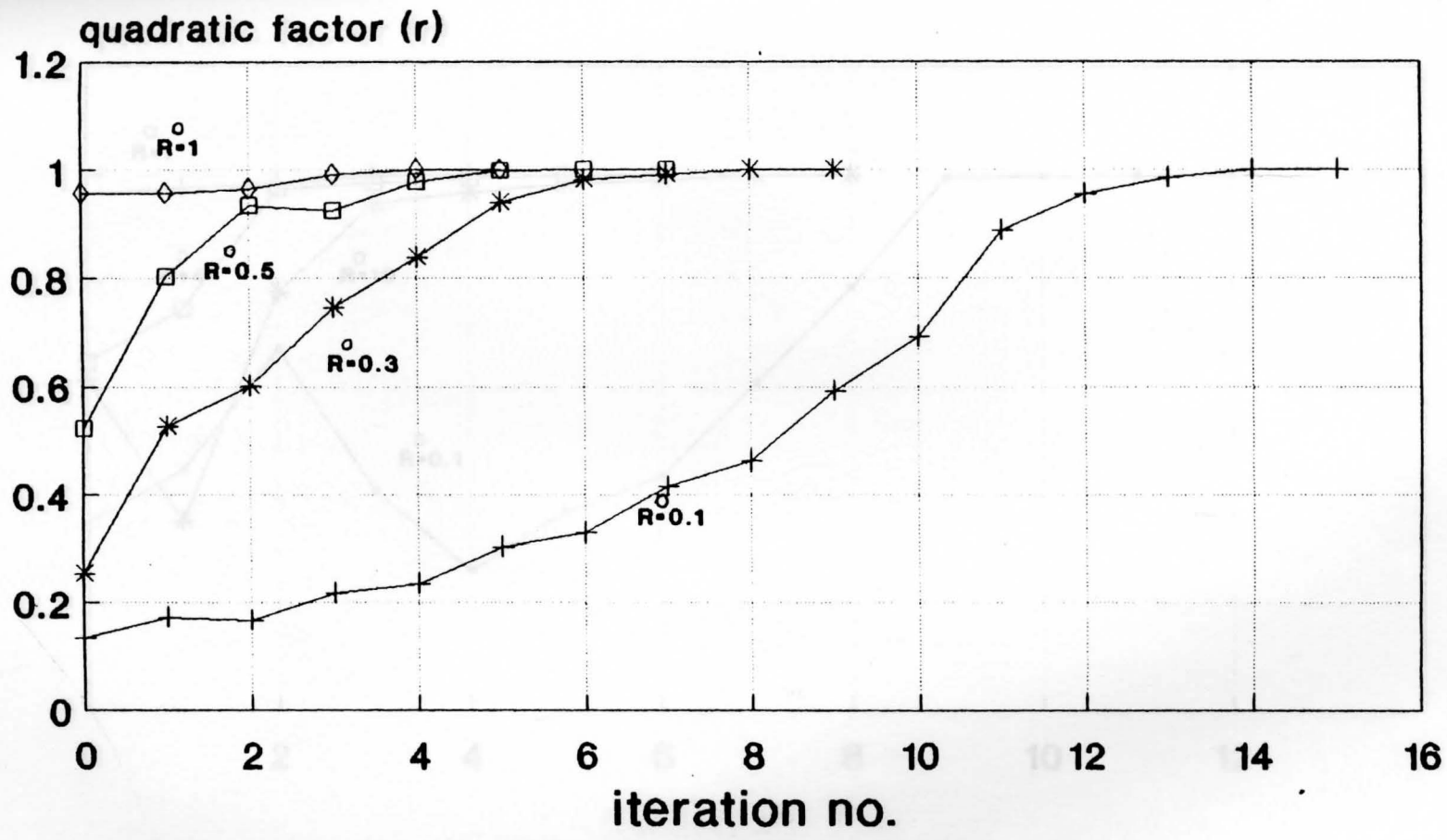


FIG 5.5 The characteristics of quadratic factor  $r$  for different values of  $R^0$  when  $|V^0| = 1.0$  p.u and  $\theta^0 = 0.0$  rad.

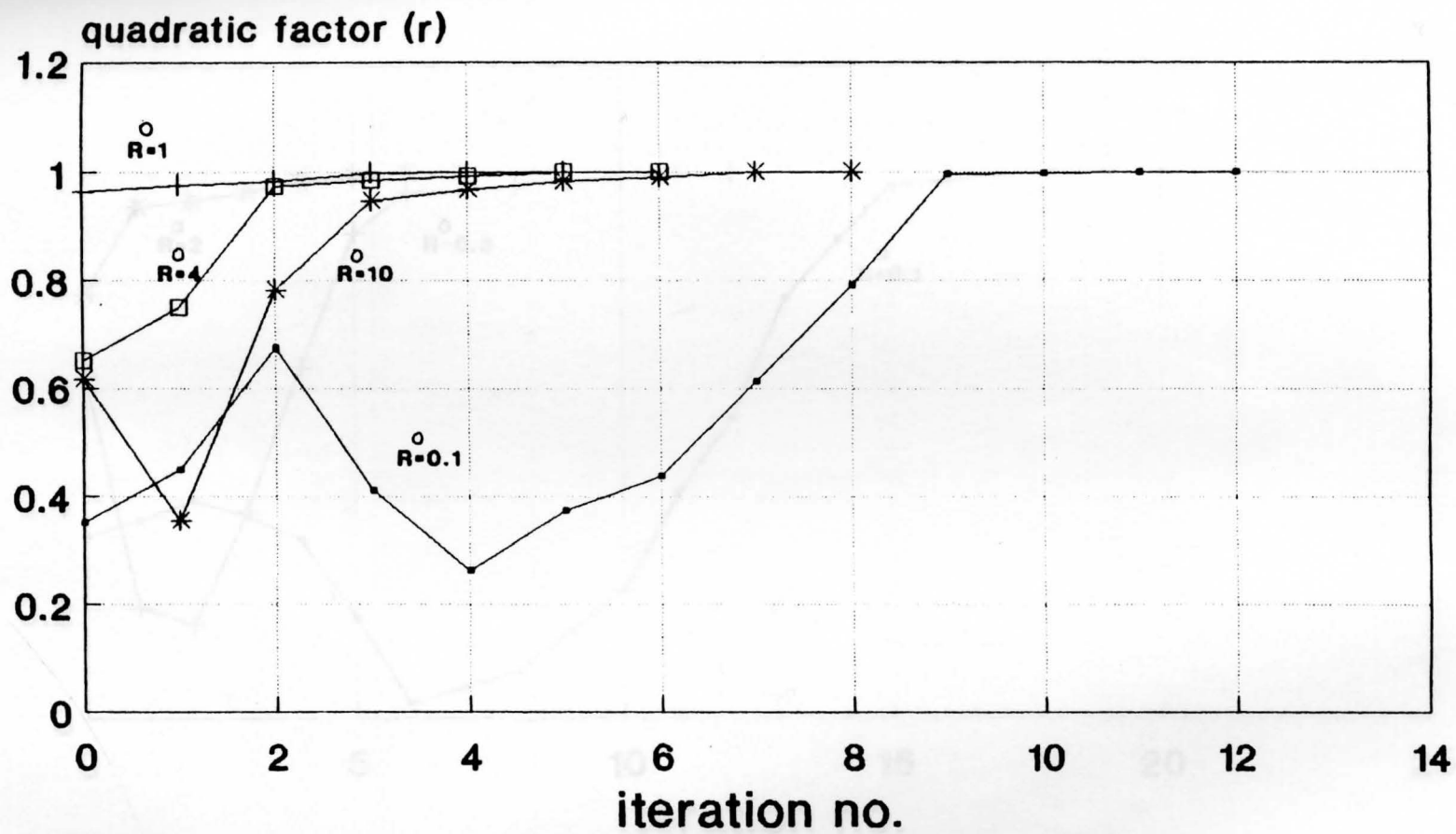


FIG 5.6 The characteristics of quadratic factor  $r$  for different values of  $R^0$  when  $|v^0| = 0.8$  p.u and  $\theta^0 = -0.5$  rad.

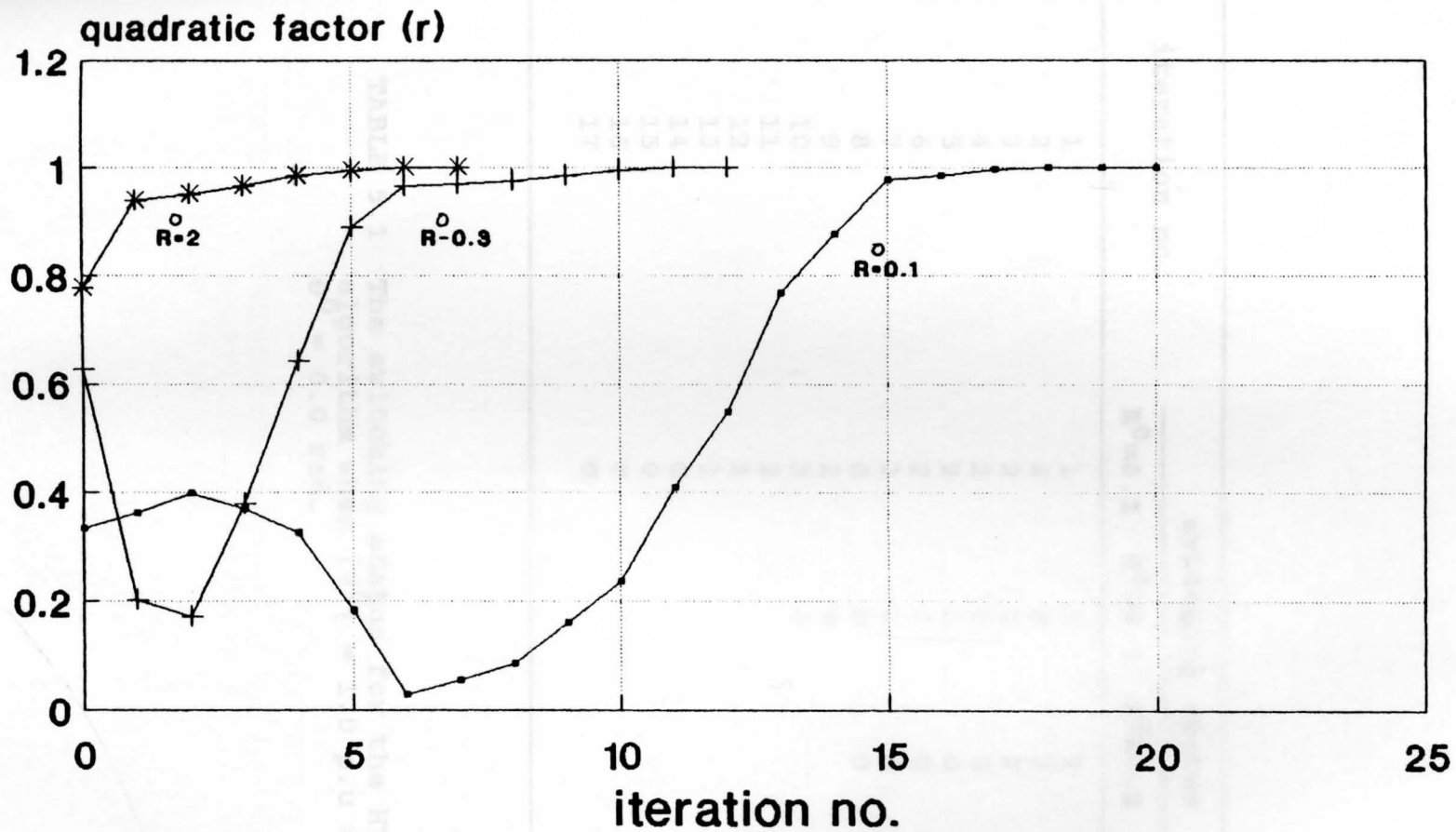


FIG 5.7 The characteristic of quadratic factor  $r$  for different values of  $R^0$  when  $|V_0| = 1.3$  p.u and  $\theta^0 = -1.0$  rad.

iteration no.	switching status			
	$R^0=0.1$	$R^0=0.3$	$R^0=0.5$	$R^0=1$
1	1	2	2	2
2	2	2	2	0
3	2	2	2	0
4	2	2	0	0
5	2	2	0	0
6	2	0	0	0
7	2	0	0	
8	2	0	0	
9	2	0		
10	2	0		
11	2			
12	2			
13	2			
14	0			
15	0			
16	0			
17	0			

TABLE 5.1 The switching status for the HYBRID algorithm when  $|V^0| = 1.0$  p.u and  $\theta^0 = 0.0$  rad.

TABLE 5.1 The switching status for the HYBRID algorithm when  $|V^0| = 0.3$  p.u and  $\theta^0 = -0.5$  rad.

iteration no.	switching status			
	$R^0=0.1$	$R^0=1$	$R^0=4$	$R^0=10$
1	2	2	2	1
2	2	0	2	2
3	2	0	2	2
4	0	0	2	2
5	2	0	0	2
6	0	0	0	0
7	0		0	0
8	0		0	0
9	0		0	0
10	0		0	
11	0		0	
12	0		0	
13	0			
14	0			
15	0			
16	0			
17	0			
18	0			
19	0			

TABLE 5.1 The switching status for the HYBRID algorithm when  $|V^0| = 0.8$  p.u and  $\theta^0 = -0.5$  rad.

## Chapter VI

## CONCLUSION

## 5.3 SUMMARY

The main contribution of this thesis is the

iteration no.	switching status		
	$R^0=0.1$	$R^0=0.3$	$R^0=2$
1	1	1	2
2	1	1	0
3	1	2	0
4	1	2	0
5	1	2	0
6	1	2	0
7	1	2	0
8	2	2	0
9	2	0	0
10	2	0	0
11	2	0	0
12	2	0	0
13	2	0	0
14	2	0	0
15	2	0	0
16	2	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0
21	0	0	0

TABLE 5.3 The switching status for the HYBRID algorithm when  $|V^0| = 1.3$  p.u and  $\theta^0 = -1.0$  rad.

## Chapter VI

### CONCLUSION

#### 6.1 SUMMARY

The main contribution of this thesis is the development of the HYBRID algorithm in order to improve the efficiency in convergence of the standard NEWTON-RAPHSON method in solving nonlinear a power flow problem, when its close initial values are not available. A defined truncation error from the neglected Taylor's series terms of the standard NEWTON-RAPHSON method can arise and cause the divergence. By introducing the method of steepest descent and the concept of the "trust region", the switching policies can be performed. The idea is to start the iteration with the method of steepest descent, if it is necessary, and then switch to the NEWTON-RAPHSON method which is better when the solutions are approached.

In this thesis, the HYBRID algorithm has been applied to the standard NEWTON-RAPHSON loadflow using bus admittance matrix. The results of the 10 bus test system by using the proposed algorithm, compared with those by using the standard NEWTON-RAPHSON method, are quite favorable when the different sets of initial values were tried. By observing the characteristic of quadratic factor, the appropriate values of  $R^0$  can be obtained to avoid a slow convergence.



## 6.2 RECOMMENDATIONS FOR FUTURE WORK

To improve the performance of the Hybrid algorithm developed in this thesis, the following suggestions are made:

1. It would be valuable to extend the proposed algorithm to the fast decouple loadflow method that is widely used on a large scale power system. In general, the fast decouple loadflow obtains the solutions by applying approximations to the NEWTON-RAPHSON method.
2. The method, that can be used to reduce the effect of the computer rounding errors and also easily incorporated to the proposed algorithm, should be provided in order to improve the performance of the HYBRID algorithm. The effect of the defined computer rounding errors can arise and cause the divergence to the iterative methods if the power system is considered to be an ill-conditioned system

[15].

## APPENDIX A

Basic Program for HYBRID algorithm applied to  
the standard NEWTON-RAPHSON load flow

```

10 *****
20 *****
30 ***----- PROGRAM " HYBRID " -----**
40 *****
100 *****
110 *****-----Create main menu screen
120 *****
130 DEFDBL A-H, M-W          ' specify variables to be in double precision
140 CLEAR
150 SCREEN , , 0, 0
160 Z1$ = "          "
170 Z2$ = "          "
180 Z3$ = STRING$(57, CHR$(177))
190 Z4$ = STRING$(57, " ")
200 Z5$ = "          Loadflow Studies For Power System Network          "
210 Z6$ = "          (1) Create bus admittance matrix                      "
220 Z7$ = "              1). From line inpedance < zij >                  "
230 Z8$ = "              2). From line admittance < yij >                  "
240 Z9$ = "          (2) Calculating loadflow problem by                  "
250 Z10$ = "              1). Standard Newton-Raphson method              "
260 Z11$ = "              2). Applying Hybrid algoritm to standard          "
270 Z12$ = "              Newton-Raphson method.                          "
280 Z13$ = "          (3) Quit                                           "
290 KEY OFF
300 COLOR 14, 9
310 CLS
320 LOCATE 2, 1
330 PRINT Z1$ + Z3$ + Z2$
340 PRINT Z1$ + Z3$ + Z2$
350 PRINT Z1$ + Z4$ + Z2$
360 PRINT Z1$ + Z5$ + Z2$
370 PRINT Z1$ + Z4$ + Z2$
380 PRINT Z1$ + Z4$ + Z2$
390 PRINT Z1$ + Z6$ + Z2$
400 PRINT Z1$ + Z7$ + Z2$
410 PRINT Z1$ + Z8$ + Z2$
420 PRINT Z1$ + Z4$ + Z2$
430 PRINT Z1$ + Z9$ + Z2$

```

```

440 PRINT Z1$ + Z10$ + Z2$
450 PRINT Z1$ + Z11$ + Z2$
460 PRINT Z1$ + Z12$ + Z2$
470 PRINT Z1$ + Z4$ + Z2$
480 PRINT Z1$ + Z13$ + Z2$
490 PRINT Z1$ + Z4$ + Z2$
500 PRINT Z1$ + Z3$ + Z2$
510 PRINT Z1$ + Z3$ + Z2$
520 PRINT
530 COLOR 12
540 A = CSRLIN
550 LOCATE A: PRINT TAB(25); "make a choice between 1, 2 and 3 .....< ? >";
560 K$ = INPUT$(1)
570 KI = VAL(K$)
580 IF (KI = 1) OR ((KI = 2) OR (KI = 3)) THEN GOTO 590 ELSE GOTO 610
590 ON KI GOSUB 1000, 3000
600 GOTO 140
610 COLOR 28
620 SOUND 50, 1
630 SOUND 400, 1
640 SOUND 1000, 1
650 GOTO 550
1000 *****
1010 !*-----CREATE BUS ADMITANCE MATRIX-----*
1020 *****
1030 ' g = # of generator buses (P-V buses)
1040 ' m = # of load buses (P-Q buses)
1050 ' n = # of buses in the system
1060 ' G(i,j) + jB(i,j) = element of line admittance in rectangular form
1070 ' Y(i,j) & DEL(i,j) = element of bus admittance matrix (Ybus) in Polar form
1080 ' II(i,j) & EE(i,j) = element of line admittance in Polar form
1090 ' r + jx = line impedance
1100 *****
1110 COLOR 3
1120 LOCATE 7
1130 PRINT Z1$ + Z6$ + Z2$
1140 PRINT Z1$ + Z7$ + Z2$
1150 PRINT Z1$ + Z8$ + Z2$
1160 FOR I = 1 TO 80
1170 LOCATE A, I: PRINT " ";
1180 LOCATE A + 1, I: PRINT " ";
1190 NEXT I
1200 LOCATE A
1210 PRINT TAB(25); "make a choice between (1) and (2) .....< ? >";
1220 K1$ = INPUT$(1)
1230 K1 = VAL(K1$)
1240 IF (K1 = 1) OR (K1 = 2) THEN GOTO 1300
1250 COLOR 19
1260 SOUND 50, 1
1270 SOUND 400, 1
1280 SOUND 1000, 1
1290 GOTO 1200

```

```

1300 COLOR 23, 4
1310 LOCATE 3, 34
1320 PRINT " please wait .. "
1330 SCREEN , , 1, 0
1340 COLOR 14, 4
1350 CLS
1360 Z20$ = CHR$(218)
1370 Z21$ = CHR$(191)
1380 Z22$ = STRING$(67, CHR$(196))
1390 Z23$ = CHR$(179)
1400 Z24$ = CHR$(192)
1410 Z25$ = CHR$(217)
1420 Z26$ = "          ** - Create Bus Admittance Matrix - **          "
1430 Z27$ = "          (by using line impedance zij)          "
1440 Z28$ = "          (by using line admittance yij)          "
1450 Z29$ = " Z = R+jX "
1460 Z30$ = " Y = G+jB "
1470 ON K1 GOTO 1480, 1510
1480 X1$ = Z27$
1490 X2$ = Z29$
1500 GOTO 1530
1510 X1$ = Z28$
1520 X2$ = Z30$
1530 COLOR 7
1540 FOR I = 1 TO 24
1550 PRINT STRING$(79, CHR$(176))
1560 NEXT I
1570 COLOR 14, 4
1580 LOCATE 2, 4: PRINT Z20$ + Z22$ + Z21$
1590 LOCATE , 4: PRINT Z23$ + Z26$ + Z23$
1600 LOCATE , 4: PRINT Z23$ + X1$ + Z23$
1610 LOCATE , 4: PRINT Z24$ + Z22$ + Z25$
1620 COLOR , 9
1630 LOCATE CSRLIN + 1, 10: PRINT Z20$ + STRING$(60, CHR$(196)) + Z21$
1640 FOR I = 8 TO 20
1650 LOCATE I, 10
1660 PRINT CHR$(179) + STRING$(60, " ") + CHR$(179)
1670 NEXT I
1680 LOCATE , 10: PRINT Z24$ + STRING$(60, CHR$(196)) + Z25$
1690 SCREEN , , 1, 1
1700 COLOR 14, 9
1710 LOCATE 8, 11: INPUT "# of generators (P-V buses) = ", G ' enter # of generator buses (P-V buses)
1720 LOCATE , 11: INPUT "# of load buses (P-Q buses) = ", M ' enter # of load buses (P-Q buses)
1730 N = M + G
1740 LOCATE , 11: PRINT " total # of buses          = "; N
1750 DIM Y(N, N), DEL(N, N)
1760 DIM G(N, N), B(N, N)
1770 DIM LL(N, N), EE(N, N)
1780 LOCATE , 11: PRINT STRING$(60, CHR$(196))
1790 COLOR 14, 4
1800 C = CSRLIN
1810 LOCATE , 11: PRINT " bus no. ";

```

```

1820 COLOR , 9: LOCATE , 25: PRINT CHR$(61) + CHR$(62); X2$
1830 FOR I = 1 TO N
1840 COLOR , 4
1850 LOCATE C, 11: PRINT " bus no. "; I
1860 FOR J = I TO N
1870 COLOR 22, 9
1880 LOCATE C + 2, 11
1890 PRINT STRING$(50, " ")
1900 LOCATE C + 2, 11
1910 PRINT CHR$(61) + CHR$(62);
1920 COLOR 14
1930 ON K1 GOTO 1940, 2050
1940 PRINT " Z("; I; ", "; J; ") = ";
1950 INPUT "", R ' enter system data from line impedance (in rectangular form)
1960 LOCATE CSRLIN - 1, 35
1970 INPUT "(+/-) j ", X
1980 IF (R = 0) AND (X = 0) THEN GOTO 1990 ELSE GOTO 2020
1990 G(I, J) = 0
2000 B(I, J) = 0
2010 GOTO 2100
2020 G(I, J) = R / (R ^ 2 + X ^ 2)
2030 B(I, J) = -X / (R ^ 2 + X ^ 2)
2040 GOTO 2100
2050 PRINT " Y("; I; ", "; J; ") =";
2060 INPUT " ", G(I, J) ' enter system data from line admittance (in rectangular form)
2070 LOCATE CSRLIN - 1, 35
2080 INPUT "(+/-) j ", B(I, J)
2090 GOTO 2100
2100 LL(I, J) = SQR(G(I, J) ^ 2 + B(I, J) ^ 2) ' formulate line admittance matrix in Polar form
2110 IF G(I, J) = 0 THEN GOTO 2130
2120 GOTO 2170
2130 IF B(I, J) > 0 THEN EE(I, J) = 3.141592654# / 2
2140 IF B(I, J) < 0 THEN EE(I, J) = -3.141592654# / 2
2150 IF B(I, J) = 0 THEN EE(I, J) = 0
2160 GOTO 2230
2170 IF G(I, J) > 0 THEN GOTO 2190
2180 IF G(I, J) < 0 THEN GOTO 2210
2190 EE(I, J) = ATN(B(I, J) / G(I, J))
2200 GOTO 2220
2210 EE(I, J) = ATN(B(I, J) / G(I, J)) + 3.141592654#
2220 IF I = J THEN GOTO 2250
2230 G(J, I) = G(I, J)
2240 B(J, I) = B(I, J)
2250 NEXT J
2260 NEXT I
2270 OPEN "o", #2, "admit1" ' file "admit1" contained elements of line admittance matrix (in Polar form)
2280 OPEN "o", #3, "admit2" ' file "admit2" contained elements of line admittance matrix (in rectangular form)
2290 FOR I = 1 TO N
2300 FOR J = I TO N
2310 WRITE #2, LL(I, J), EE(I, J)
2320 WRITE #3, G(I, J), B(I, J)
2330 NEXT J

```

```

2340 NEXT I
2350 CLOSE #2
2360 CLOSE #3
2370 FOR I = 1 TO N
2380 FOR J = 1 TO N
2390 IF I = J THEN GOTO 2420
2400 G(I, I) = G(I, I) + G(I, J)
2410 B(I, I) = B(I, I) + B(I, J)
2420 NEXT J
2430 NEXT I
2440 FOR I = 1 TO N
2450 FOR J = I TO N
2460 IF I = J THEN GOTO 2510
2470 G(I, J) = -G(I, J)
2480 G(J, I) = G(I, J)
2490 B(I, J) = -B(I, J)
2500 B(J, I) = B(I, J)
2510 NEXT J
2520 NEXT I
2530 FOR I = 1 TO N          ' formulate bus admittance matrix Ybus (in Polar form)
2540 FOR J = 1 TO N
2550 Y(I, J) = SQR(G(I, J) ^ 2 + B(I, J) ^ 2)
2560 IF G(I, J) = 0 THEN GOTO 2580
2570 GOTO 2620
2580 IF B(I, J) > 0 THEN DEL(I, J) = 3.141592654# / 2
2590 IF B(I, J) < 0 THEN DEL(I, J) = -3.141592654# / 2
2600 IF B(I, J) = 0 THEN DEL(I, J) = 0
2610 GOTO 2670
2620 IF G(I, J) > 0 THEN GOTO 2640
2630 IF G(I, J) < 0 THEN GOTO 2660
2640 DEL(I, J) = ATN(B(I, J) / G(I, J))
2650 GOTO 2670
2660 DEL(I, J) = ATN(B(I, J) / G(I, J)) + 3.141592654#
2670 NEXT J
2680 NEXT I
2690 OPEN "O", #1, "ADMIT"    ' file "admit" contained elements of bus admittance matrix Ybus (in Polar form)
2700 WRITE #1, G, M, N
2710 FOR I = 1 TO N
2720 FOR J = 1 TO N
2730 WRITE #1, Y(I, J), DEL(I, J)
2740 NEXT J
2750 NEXT I
2760 CLOSE #1
2770 ERASE G, B, Y, DEL, LL, EE
2780 RETURN
3000 *****
3010 '*----- LOADFLOW CALCULATION -----*'
3020 *****
3030 ' V(i)   =   voltage magnitude at bus i
3040 ' se(i)  =   voltage phase angle at bus i
3050 ' PG(i)  =   real power generation at bus i
3060 ' QG(i)  =   reactive power generation at bus i

```

```

3070 ' PL(i) = real load supplied at bus i
3080 ' QL(i) = reactive load supplied at bus i
3090 ' P(i) = net real power injection at bus i (Pi,spec)
3100 ' Q(i) = net reactive power injection at bus i (Qi,spec)
3110 ' PI(i) = Pi,calc
3120 ' QI(i) = Qi,calc
3130 ' DP(i) = real power mismatch
3140 ' DQ(i) = reactive power mismatch
3150 !*****
3160 COLOR 3
3170 LOCATE 11
3180 PRINT Z1$ + Z9$ + Z2$
3190 PRINT Z1$ + Z10$ + Z2$
3200 PRINT Z1$ + Z11$ + Z2$
3210 PRINT Z1$ + Z12$ + Z2$
3220 FOR I = 1 TO 80
3230 LOCATE A, I: PRINT " ";
3240 LOCATE A + 1, I: PRINT " ";
3250 NEXT I
3260 LOCATE A
3270 PRINT TAB(25); "make a choice between (1) and (2) .....< ? >";
3280 K2$ = INPUT$(1)
3290 K2 = VAL(K2$)
3300 IF (K2 = 1) OR (K2 = 2) THEN GOTO 3360
3310 COLOR 19
3320 SOUND 50, 1
3330 SOUND 400, 1
3340 SOUND 1000, 1
3350 GOTO 3260
3360 COLOR 23, 4
3370 LOCATE 3, 34
3380 PRINT " please wait .. "
3390 SCREEN , , 1, 0
3400 COLOR , 4
3410 CLS
3420 Z40$ = CHR$(218)
3430 Z41$ = CHR$(191)
3440 Z42$ = STRING$(67, CHR$(196))
3450 Z43$ = CHR$(179)
3460 Z44$ = CHR$(192)
3470 Z45$ = CHR$(217)
3480 Z46$ = " ** - input data for loadflow calculation - ** "
3490 Z47$ = " (classical Newton-Raphson loadflow) "
3500 Z48$ = " (apply Hybrid algorithm to NPL) "
3510 ON K2 GOTO 3520, 3540
3520 X10$ = Z47$
3530 GOTO 3550
3540 X10$ = Z48$
3550 COLOR 14, 9
3560 LOCATE 2, 6: PRINT Z40$ + Z42$ + Z41$
3570 LOCATE , 6: PRINT Z43$ + Z46$ + Z43$
3580 LOCATE , 6: PRINT Z43$ + X10$ + Z43$

```

```

3590 LOCATE , 6: PRINT Z44$ + Z42$ + Z45$
3600 COLOR , 4
3610 OPEN "I", #1, "ADMIT"
3620 INPUT #1, G, M, N
3630 DIM P(N), PG(G), PL(N)
3640 DIM Q(N), QG(G), QL(N)
3650 DIM V(N), SE(N)
3660 DIM AINV(2 * (M + N + 1), 2 * (M + N + 1))
3670 DIM A(M + N + 1, M + N + 1)
3680 DIM DP(N), DQ(N), DSE(N), DV(N)
3690 DIM PI(N), QI(N), Y(N, N), DEL(N, N)
3700 DIM PLINE(N, N), QLINE(N, N)
3710 DIM PLOSS(N, N), QLOSS(N, N)
3720 FOR I = 1 TO N
3730 FOR J = 1 TO N
3740 INPUT #1, Y(I, J), DEL(I, J)
3750 NEXT J
3760 NEXT I
3770 CLOSE #1
3780 LOCATE CSRLIN + 1, 6: PRINT "# of generators (P-V buses) = "; G
3790 PRINT TAB(6); "# of load buses (P-Q buses) = "; M
3800 PRINT TAB(6); "total # of buses          = "; N
3810 COLOR , 10
3820 FOR I = 11 TO 22
3830 LOCATE I, 6: PRINT STRING$(70, " ")
3840 NEXT I
3850 SCREEN , , 1, 1
3860 COLOR 14, 9
3870 LOCATE 14, 20: PRINT " P,gen = "
3880 LOCATE , 20: PRINT " V,bus = "
3890 LOCATE , 20: PRINT " P,load = "
3900 LOCATE , 20: PRINT " Q,load = "
3910 LOCATE 12, 20: PRINT " bus no. 1 (slack bus) "
3920 LOCATE 14, 31: PRINT " -" ' enter input data for the swing bus
3930 LOCATE , 31: INPUT " ", V(1)
3940 LOCATE , 31: INPUT " ", PL(1)
3950 LOCATE , 31: INPUT " ", QL(1)
3960 LOCATE 12, 20: PRINT " bus no. 2 (P-V bus (generator bus) )"
3970 FOR I = 2 TO G ' enter input data for P-V buses
3980 LOCATE 12, 20: PRINT " bus no."; I
3990 LOCATE 14, 31: PRINT STRING$(20, " ")
4000 LOCATE , 31: PRINT STRING$(20, " ")
4010 LOCATE , 31: PRINT STRING$(20, " ")
4020 LOCATE , 31: PRINT STRING$(20, " ")
4030 LOCATE 14, 31: INPUT " ", PG(I)
4040 LOCATE , 31: INPUT " ", V(I)
4050 LOCATE , 31: INPUT " ", PL(I)
4060 LOCATE , 31: INPUT " ", QL(I)
4070 P(I) = PG(I) - PL(I)
4080 NEXT I
4090 LOCATE 12, 20: PRINT " bus no. (P-Q bus (load bus )) "
4100 FOR I = G + 1 TO N ' enter input data for P-Q buses

```



```

4110 LOCATE 12, 20: PRINT " bus no.": I
4120 LOCATE 14, 31: PRINT STRING$(20, " ")
4130 LOCATE , 31: PRINT STRING$(20, " ")
4140 LOCATE , 31: PRINT STRING$(20, " ")
4150 LOCATE , 31: PRINT STRING$(20, " ")
4160 LOCATE 14, 31: PRINT " - "
4170 LOCATE , 31: PRINT " - "
4180 LOCATE , 31: INPUT " ", PL(I)
4190 LOCATE , 31: INPUT " ", QL(I)
4200 P(I) = -PL(I)
4210 Q(I) = -QL(I)
4220 NEXT I
4230 MN = M + N - 1
4240 ***** -- set initial condition
4250 SE(I) = 0 ' set swing bus angle = 0
4260 FOR I = 2 TO N ' set initial starts for bus voltage phase angles (all buses) in rad.
4270 SE(I) = 0
4280 NEXT I
4290 FOR I = G + 1 TO N ' set initial starts for load bus voltage magnitude in p.u
4300 V(I) = 1
4310 NEXT I
4320 ***** ---- whether to swich to Hybrid method or classical Newton-Raphson
4330 DIM R(MN), DX(MN), GD(MN)
4340 GOSUB 15150
4350 IF K2 = 2 THEN GOSUB 6000 ELSE GOSUB 5000
4360 GOSUB 11010
4370 RETURN
5000 *****
5010 *****----- LOADFLOW BY STANDARD NEWTON-RAPHSON METHOD -----*****
5020 *****
5030 GOSUB 15150
5040 *****-- main program
5050 E = .000000000000001# ' set epselon
5060 IT = 0 ' set iteration count
5070 IT = IT + 1
5080 GOSUB 7510 ' calc. residual
5090 GOSUB 15390
5100 GOSUB 9000 ' calc. Jacobian matrix
5110 FOR I = 1 TO MN ' calc. Inverse Jacobain
5120 FOR J = 1 TO MN
5130 AINV(I, J) = A(I, J)
5140 NEXT J
5150 NEXT I
5160 GOSUB 10010
5170 FOR I = 1 TO MN ' calc. step dx
5180 DX(I) = 0
5190 FOR J = 1 TO MN
5200 DX(I) = DX(I) + AINV(I, J) * R(J)
5210 NEXT J
5220 NEXT I
5230 NN = 0
5240 FOR I = 2 TO N ' updating current angle

```

```

5250 NN = NN + 1
5260 SE(I) = SE(I) + DX(NN)
5270 NEXT I
5280 FOR I = G + 1 TO N           ' updating current voltage
5290 NN = NN + 1
5300 V(I) = V(I) + DX(NN)
5310 NEXT I
5320 GOSUB 7510                 ' calc. current residual
5330 IF F < E THEN RETURN      ' test convergence
5340 GOTO 5070

6000 *****
6010 ****--- HYBRID ALGORITHM FOR STANDARD NEWTON-RAPHSON LOADFLOW ---*****
6020 *****
6030 ' A(,)      =      Jacobian matrix
6040 ' Gd(,)     =      Grandient matrix of f
6050 ' g1        =      norm of Gd(,)
6060 ' R(,)      =      residual , dim M
6070 ' X(,)      =      variable vector
6080 ' DX(,)     =      vector of the predicted step
6090 ' normx     =      norm of NEWTON-RAPHSON step
6100 ' F1        =      sum of residuals in previous iteration
6110 ' F         =      sum of residuals in current iteration
6120 ' R         =      trust radius
6130 ' E        =      specified tolerance limit
6140 *****
6150 ***** -- main program
6160 DIM FEE(MN)
6170 E = .0000000000001#: TMAX = 1000: TINC = 1
6180 DV = .2
6190 DSE = .2
6200 R = SQR(M * (DV ^ 2) + N * (DSE ^ 2)) ' calc. approx. trust radius R
6220 IT = 0                               ' set init. iteration no.
6230 GOSUB 7510                           ' calc. first residual
6240 F1 = F
6250 ***** ----- re-entry point for new iteration
6260 IT = IT + 1                           ' iteration count
6270 STATUS = 0
6280 F1 = F
6290 CUT = 0
6300 UG = 0
6310 GOSUB 9000                           ' calc. Jacobian matrix J(,)
6320 G1 = 0                               ' calc. gradient Gd(,) and norm g1
6330 FOR I = 1 TO MN
6340 GD(I) = 0
6350 FOR J = 1 TO MN
6360 GD(I) = GD(I) + A(J, I) * R(J)
6370 NEXT J
6380 GD(I) = -2 * GD(I)
6390 G1 = G1 + GD(I) * GD(I)
6400 NEXT I
6410 G1 = SQR(G1)

```

```

6420 IF G1 = 0 THEN RETURN
6430 !**** ----- Standard Newton-Raphson iteration
6440 FOR I = 1 TO MN
6450 FOR J = 1 TO MN
6460 AINV(I, J) = A(I, J)
6470 NEXT J
6480 NEXT I
6490 GOSUB 10010 ' calc. inverse Jacobian matrix
6500 FOR I = 1 TO MN
6510 DX(I) = 0
6520 FOR J = 1 TO MN
6530 DX(I) = DX(I) + AINV(I, J) * R(J)
6540 NEXT J
6550 NEXT I
6560 NORMX = 0 ' calc. norm of dx
6570 FOR I = 1 TO MN
6580 NORMX = NORMX + DX(I) * DX(I)
6590 NEXT I
6600 NORMX = SQR(NORMX)
6610 DENO = 0 ' calc. deno. factor for quadratic factor
6620 FOR I = 1 TO MN
6630 DENO = DENO + GD(I) * DX(I)
6640 NEXT I
6650 DENO = -DENO / 2
6660 IF NORMX > R THEN GOTO 6720 ' test whether normx is inside R
6670 GOSUB 7670 ' updating X(,)
6680 GOSUB 7510 ' calc. current residual
6690 IF F < F1 THEN GOTO 7150 ' test whether current Newton step
6700 R = R / 2: TINC = 1 ' is successful to decrease residual
6710 GOSUB 7780
6720 !**** ----- steepest descent iteration
6730 U = 0 ' calc. predicted step to min. F(x)
6740 FOR I = 1 TO MN
6750 AA = 0
6760 FOR J = 1 TO MN
6770 AA = AA + A(I, J) * GD(J)
6780 NEXT J
6790 U = U + AA * AA
6800 NEXT I
6810 U = (G1 * G1) / U
6820 UG = U * G1
6830 IF UG >= R THEN GOTO 7010 ' test whether predicted step > r
6840 GX = 0
6850 FOR I = 1 TO MN
6860 GX = GX + GD(I) * DX(I)
6870 NEXT I
6880 A = (U ^ 2) * (G1 ^ 2) + (NORMX ^ 2) + 2 * U * GX
6890 B = U * GX + (U ^ 2) * (G1 ^ 2)
6900 C = (U ^ 2) * (G1 ^ 2) - (R ^ 2)
6910 IF (B ^ 2 - A * C) < 0 THEN GOTO 6960
6920 ALPA = (B + SQR((B ^ 2) - A * C)) / A
6930 IF (ALPA > 0) AND (ALPA < 1) THEN GOTO 6960

```

```

6940 ALPA = (B - SQR((B ^ 2) - A * C)) / A
6950 IF (ALPA > 0) AND (ALPA < 1) THEN GOTO 6960 ELSE GOTO 7010
6960 STATUS = 2
6970 FOR I = 1 TO MN
6980 DX(I) = (ALPA - 1) * U * GD(I) + ALPA * DX(I)
6990 NEXT I
7000 GOTO 7050
7010 STATUS = 1
7020 FOR I = 1 TO MN          ' calc. pure cauchy step
7030 DX(I) = -R * (GD(I) / G1)
7040 NEXT I
7050 GOSUB 7670              ' updating X(,)
7060 GOSUB 7510              ' calc. current residual
7070 IF F < F1 THEN GOTO 7150 ' test whether current cauchy step
7080                          ' is successful to decrease residual
7090 GOSUB 7780              ' set X(,) back to last turning pt
7100 R = R / 2: TINC = 1     ' reduce restrict step length
7110 CUT = CUT + 1           ' cut back count
7120 IF CUT > 1000 THEN RETURN ' test whether cauchy step is too
7130                          ' small
7140 GOTO 6830               ' updating X(,) and recal. residual
7150 RQ = (F1 - F) / DENO
7160 ***** ----- revise restrict step length
7170 IF F > (F1 - .1 * (F1 - FEE)) THEN GOTO 7180 ELSE GOTO 7200
7180 R = R / 2
7190 GOTO 7350
7200 AA = F1 - .1 * (F1 - FEE) - F ' calc. dmult
7210 BB = 0
7220 CC = 0
7230 FOR I = 1 TO MN          ' calc. sp & ss
7240 BB = BB + ABS(R(I) * (R(I) - FEE(I)))
7250 CC = CC + (R(I) - FEE(I)) ^ 2
7260 NEXT I
7270 RAMDA = 1 + AA / (BB + SQR(BB * BB + AA * CC))
7280 RAMDA = SQR(RAMDA)
7290 IF RAMDA < 2 THEN U1 = RAMDA ELSE GOTO 7320
7300 IF RAMDA < TINC THEN U2 = RAMDA ELSE U2 = TINC
7310 GOTO 7330
7320 IF 2 < TINC THEN U2 = 2 ELSE U2 = TINC
7330 TINC = RAMDA / U2
7340 R = U2 * R
7350 FEE = 0
7360 FOR I = 1 TO MN
7370 FEE(I) = 0
7380 FOR J = 1 TO MN
7390 FEE(I) = FEE(I) + A(I, J) * DX(J)
7400 NEXT J
7410 FEE(I) = FEE(I) + R(I)
7420 NEXT I
7430 FOR I = 1 TO MN
7440 FEE = FEE + FEE(I) * FEE(I)
7450 NEXT I

```

```

7460 GOSUB 15390
7470 !**** ----- test convegence
7480 IF F < E THEN RETURN
7490 IF IT > TMAX THEN RETURN          ' test whether too many iteration
7500 GOTO 6260
7510 !**** ----- subroutine for calc. residual F(x) and R(,)
7520 GOSUB 8000
7530 NN = 0
7540 FOR I = 2 TO N
7550 NN = NN + 1
7560 R(NN) = DP(I)
7570 NEXT I
7580 FOR I = G + 1 TO N
7590 NN = NN + 1
7600 R(NN) = DQ(I)
7610 NEXT I
7620 F = 0
7630 FOR I = 1 TO MN
7640 F = F + R(I) * R(I)
7650 NEXT I
7660 RETURN
7670 !**** ----- subroutine for updating X(,)
7680 NN = 0
7690 FOR I = 2 TO N
7700 NN = NN + 1
7710 SE(I) = SE(I) + DX(NN)
7720 NEXT I
7730 FOR I = G + 1 TO N
7740 NN = NN + 1
7750 V(I) = V(I) + DX(NN)
7760 NEXT I
7770 RETURN
7780 !**** ----- subroutine for setting X(,) back to last turning points
7790 NN = 0
7800 FOR I = 2 TO N
7810 NN = NN + 1
7820 SE(I) = SE(I) - DX(NN)
7830 NEXT I
7840 FOR I = G + 1 TO N
7850 NN = NN + 1
7860 V(I) = V(I) - DX(NN)
7870 NEXT I
7880 RETURN
8000 !****--- subroutine for calculating real and reactive power mismatch ---*****
8010 FOR I = 2 TO N          ' updating real power
8020 PI(I) = 0
8030 FOR J = 1 TO N
8040 PI(I) = PI(I) + V(I) * Y(I, J) * V(J) * COS(DEL(I, J) + SE(J) - SE(I))
8050 NEXT J
8060 NEXT I
8070 FOR I = G + 1 TO N          ' updating reactive power
8080 QI(I) = 0

```

```

8090 FOR J = 1 TO N
8100 QI(I) = QI(I) - V(I) * Y(I, J) * V(J) * SIN(DEL(I, J) + SE(J) - SE(I))
8110 NEXT J
8120 NEXT I
8130 FOR I = 2 TO N
8140 DP(I) = P(I) - PI(I)
8150 NEXT I
8160 FOR I = G + 1 TO N
8170 DQ(I) = Q(I) - QI(I)
8180 NEXT I
8190 RETURN
9000 *****
9010 *****--- subroutine for calc. Jacobian matrix ---*****
9020 ERASE A
9030 DIM A(2 * (M + N + 1), 2 * (M + N + 1))
9040 FOR I = 2 TO N           'form submatrix J1
9050 FOR J = 2 TO N
9060 IF I <> J THEN GOTO 9120
9070 FOR JJ = 1 TO N
9080 IF I = JJ THEN GOTO 9100
9090 A(I, I) = A(I, I) + V(I) * Y(I, JJ) * V(JJ) * SIN(DEL(I, JJ) + SE(JJ) - SE(I))
9100 NEXT JJ
9110 GOTO 9130
9120 A(I, J) = -V(I) * Y(I, J) * V(J) * SIN(DEL(I, J) + SE(J) - SE(I))
9130 NEXT J
9140 NEXT I
9150 FOR I = 2 TO N           'form submatrix J2
9160 FOR J = G + 1 TO N
9170 IF I <> J THEN GOTO 9260
9180 FOR JJ = 1 TO N
9190 IF I = JJ THEN GOTO 9210
9200 GOTO 9230
9210 A(I, N + J - G) = A(I, N + J - G) + 2 * Y(I, JJ) * V(JJ) * COS(DEL(I, JJ))
9220 GOTO 9240
9230 A(I, N + J - G) = A(I, N + J - G) + Y(I, JJ) * V(JJ) * COS(DEL(I, JJ) + SE(JJ) - SE(I))
9240 NEXT JJ
9250 GOTO 9270
9260 A(I, N + J - G) = V(I) * Y(I, J) * COS(DEL(I, J) + SE(J) - SE(I))
9270 NEXT J
9280 NEXT I
9290 FOR I = G + 1 TO N           'form submatrix J3
9300 FOR J = 2 TO N
9310 IF I <> J THEN GOTO 9370
9320 FOR JJ = 1 TO N
9330 IF I = JJ THEN GOTO 9350
9340 A(N + I - G, J) = A(N + I - G, J) + V(I) * Y(I, JJ) * V(JJ) * COS(DEL(I, JJ) + SE(JJ) - SE(I))
9350 NEXT JJ
9360 GOTO 9380
9370 A(N + I - G, J) = -V(I) * Y(I, J) * V(J) * COS(DEL(I, J) + SE(J) - SE(I))
9380 NEXT J
9390 NEXT I
9400 FOR I = G + 1 TO N           'form submatrix J4

```

```

9410 FOR J = G + 1 TO N
9420 IF I <> J THEN GOTO 9510
9430 FOR JJ = 1 TO N
9440 IF I = JJ THEN GOTO 9460
9450 GOTO 9480
9460 A(N + I - G, N + J - G) = A(N + I - G, N + J - G) - 2 * V(I) * Y(I, JJ) * SIN(DEL(I, JJ) + SE(JJ) - SE(I))
9470 GOTO 9490
9480 A(N + I - G, N + J - G) = A(N + I - G, N + J - G) - Y(I, JJ) * V(JJ) * SIN(DEL(I, JJ) + SE(JJ) - SE(I))
9490 NEXT JJ
9500 GOTO 9520
9510 A(N + I - G, N + J - G) = -V(I) * Y(I, J) * SIN(DEL(I, J) + SE(J) - SE(I))
9520 NEXT J
9530 NEXT I
9540 FOR I = 1 TO MN
9550 FOR J = 1 TO MN
9560 A(I, J) = A(I + 1, J + 1)
9570 NEXT J
9580 NEXT I
9590 RETURN
10000 *****
10010 *****----subroutine for calc. Inverse matrix
10020 K = 0
10030 FOR X = 1 TO MN 'form RHS unity matrix
10040 FOR Y = 1 TO MN
10050 IF X = Y THEN GOTO 10080
10060 AINV(X, Y + MN) = 0
10070 GOTO 10090
10080 AINV(X, Y + MN) = 1
10090 NEXT Y
10100 NEXT X
10110 FOR L = 1 TO MN
10120 K = K + 1
10130 IF AINV(L, K) <> 0 OR L <> MN THEN GOTO 10160
10140 K = K + 1
10150 GOTO 10130
10160 IF AINV(L, K) = 0 AND L <> MN THEN GOTO 10510
10170 FOR X = 2 * MN TO 1 STEP -1
10180 AINV(L, X) = AINV(L, X) / AINV(L, K)
10190 NEXT X
10200 FOR J = 1 TO MN
10210 IF J = L THEN GOTO 10260
10220 FOR X = 2 * MN TO 1 STEP -1
10230 AINV(J, X) = AINV(J, X) - AINV(J, K) * AINV(L, X)
10240 IF ABS(AINV(J, X)) < ABS(AINV(J, X + 1)) / 100000000000# THEN AINV(J, X) = 0
10250 NEXT X
10260 NEXT J
10270 NEXT L
10280 FOR Y = 1 TO MN
10290 S = 0
10300 FOR X = 1 TO MN
10310 IF AINV(Y, X) <> 0 THEN S = 1
10320 NEXT X

```

```

10330 FOR XX = MN + 1 TO 2 * MN
10340 IF S = 0 AND AINV(Y, XX) <> 0 THEN GOTO 10640
10350 NEXT XX
10360 NEXT Y
10370 FOR L = 1 TO MN
10380 IF AINV(L, L) <> 1 THEN GOTO 10630
10390 NEXT L
10400 FOR Y = 1 TO MN
10410 FOR X = 1 TO MN
10420 IF X <> Y AND AINV(Y, X) <> 0 THEN GOTO 10630
10430 NEXT X
10440 NEXT Y
10450 FOR X = 1 TO MN
10460 FOR Y = 1 TO MN
10470 AINV(X, Y) = AINV(X, Y + MN)
10480 NEXT Y
10490 NEXT X
10500 RETURN
10510 FOR H = L + 1 TO MN
10520 IF AINV(H, K) <> 0 THEN GOTO 10570
10530 NEXT H
10540 K = K + 1
10550 IF K > 2 * MN THEN GOTO 10280
10560 GOTO 10130
10570 FOR Z = 1 TO 2 * MN
10580 T = AINV(L, Z)
10590 AINV(L, Z) = AINV(H, Z)
10600 AINV(H, Z) = T
10610 NEXT Z
10620 GOTO 10130
10630 CLS : PRINT "NOT INDEPENDENT": END
10640 CLS : PRINT "CONTRADICTION": END
11000 *****
11010 *****-----subroutine for calc. lineflow and line loss-----*****
11020 *****
11030 ' PLINE(i,j)   =   real power flow from bus i to bus j
11040 ' QLINE(i,j)   =   reactive power flow from bus i to bus j
11050 ' PLOSS(i,j)   =   real power loss along line i,j
11060 ' QLOSS(i,j)   =   reactive power along line i,j
11070 *****
11080 FOR J = 1 TO N           ' compute net real and reactive power injected at the swing bus
11090 P(1) = P(1) + V(1) * Y(1, J) * V(J) * COS(DEL(1, J) + SE(J) - SE(1))
11100 Q(1) = Q(1) - V(1) * Y(1, J) * V(J) * SIN(DEL(1, J) + SE(J) - SE(1))
11110 NEXT J
11120 FOR I = 2 TO G         ' compute reactive power injected at P-V buses
11130 FOR J = 1 TO N
11140 Q(I) = Q(I) - V(I) * Y(I, J) * V(J) * SIN(DEL(I, J) + SE(J) - SE(I))
11150 NEXT J
11160 NEXT I
11170 PG(1) = P(1) - PL(1)  ' compute real and reactive power generation at swing bus
11180 QG(1) = Q(1) - QL(1)
11190 FOR I = 2 TO G       ' compute reactive power generation at P-V buses

```



```

11200 QG(I) = Q(I) - QL(I)
11210 NEXT I
11220 ERASE A, DP, DQ, PI, QI, Y, DEL
11230 DIM PLINE, QLINE, PLOSS, QLOSS
11240 DIM G(N, N), B(N, N), LL(N, N), EE(N, N), I(N, N), AI(N, N)
11250 OPEN "i", #2, "admitL"
11260 OPEN "j", #3, "admit2"
11270 FOR I = 1 TO N
11280 FOR J = I TO N
11290 INPUT #2, LL(I, J), EE(I, J)
11300 INPUT #3, G(I, J), B(I, J)
11310 NEXT J
11320 NEXT I
11330 CLOSE #2
11340 CLOSE #3
11350 FOR I = 1 TO N
11360 FOR J = I TO N
11370 IF I = J THEN GOTO 11640
11380 VR = 0
11390 VI = 0
11400 VV = 0
11410 DV = 0
11420 VR = V(I) * COS(SE(I)) - V(J) * COS(SE(J))
11430 VI = V(I) * SIN(SE(I)) - V(J) * SIN(SE(J))
11440 VV = SQR(VR ^ 2 + VI ^ 2)
11450 IF VR = 0 THEN GOTO 11470
11460 GOTO 11510
11470 IF VI > 0 THEN DV = 3.141592654# / 2
11480 IF VI < 0 THEN DV = -3.141592654# / 2
11490 IF VI = 0 THEN DV = 0
11500 GOTO 11560
11510 IF VI > 0 THEN GOTO 11530
11520 IF VI < 0 THEN GOTO 11550
11530 DV = ATN(VI / VR)
11540 GOTO 11560
11550 DV = ATN(VI / VR) + 3.141592654#
11560 I(I, J) = VV * LL(I, J) ' calc. current magnitude along line i,j
11570 AI(I, J) = DV + EE(I, J) ' calc. current phase angle along line i,j
11580 PLINE(I, J) = V(I) * I(I, J) * COS(SE(I) - AI(I, J)) ' calc. real and reactive power flow from bus i to bus
11590 QLINE(I, J) = V(I) * I(I, J) * SIN(SE(I) - AI(I, J))
11600 PLINE(J, I) = V(J) * I(I, J) * COS(SE(J) - AI(I, J)) ' calc. real and reactive power flow from bus j to bus
11610 QLINE(J, I) = V(J) * I(I, J) * SIN(SE(J) - AI(I, J))
11620 PLOSS(I, J) = ABS(PLINE(I, J) - PLINE(J, I)) ' calc. real & reactive power losses along line i,j
11630 QLOSS(I, J) = ABS(QLINE(I, J) - QLINE(J, I))
11640 NEXT J
11650 NEXT I
12000 *****
12010 *****----subroutine for display. or print out the result
12020 COLOR , 0
12030 CLS
12040 COLOR , 4
12050 FOR I = 6 TO 16

```

```

12060 LOCATE 1, 12: PRINT STRING$(55, " ")
12070 NEXT I
12080 COLOR 3
12090 LOCATE 7, 15: PRINT "Show the results from calculation .... "
12100 LOCATE 9, 35: PRINT "1.) on screen"
12110 LOCATE 10, 35: PRINT "2.) by printer"
12120 LOCATE 11, 35: PRINT "3.) go back to main menu"
12130 LOCATE 13, 35: PRINT "select options ..... ";
12140 S$ = INPUT$(1)
12150 S = VAL(S$)
12160 IF (S = 1) OR ((S = 2) OR (S = 3)) THEN GOTO 12170 ELSE GOTO 12200
12170 IF S = 3 THEN RETURN
12180 ON S GOSUB 12270, 14000
12190 GOTO 12020
12200 SOUND 100, 1
12210 SOUND 500, 1
12220 SOUND 1000, 1
12230 LOCATE 13, 35: PRINT STRING$(25, " ")
12240 COLOR 19
12250 GOTO 12130
12260 RETURN
12270 *****
12280 *****-- display the result on screen
12290 GOSUB 13500
12300 PRINT TAB(3); "1 ."; TAB(11); "SW."; TAB(17);
12310 PRINT USING "##.###"; V(1);
12320 PRINT TAB(24);
12330 PRINT USING "###.###"; SE(1);
12340 PRINT TAB(37);
12350 PRINT USING "##.#####"; PG(1);
12360 PRINT TAB(47);
12370 PRINT USING "##.#####"; QG(1);
12380 PRINT TAB(61);
12390 PRINT USING "##.#####"; PL(1);
12400 PRINT TAB(71);
12410 PRINT USING "##.#####"; QL(1)
12420 FOR I = 2 TO G
12430 IF CSRLIN > 22 THEN GOTO 12450
12440 GOTO 12490
12450 PRINT
12460 PRINT TAB(25); "press anykey to see more ..";
12470 KK$ = INPUT$(1)
12480 GOSUB 13500
12490 PRINT TAB(2); I; ". "; TAB(11); "P-V"; TAB(17);
12500 PRINT USING "##.###"; V(I);
12510 PRINT TAB(24);
12520 PRINT USING "###.###"; SE(I) * 57.3;
12530 PRINT TAB(37);
12540 PRINT USING "##.#####"; PG(I);
12550 PRINT TAB(47);
12560 PRINT USING "##.#####"; QG(I);
12570 PRINT TAB(61);

```

```
12580 PRINT USING "##.####"; PL(I);
12590 PRINT TAB(71);
12600 PRINT USING "##.####"; QL(I);
12610 NEXT I
12620 FOR I = G + 1 TO N
12630 PRINT TAB(2); I; ". "; TAB(11); "P-Q"; TAB(17);
12640 PRINT USING "##.####"; V(I);
12650 PRINT TAB(24);
12660 PRINT USING "###.###"; SE(I) * 57.3;
12670 PRINT TAB(40); "- "; TAB(50); "- ";
12680 PRINT TAB(61);
12690 PRINT USING "##.####"; -P(I);
12700 PRINT TAB(71);
12710 PRINT USING "##.####"; -Q(I)
12720 IF CSRLIN > 22 THEN GOTO 12740
12730 GOTO 12770
12740 PRINT TAB(25); "press anykey to see more ..";
12750 KK$ = INPUT$(1)
12760 GOSUB 13500
12770 NEXT I
12780 PRINT
12790 FOR I = 1 TO 80
12800 LOCATE CSRLIN, I
12810 PRINT CHR$(95);
12820 NEXT I
12830 PRINT
12840 PRINT TAB(25); "press anykey to see lineflows and line loss"
12850 KK$ = INPUT$(1)
12860 GOSUB 13270
12870 FOR I = 1 TO N
12880 FOR J = 1 TO N
12890 IF I = J THEN GOTO 13150
12900 IF G(I, J) = 0 AND B(I, J) = 0 THEN GOTO 13150
12910 IF CSRLIN > 22 THEN GOTO 12930
12920 GOTO 12970
12930 PRINT
12940 PRINT TAB(25); "press anykey to see more ..";
12950 KK$ = INPUT$(1)
12960 GOSUB 13310
12970 PRINT USING "##"; I;
12980 PRINT "- ";
12990 PRINT USING "##"; J;
13000 LOCATE CSRLIN, 9
13010 PRINT USING "##.###"; G(1, J);
13020 IF B(I, J) >= 0 THEN PRINT "+j";
13030 IF B(I, J) < 0 THEN PRINT "-j";
13040 PRINT USING "##.###"; ABS(B(I, J));
13050 LOCATE CSRLIN, 23
13060 PRINT USING "##.####"; I(I, J);
13070 LOCATE CSRLIN, 33
13080 PRINT USING "###.####"; PLINE(I, J);
13090 LOCATE CSRLIN, 43
```

```

13100 PRINT USING "###.###"; QLINE(I, J);
13110 LOCATE CSRLIN, 59
13120 PRINT USING "###.###"; PLOSS(I, J);
13130 LOCATE CSRLIN, 69
13140 PRINT USING "###.###"; QLOSS(I, J)
13150 NEXT J
13160 NEXT I
13170 PRINT
13180 FOR I = 1 TO 80
13190 LOCATE CSRLIN, I
13200 PRINT CHR$(95);
13210 NEXT I
13220 PRINT
13230 PRINT TAB(25); "press anykey to go back to menu"
13240 KK$ = INPUT$(1)
13250 RETURN
13260 '
13270 COLOR , 4
13280 CLS
13290 COLOR 14, 9
13300 Z60$ = "          <<.. Report From Loadflow Calculation ..>>          "
13310 LOCATE 2, 2: PRINT CHR$(218) + STRING$(75, CHR$(196)) + CHR$(191)
13320 LOCATE , 2: PRINT CHR$(179) + Z60$ + CHR$(179)
13330 LOCATE , 2: PRINT CHR$(192) + STRING$(75, CHR$(196)) + CHR$(217)
13340 COLOR , 4
13350 PRINT
13360 PRINT STRING$(79, CHR$(196))
13370 PRINT TAB(3); "bus"; TAB(10); "admitance";
13380 PRINT TAB(26); "I";
13390 PRINT TAB(32); "*** -- Line Flow -- ***";
13400 PRINT SPACE$(5); "*** -- Line Loss -- ***"
13410 LOCATE CSRLIN, 32: PRINT STRING$(20, CHR$(196));
13420 LOCATE , 58: PRINT STRING$(20, CHR$(196))
13430 LOCATE CSRLIN, 36
13440 PRINT "P(p.u)"; SPACE$(8); "Q(p.u)";
13450 LOCATE CSRLIN, 62
13460 PRINT "P(p.u)"; SPACE$(8); "Q(p.u)"
13470 PRINT STRING$(79, CHR$(196))
13480 PRINT
13490 RETURN
13500 '
13510 COLOR , 4
13520 CLS
13530 COLOR 14, 9
13540 Z60$ = "          <<.. Report From Loadflow Calculation ..>>          "
13550 LOCATE 2, 2: PRINT CHR$(218) + STRING$(75, CHR$(196)) + CHR$(191)
13560 LOCATE , 2: PRINT CHR$(179) + Z60$ + CHR$(179)
13570 LOCATE , 2: PRINT CHR$(192) + STRING$(75, CHR$(196)) + CHR$(217)
13580 COLOR , 4
13590 PRINT
13600 PRINT STRING$(79, CHR$(196))
13610 PRINT "Bus no."; SPACE$(2); "Type";

```

```

13620 PRINT SPACE$(3); "Volts"; SPACE$(3); "Angle";
13630 PRINT SPACE$(5); "****---Generation---****";
13640 PRINT SPACE$(2); "****-----Load-----****";
13650 LOCATE , 35: PRINT STRING$(45, CHR$(196))
13660 PRINT TAB(40); "P(p.u)"; SPACE$(7); "Q(p.u)";
13670 PRINT SPACE$(11); "P(p.u)"; SPACE$(7); "Q(p.u)"
13680 PRINT STRING$(79, CHR$(196))
13690 PRINT
13700 RETURN
14000 *****
14010 *****--- print out the result
14020 CLS
14030 LOCATE 12, 25
14040 COLOR 0, 7
14050 PRINT "**** -- printing the results -- ****"
14060 LPRINT CHR$(27); "-"; CHR$(1); "RESULT OF POWER FLOW CALCULATION";
14070 LPRINT CHR$(27); "-"; CHR$(0);
14080 LPRINT TAB(50); "# of iterations = "; IT
14090 LPRINT
14100 FOR I = 1 TO 80
14110 LPRINT CHR$(95);
14120 NEXT I
14130 LPRINT
14140 LPRINT "BUS NO."; SPACE$(2); "TYPE"; SPACE$(3); "VOLTS"; SPACE$(3); "ANGLE"; SPACE$(5);
14150 LPRINT "****---GENERATION---****"; SPACE$(2); "****-----LOAD-----****";
14160 LPRINT TAB(35);
14170 FOR I = 34 TO 79
14180 LPRINT CHR$(95);
14190 NEXT I
14200 LPRINT TAB(38); "P(p.u)"; SPACE$(7); "Q(p.u)"; SPACE$(10); "P(p.u)"; SPACE$(7); "Q(p.u)"
14210 FOR I = 1 TO 80
14220 LPRINT CHR$(95);
14230 NEXT I
14240 LPRINT
14250 LPRINT TAB(3); "1 ."; TAB(11); "SW"; TAB(17);
14260 LPRINT USING "##.###"; V(1);
14270 LPRINT TAB(24);
14280 LPRINT USING "###.###"; SE(1);
14290 LPRINT TAB(37);
14300 LPRINT USING "##.#####"; PG(1);
14310 LPRINT TAB(47);
14320 LPRINT USING "##.#####"; QG(1);
14330 LPRINT TAB(61);
14340 LPRINT USING "##.#####"; PL(1);
14350 LPRINT TAB(71);
14360 LPRINT USING "##.#####"; QL(1);
14370 FOR I = 2 TO G
14380 LPRINT TAB(2); I; ". "; TAB(11); "P-V"; TAB(17);
14390 LPRINT USING "##.###"; V(1);
14400 LPRINT TAB(24);
14410 LPRINT USING "###.###"; SE(I) * 57.3;
14420 LPRINT TAB(37);

```

```

14430 LPRINT USING "##.#####"; PG(I);
14440 LPRINT TAB(47);
14450 LPRINT USING "##.#####"; QG(I);
14460 LPRINT TAB(61);
14470 LPRINT USING "##.#####"; PL(I);
14480 LPRINT TAB(71);
14490 LPRINT USING "##.#####"; QL(I);
14500 NEXT I
14510 FOR I = G + 1 TO N
14520 LPRINT TAB(2); I; ". "; TAB(11); "P-Q"; TAB(17);
14530 LPRINT USING "##.###"; V(I);
14540 LPRINT TAB(24);
14550 LPRINT USING "###.###"; SE(I) * 57.3;
14560 LPRINT TAB(40); "- "; TAB(50); "- ";
14570 LPRINT TAB(61);
14580 LPRINT USING "##.#####"; -P(I);
14590 LPRINT TAB(71);
14600 LPRINT USING "##.#####"; -Q(I);
14610 NEXT I
14620 LPRINT
14630 FOR I = 1 TO 80
14640 LPRINT CHR$(95);
14650 NEXT I
14660 LPRINT : LPRINT
14670 LPRINT TAB(13); "----- LINEFLOW -----"
14680 FOR I = 1 TO 80
14690 LPRINT CHR$(95);
14700 NEXT I
14710 LPRINT
14720 LPRINT TAB(3); "BUS"; TAB(10); "ADMITANCE";
14730 LPRINT TAB(26); "I";
14740 LPRINT TAB(32); "*** -- LINE FLOW -- ***";
14750 LPRINT SPACE$(5); "*** -- LINE LOSS -- ***"
14760 LPRINT TAB(32);
14770 FOR I = 1 TO 46
14780 LPRINT "- ";
14790 NEXT I
14800 LPRINT TAB(36); "P(p.u)"; SPACE$(8); "Q(p.u)";
14810 LPRINT TAB(62); "P(p.u)"; SPACE$(8); "Q(p.u)"
14820 FOR I = 1 TO 80
14830 LPRINT CHR$(95);
14840 NEXT I
14850 LPRINT : LPRINT
14860 FOR I = 1 TO N
14870 FOR J = 1 TO N
14880 IF I = J THEN GOTO 15080
14890 IF G(I, J) = 0 AND B(I, J) = 0 THEN GOTO 15080
14900 LPRINT USING "##"; I;
14910 LPRINT "- ";
14920 LPRINT USING "##"; J;
14930 LPRINT TAB(8);
14940 LPRINT USING "##.##"; G(I, J);

```

```

14950 IF B(I, J) >= 0 THEN LPRINT "+j";
14960 IF B(I, J) < 0 THEN LPRINT "-j";
14970 LPRINT USING "##.##"; ABS(B(I, J));
14980 LPRINT TAB(23);
14990 LPRINT USING "##.###"; I(I, J);
15000 LPRINT TAB(33);
15010 LPRINT USING "###.###"; PLINE(I, J);
15020 LPRINT TAB(43);
15030 LPRINT USING "###.###"; QLINE(I, J);
15040 LPRINT TAB(59);
15050 LPRINT USING "###.###"; PLOSS(I, J);
15060 LPRINT TAB(69);
15070 LPRINT USING "###.###"; QLOSS(I, J)
15080 NEXT J
15090 NEXT I
15100 LPRINT
15110 FOR I = 1 TO 80
15120 LPRINT CHR$(95);
15130 NEXT I
15140 RETURN
15150 !*****
15160 !**-----SUBROUTINE FOR GENERATING SCREEN DURING COMPUTING-----*
15170 !*****
15180 Z90$ = "          ** -- Convegence Test During Computing -- **          "
15190 SCREEN , , 3, 1
15200 COLOR 4, 2
15210 CLS
15220 COLOR 14, 6
15230 LOCATE 2, 6: PRINT CHR$(218) + STRING$(67, CHR$(196)) + CHR$(191)
15240 LOCATE , 6: PRINT CHR$(179) + Z90$ + CHR$(179)
15250 LOCATE , 6: PRINT CHR$(192) + STRING$(67, CHR$(196)) + CHR$(217)
15260 LOCATE CSRLIN + 2, 6: PRINT " iteration no. ";
15270 COLOR 20, 2
15280 LOCATE , 50: PRINT ".....computing"
15290 COLOR 14, 9
15300 FOR I = 9 TO 22
15310 LOCATE I, 6: PRINT STRING$(67, " ")
15320 NEXT I
15330 COLOR 14, 6
15340 LOCATE 10, 10
15350 PRINT " K"; SPACE$(9); "F(x)"; SPACE$(9); " CSTEP "; SPACE$(10); "NEWTON-GRAD. "
15360 PRINT
15370 SCREEN , , 3, 3
15380 RETURN
15390 !*****-----show convegence test on screen
15400 COLOR 14, 9
15410 IF CSRLIN > 21 THEN GOTO 15420 ELSE GOTO 15460
15420 FOR I = 12 TO 21
15430 LOCATE I, 6: PRINT STRING$(67, " ")
15440 NEXT I
15450 LOCATE 12, 11
15460 LOCATE , 11

```

15470 PRINT IT; TAB(19);  
 15480 PRINT USING "###.#####"; F;  
 15490 LOCATE , 35: PRINT USING "##.####"; R;  
 15500 LOCATE , 55: PRINT USING "##.####"; UG  
 15510 LPRINT IT; : LPRINT USING "###.#####"; F1; : LPRINT STATUS; : LPRINT USING "###.####"; NORMX, R, RQ  
 15520 RETURN

i-j	$B_{ij}$	$B_{ij}$	i-j	$B_{ij}$	$B_{ij}$
1-1	0.0000	0.0050	4-7	0.4500	-2.4000
1-3	1.7320	-4.3100	7-8	0.0000	0.0650
1-4	10.0000	-20.0000	8-9	0.0000	-13.0000
2-2	0.0000	0.0650	8-5	0.0000	0.0400
2-8	1.0400	-7.1840	7-7	0.0000	0.0750
2-9	1.1300	-4.8770	7-3	1.6000	-5.6020
2-10	0.7900	-2.8010	7-9	1.8000	-7.4850
3-3	0.0000	0.0650	8-6	0.0000	0.0650
3-5	0.8700	-3.1900	9-9	0.0000	0.0650
3-8	10.0000	-20.0000	9-10	0.9340	-3.7350
4-4	0.0000	0.0750	10-10	0.0000	0.0300

TABLE B-1 Line admittance data for 10 bus system

bus no.	V  (p.u.)	$\theta$ (rad.)	generation (p.u.)		Load (p.u.)
			P	Q	P
1	1.06	0.0	-	-	0.00
2	1.02	-	1.50	-	0.00
3	-	-	0.00	0.00	-0.00
4	-	-	0.00	0.00	-0.00
5	-	-	0.00	0.00	-0.00
6	-	-	0.00	0.00	-0.00
7	-	-	0.00	0.00	-0.00
8	-	-	0.00	0.00	-0.00
9	-	-	0.00	0.00	-0.00
10	-	-	0.00	0.00	-0.00

TABLE B-2 Operating conditions of 10 bus system



## APPENDIX B

System Data for 10 Bus Test System

i-j	$G_{ij}$	$B_{ij}$	i-j	$G_{ij}$	$B_{ij}$
1-1	0.0000	0.0550	4-7	0.4500	-2.4000
1-3	1.7320	-4.3100	5-5	0.0000	0.0650
1-4	10.0000	-20.0000	5-6	5.0000	-15.0000
2-2	0.0000	0.0650	6-6	0.0000	0.0400
2-8	1.8400	-7.4840	7-7	0.0000	0.0750
2-9	1.1300	-4.4770	7-8	1.4000	-5.6020
2-10	0.7000	-2.8010	7-9	1.8400	-7.4840
3-3	0.0000	0.0850	8-8	0.0000	0.0650
3-5	0.8200	-2.1900	9-9	0.0000	0.0850
3-6	10.0000	-20.0000	9-10	0.9340	-3.7350
4-4	0.0000	0.0750	10-10	0.0000	0.0300

TABLE B-1 Line admittance data for 10 bus system

bus no.	$ V $ (p. u)	$\theta$ (rad.)	generation (p. u)		Load (p. u)	
			P	Q	P	Q
1	1.08	0.0	-	-	0.00	0.00
2	1.02	-	1.50	-	0.00	0.00
3	-	-	0.00	0.00	-0.85	-0.30
4	-	-	0.00	0.00	-0.35	-0.25
5	-	-	0.00	0.00	-0.75	-0.45
6	-	-	0.00	0.00	-0.75	-0.25
7	-	-	0.00	0.00	-0.65	-0.30
8	-	-	0.00	0.00	-0.40	-0.05
9	-	-	0.00	0.00	-0.85	-0.40
10	-	-	0.00	0.00	-0.70	-0.30

TABLE B-2 Operating condition of 10 bus system

APPENDIX C

Computer Results for the 10 Bus Tested System

RESULT OF POWER FLOW CALCULATION

# of iterations = 5

BUS NO.	TYPE	VOLTS	ANGLE	GENERATION		LOAD	
				P(p.u)	Q(p.u)	P(p.u)	Q(p.u)
1	SW	1.050	0.000	4.80769	3.30261	0.00000	0.00000
2	P-V	1.020	-50.458	1.50000	1.92042	0.00000	0.00000
3	P-Q	0.757	-17.677	-	-	0.65070	0.10000
4	P-Q	0.547	-4.492	-	-	0.35070	0.20000
5	P-Q	0.744	-18.383	-	-	0.15000	0.20000
6	P-Q	0.733	-19.570	-	-	0.15000	0.20000
7	P-Q	0.811	-32.465	-	-	0.45000	0.20000
8	P-Q	0.823	-51.975	-	-	0.40000	0.20000
9	P-Q	0.830	-57.772	-	-	0.35000	0.20000
10	P-Q	0.814	-50.116	-	-	0.70000	0.20000

TABLE C-1. The solution of all unknowns for 10 bus tested system

RESULT OF POWER FLOW CALCULATION

# of iterations = 6

BUS NO.	TYPE	VOLTS	ANGLE	**-----GENERATION-----**		**-----LOAD-----**	
				P(p.u)	Q(p.u)	P(p.u)	Q(p.u)
1 .	SW	1.080	0.000	4.80769	3.30281	0.00000	0.00000
2 .	P-V	1.020	-50.408	1.50000	1.92042	0.00000	0.00000
3 .	P-Q	0.757	-17.677	-	-	0.85000	0.30000
4 .	P-Q	0.947	-4.492	-	-	0.35000	0.25000
5 .	P-Q	0.744	-18.383	-	-	0.75000	0.45000
6 .	P-Q	0.733	-19.570	-	-	0.75000	0.25000
7 .	P-Q	0.811	-49.405	-	-	0.65000	0.30000
8 .	P-Q	0.923	-51.975	-	-	0.40000	0.05000
9 .	P-Q	0.830	-55.772	-	-	0.85000	0.40000
10 .	P-Q	0.824	-60.146	-	-	0.70000	0.30000

TABLE C-1 The solution of all unknowns for 10 bus tested system

BUS	ADMITTANCE	I	** -- LINE FLOW -- **		** -- LINE LOSS -- **	
			P(p.u)	Q(p.u)	P(p.u)	Q(p.u)
1- 3	1.73-j 4.31	1.979	1.7410	1.2403	0.3145	0.7825
1- 4	10.00-j20.00	3.456	3.0667	2.1267	0.2388	0.4776
2- 8	1.84-j 7.48	0.777	-0.3758	-0.6981	0.0187	0.0761
2- 9	1.13-j 4.48	0.964	-0.5776	-0.7960	0.0493	0.1952
2-10	0.70-j 2.80	0.722	-0.5466	-0.4939	0.0438	0.1753
3- 5	0.82-j 2.19	0.038	0.0235	0.0164	0.0002	0.0006
3- 6	10.00-j20.00	0.773	0.5530	0.1900	0.0119	0.0239
4- 5	0.96-j 4.80	1.408	1.0168	0.8629	0.0794	0.3970
4- 7	0.45-j 2.40	1.669	1.4611	0.6035	0.2102	1.1208
5- 6	5.00-j15.00	0.298	0.2107	0.0677	0.0018	0.0053
7- 8	1.40-j 5.60	0.681	-0.0624	0.5493	0.0195	0.0780
7- 9	1.84-j 7.48	0.716	0.5385	-0.2186	0.0159	0.0647
9-10	0.93-j 3.74	0.244	0.2010	-0.0240	0.0038	0.0150

TABLE C-2 The list of line flows and line losses for 10 bus tested system

## REFERENCES

- [1] Philip Rabinowitz (1970). Numerical method for nonlinear algebraic equation, Gordon and Breach, London.
- [2] Thomas R. Cuthbert, Jr (1987). Optimization using personal computer, Wiley, New York.
- [3] A. Iserles and M.J.D. Powell (1987). The state of art in numerical analysis, Oxford University Press, New York.
- [4] George D. Byrne and Charles A. Hall (1973). Numerical solution of system of nonlinear algebraic equation, Academic Press, New York.
- [5] C. Phillips and B. Cornelius (1986). Computational numerical methods, Ellis Horwood, Chichester.
- [6] Germund Dahlquist and Ake Bjorck. Numerical methods, Prentice-Hall, New Jersey.
- [7] John L.I. Morris (1983). Computational methods in elementary numerical analysis, Wiley, New York.
- [8] Richard W. Daniels (1978). An introduction to numerical methods and optimization techniques, North-Holland, New York.
- [9] V.L Zaguskin (1961). Handbook of numerical methods for the solution of algebraic and transcendental equations, Pergamon Press, London.
- [10] George L. Kusic (1986). Computer-aided power system analysis, Prentice-Hall, New Jersey.
- [11] G.T Heydt (1986). Computer analysis methods for power systems, Macmillan, New York.
- [12] William D. Stevenson, Jr (1982). Elements of power system analysis, McGraw-Hill, New York.
- [13] Stagg and Ei-Abiad (1968). Computer methods in power system analysis, McGraw-hill, New York.
- [14] The Institute of Electrical and Electronics Engineers, Inc. IEEE Recommended Practice for Industrial and Commercial Power System Analysis

- [15] S.C Tripathy, G.D Prasad, O.P Malik and G.S Hope. "Load-flow solution for ill-conditioned power system by a Newton-Like method", IEEE Trans., 1982, pp 3648-3657