Load Flow Studies by Fast Decoupled Method

P-148

Project Report

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN ELECTRICAL & ELECTRONICS ENGINEERING OF BHARATHIAR UNIVERSITY, COIMBATORE

1992-93

BY

G. V. Arun Gandhi
P. Subramaniam
K. Mohanraj

GUIDED BY

Miss. N. Umadevi B.E.,

Department of Electrical and Electronics Engineering
Kumaraguru College of Technology
Coimbatore-641 006

ACKNOWLEDGEMENT

We would like to express our heartiest thanks and gratitude to our guide **Ms.N.UMA DEVI.B.E.,** for her excellent guidance towards the successful completion of this project.

We are also very greatful to **Dr.K.A.PALANISWAMY,B.E., M.Sc.(Engg.), Ph.D., M.I.S.T.E., C Eng.(I), F.I.E.,** Professor and Head of the Department of Electrical and Electronics Engineering, for his valuable suggestions and encouragement throughout the project.

We are also indebted to **Prof.P.SHANMUGAM, M.Sc.(Engg), M.S.(HAWAI), M.I.E.E.E., M.I.S.T.E.,** Professor and Head of the department of CSE.

We would like to express our thanks to **Major.T.S.RAMAMURTHY, B.E., M.I.S.T.E.,** the principal, for his encouragement.

SYNOPSIS

This project aims at developing a software for load flow studies by Fast Decoupled method. The program is written in FORTRAN77 and tested for number of power systems on LAN computer system. The detailed results for a 4 bus system is presented. This software proves to be very flexible and user friendly. It has wide applications. The limitations while implementing the available "Fast Decoupled " theory, in a digital computer are discussed.

CONTENTS

NOTATIONS

$\Delta P_k + j \Delta Q_k$ = Complex power mismatch at bus k,

where,

$$\Delta P_k = P_k^{sp} - U_k \sum_{m \in k} U_m (G_{km} \cos\theta_{km} + B_{km} \sin\theta_{km})$$

$$\Delta Q_k = Q_k^{sp} - U_k \sum_{m \in k} U_m (G_{km} \sin\theta_{km} - B_{km} \cos\theta_{km})$$

$P_k^{sp} + jQ_k^{sp}$ = scheduled complex power at bus k.

θ_k , U_k = voltage angle, magnitude at bus k.

$\theta_{km} = \theta_k - \theta_m$

$G_{km} + jB_{km}$ = (k,m)TH element of bus admittance matrix [G]+j[B]

$\Delta\theta$, ΔU = voltage angle, magnitude corrections.

$m \in k$ signifies that bus m is connected to bus k, including the case m=k, and [] signifies vector or matrix.

max $|\Delta P|$, max $|\Delta Q|$ = largest absolute element of $[\Delta P]$, $[\Delta Q]$, max$|e_v|$, max$|e_s|$ = largest absolute bus voltage magnitude error, branch MVA-flow error, respectively.

CHAPTER I

INTRODUCTION

1.1 LOAD FLOW ANALYSIS

A load flow study is the determination of the voltage, current, Power factor or reactive power at various points in an electric network under existing or contemplated conditions of normal operation. Load studies are essential in planning the future development of the system because satisfactorey operation of the system depends on knowing the effects of interconnections with other power system of new loads, new generating station and new transmission lines before they are installed.

A load flow solution of the power system requires mainly the following steps.

(i) Formulation of the network equations.
(ii) Suitable technique for solution of the equations.

There are numerous techniques available for the load flow studies. Gauss seidal and Newton Raphson methods are considered most important of

them. The Gauss seidal method is known for its less core space but takes many iterations. The Newton Raphson method is good in convergence but consumes much of core memory. The FDLF method consumes less core space and equally good in convergence. The comparison of the various methods can be described as below.

1.2 COMPARISON AND CHOICE OF LOAD FLOW METHODS

Load-calculations are performed in system planning, operational planning, and operational control. The choice of a solution method of practical application is frequently difficult. It requires a careful analysis of merits and demerits of many available methods in such respects as storage, speed and convergence charecteristics, to name but the most obvious, and to relate these to the requirements of the specific applications and computing facilities. The difficulties arise from the fact that no one method possesses all the desirable features of the others. For routine solutions Newtons method has gained widespread popularity. However it is limited for small-core applications where the weakly-convergent

Gaus-Seidal-type is the most economical, and it is not as fast as newer methods such as the FDLF method, for approximate repetitive solutions such as in AC security monitoring.

As described, a trade off is to be made, between certain better qualities of a method, & some other good qualities of other methods. Newton's method has very good converging scale, but lags behind the other methods, considering the time per iteration. Though the time per iteration of an FDLF method is not as fast as the Gauss family of methods, it proves to be better converging as its convergence is geometrical. It is worth mentioning that it is faster than the Newton's method due to the fact that the Jacobian matrix need not be formed. The B' and B'' matrices formed in the FDLF method is at the most, half of the order of the Jacobian matrix. So this reduces the iteration time and effort considerably. But the tradeoff here is the accuracy being overlooked a little for reducing the precious time. But the FDLF method is equally reliable, due to the fact that the elements of [B'] and [B''] are fixed approximation to the tangents of

the defining functions [ΔP / |V|] and [ΔQ / |V|], and are not sensitive to any humps in the defining functions. On considering the ease of programming and very low core required; FDLF proves better in some situations, such as dealing with, systems having long and short lines terminating on the same bus with long to short ratios above 1000. The Fast decoupled method does not require any acceleration factors. For algorithms with a Gaussian Skeleton the time per iteration, as well, will increase with the increase in number of buses present in the system.

1.3 FAST DECOUPLED METHOD

The Fast Decoupled method describes a simple, very reliable and extremely fast load flow solution, when compared to its parent methods viz., the Newton- Raphson and Gauss methods. This method has wide range of practical applications. It is attractive for accurate off - line and on - line routine and contingency calculations for networks of any size, and can be implemented efficiently on computers with restrictive core store capacities. This method is theoretically capable and suitable to

perform on a series of practical problems of upto 1080 buses. But practical constraints, such as matrix inversion, limit the number of buses that can be implemented by this method. The theoretical limit can be obtained by improving the algorithms used for the ancillary routines. A solution of within 0.01 MW / MVAR maximum bus mismatches are normally obtained in 4 to 7 iterations, each iteration being equal in speed to 3/2 Gauss-Seidel iterations or 1/5 of a Newton Raphson iteration. Correlations of general interest between the power-mismatch convergence criterion and actual solution accuracy are obtained.

1.4 TYPES OF LOAD FLOWS SOLUTIONS

The analysis can be classified as "off-line" or "batch" analysis and "on line" Power flow analysis. The former is the one in which executions of power flow analysis are performed months or years ahead of the actual situation and the later involves the periodically executed program in the degital computer which is monitoring and controlling the power system. This 'on line' type requires the updating of the system data now and then. It takes averaged or processed real-time measured data for the

P,Q or P,V conditions at buses in order to calculate bus voltages and phase angles for the entire network. This project assumes, for its implementation, that loads and hence generations are fixed at a particular value over a suitable period of time.

1.5 NEED FOR COMPUTER SIMULATION OF A POWER SYSTEM

Prior to the computer revolutions, the load flow studies for complex networks couldnot have been dreamt to be done, mannualy, in papers or some other theoritical means. The only possible way was to simulate the existing system with prototypes of the system components, with similar characteristics. In such 'AC calculating boards', the addition or expantion of the real network is to be immediately represented with similar prototypes. More over, in ac calculating boards, setting up the connections, making adjustments and reading the data were tedious and time consuming. To avoid these problems, digital computers can be used to compute the complex calculations and to fruit with the fore said results of load flows.

CHAPTER II

FAST DECOUPLED LOAD FLOW METHOD

2.1 SUPERIORITY OF FAST DECOUPLED LOAD FLOW METHOD.

The Fast Decoupled Load Flow method combines the advantages of the other methods and is simpler, faster and reliable. A detailed explanation is as given below.

Numerical methods are generally at their most efficient when they take advantage of the system being solved. Hence, for example, the exploitation of network sparsity by ordered elimination and skillful programming in Newton and other methods has been very important. Attention has been given to the exploitation of the loose physical interaction between MW and MVAR flows in a power system, by mathematically decoupling the MW-θ and MVAR-V calculations.

The Fast decoupled load flow method is a

rational integrations of some of these ideas. It combines many of the advantages of the existing good methods. This algorithm is simpler faster and as reliable as Newtons methods, and has lower storage requirements for 'entirely in-core' solutions. Using a small number of core-disk block transfers its core requirements are similar to those of the Gauss-Seidel method. The method is equally suitable for routine accurate load flows as for outage-contingency evaluation studies performed on or off line.

2.2.CLASSIFICATION OF BUSES

In a power system each bus or node is associated with four quantities, real and reactive powers, bus voltage magnitude and its phase angle. In a load flow solution two out of the four quantities are specified and remaining two are required to be obtained through the solution of the equations. Depending upon which quantities have been specified, the buses are classified in the following three categories.

2.2.1 LOAD BUS :

At this bus the real and reactive components of power are specified. It is desired to find out the voltage magnitude and phase angle through the load flow solution.

2.2.2 GENERATOR BUS OR VOLTAGE CONTROLLED BUS.

Here the voltage magnitude and real power are specified. It is required to find out the reactive power generations and the phase angle of the bus voltage.

2.2.3 SLACK, SWING or REFERENCE BUS :

The losses in power system remain unknown until the load flow solution is complete. It is for this reason that one of the generator buses is made to take the additionsl real and reactive power to supply transmission losses. That is why this type of bus is also known as the slack or swing bus. At this bus, the voltage magnitude and phase angle are specified whereas real and reactive powers are obtained through the load flow solution. The phase angle of the voltage at the slack bus is usually taken as the reference.

2.3 BASIC ALGORITHM

We know that,

$$P_k = P_k^{sp} - V_k \quad V_m(G_{km}\text{Cos}\ \theta_{km} + B_{km}\text{Sin}\ \theta_{km}) \ldots (2.1)$$

$$Q_k = Q_k^{sp} - V_k \quad V_m(G_{km}\text{Sin}\ \theta_{km} - B_{km}\text{Cos}\ \theta_{km}) \ldots (2.2)$$

The well-known Newton method is taken as the convenient and meaningful starting point for the derivation. The Newton is the formal application of a general algorithm for solving nonlinear equations and constitutes successive solutions of the spares real Jacobian matrix equation.

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ J & L \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta V/V \end{bmatrix} \quad (2.3)$$

The first step in applying the MW-θ/MVAR-V decoupling principle is to neglect the coupling sub matrices [N] and [J] in eqn.(2.3), giving two separated equations.

$$[\Delta P] = [H]\ [\Delta\theta] \quad \ldots\ldots\ldots (2.4)$$

$$[\triangle Q] = [L]\ [\triangle V/V] \quad \ldots\ldots\ldots (2.5)$$

where,

Eqn.s (2.4) and (2.5) may be solved alternatively, by reevaluating [H] and [L] at each iteration, but further physically justifiable simplifications may be made. The following assumptions are considered almost valid in practical power system calculations.

$$\cos \theta_{km} \approx 1$$

$$G_{km} \sin \theta_{km} << B_{km}$$

$$Q_k << B_{kk} V_k^2$$

so that good approximations to (2.4) and (2.5) are,

$$[\triangle P] = [V.B'.V]\ [\triangle \theta] \quad \ldots\ldots\ldots (2.6)$$

$$[\triangle Q] = [V.B''.V]\ [\triangle V/V] \quad \ldots\ldots\ldots (2.7)$$

At this stage of derivation the elements of the matrices [B'] and [B''] are strictly elements of [-B]. The decoupling process and the final algorithmic forms are completed by

(a) omitting from [B'] the representation of those network elements that predominantly affect MVAR

flows, ie., shunt reactances and off-nominal in-phase transformer taps.

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

(c) taking the left-hand V terms in (2.6) and (2.7) on to the left hand sides of equations, and then in (2.6) removing the influence of MVAR flows on the calculation of $[\triangle\theta]$ by setting all the right-hand V terms to 1 Pu. It is to be noted that the V terms on the left-hand sides of (2.6) and (2.7) affect the behaviours of the defining functions and not the coupling.

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

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

$$[\triangle P/V] = [B'] [\triangle\theta] \quad \text{.....(2.8)}$$

$$[\triangle Q/V] = [B''] [\triangle V] \quad \text{.....(2.9)}$$

Both [B'] and [B''] are real, sparse and have the structures of [H] and [L] respectively. Since they contain only network admittances they are constant and need to be triangulated once only at the beginning of the analysis. [B''] is symmetrical so that only its upper triangular factor may be stored. In case phase shifters are absent [B'] is also symmetrical.

The immediate appeal of (2.8) and (2.9) is that very fast repeat solutions for $[\Delta\theta]$ and $[\Delta V]$ can be obtained using the constant triangular factors of [B'] and [B'']. These solutions may be iterated with each other in some defined manner towards the exact solution. ie.,when $[\Delta P/V]$ and $[\Delta Q/V]$ are zero. But absolute zero is not possible in practical situations. So a value nearer to zero, say 0.001, may be taken as the converging target.

2.4 GENERATOR Q LIMITS.

Once a load-flow solution is moderately converged, any PV-bus Q-limit violations can be corrected. Provision must be made for subsequent interactive effects, i.e. MVARs backing off limits and new Q violations.

Two approaches are available. In the first, each violating bus is explicitly converted to PQ type so that the MVAR output is held at the limiting value. The bus remains a PQ type during the rest of the solution unless at some stage it can be re-converted to a PV type at the original voltage magnitude without the violation. In the fast decoupled program, converting any number of bus types at one time involves retrianulating [B''].

The second approach is to correct the voltage of each violating PV bus k by an amount ΔV_k at each following iteration to reduce the error $Q_k = (Q_k^{limit} - Q_k)$ to zero. The convergence of this process is rapid when an approximate sensitivity factor relating and is used thus.

$$\Delta V_k = S_k \cdot \Delta Q_k / V_k \qquad \text{----------(2.10)}$$

If S_k is defined according to (2.9) it is the diagonal element corresponding to bus k in the inverse of matrix [B''] augmented by the previously absent row and column for bus k. For an operational network each S_k may be stored permanently and updated only for significant systems configuration changes. The correction (2.10) ceases to be applied if at some stage in the solution the value of V_k is restored to or goes beyond its original value.

Both approaches are said to be equally effective. The sensitivity method usually takes more iterations, but against this it requires no retriangulations for limit enforcement and back-off, and is simpler .

2.5 SOLUTION SPEED AND STORAGE

Central to the fast decoupled method is the use of efficient sparsity programmed routines for the triangulation of the sparse, real, symmetric matrices

[B'] and [B''] and the subsequent solutions of (2.8) and (2.9) by forward and backward substitutions. If, as is becoming more widespread, such routines are available as standard packages, the method is not difficult to program efficiently.

At the beginning of the load-flow study the system, excessive fill up in the table of factors during the triangulation of [B'] may be avoided by rearrangeing the data properly. It can be shown that this bus ordering remains equally suitable for [B''] with the P.V buses by-passed in the ordering list. If the system is very large or if consecutive load-flow solutions are to be calculated then it is advantageous to minimise fill-up as far as possible by using a good ordering scheme. Since [B'] and [B''] remain unchanged, triangulation routines that perform the ordering during elimination are at no great disadvantage.

The triangulation routine is programmed to take account of matrix symmetry so that only the upper-triangular factors of [B'] and [B''] are stored for

use in the solution routine. Using dynamic re-ordering according to row sparsity, the ordering, triangulation and solution times each vary roughly linearly with network size. For a typical well-developed network with n buses and b branches the number of elements in the upper-triangular factor of [B'] is about 3(n+b)/2. Depending on the number and location of PV buses, the triangulation and solution processes for [B''], in some cases considerably so.

The calculations of the vectors $[\Delta P/V]$ and $[\Delta Q/V]$ can each be performed rapidly in a single sweep of the branch admittance list. For each series-branch connecting buses k and m, it can be seen from (2.1) and (2.2) that where appropriate, the terms $V_k V_m G_{km}$ and $V_k V_m B_{km}$ can be used directly in accumulating both ΔP_k and ΔP_m, or ΔQ_k and ΔQ_m. Also, it is noted that $\text{Sin } \theta_{km} = - \text{Sin } \theta_{mk}$ and $\text{Cos } \theta_{km} = \text{Cos } \theta_{mk}$. These trigonometrical functions are calculated using the library functions of FORTRAN 77, which have been found to give final load-flow results indistinguishable from those obtained with the accurate function evaluations. If storage is not

critical, the sine and cosine terms calculated during the construction of $[\Delta Q/V]$ can be stored for use in the next construction of $[\Delta P/V]$. From a flat voltage start, each sine and cosine term can be initialised to 0 and 1 respectively for the first solution of (2.8).

The storage requirements of the fast decoupled method are about 40 % less than those of Newton's method. This saving is reduced somewhat if the sine and cosine terms are stored. Apart from simple subroutine overlaying, core storage can be economised by reading certain selected vectors from disk when they are required and writing them back after the relevant subroutine. These block transfers are performed a limited number of times during a solution, and should not degrade the method's speed too severely. Using this scheme, the core storage requirements of the method are about the same as those of the Gauss-Seidel method.

Network tearing for load-flow calculations is not usually economical in computing speed when ordered elimination is used for solving the sparse network

equations. However, if a piecewise approach is desirable for other reasons, e.g. storage, any of the available Y-matrix decomposition methods can be applied to give a series of smaller constant [B'] and [B''] matrices. It is to be noted that since branches connected to PV buses are absent from [B''], a certain amount of network tearing is already inherent in this matrix.

2.6 ITERATION SCHEME

Of the available types of iteration schemes, the best and simple would be to have two different flags for convergence of ΔP and ΔQ values and testing them in each iteration. Each iteration cycle comprises one solution for 0 to update the [v]. Seperate convergence tests are used for eqn.s (2.8) and (2.9) with the criteria

$$\max |\Delta P| < Cp$$

$$\max |\Delta Q| < Cq$$

The flow diagram of the process is given in the fig.(2.1). The convergence testing logic

permits the calculation to terminate after a [$\Delta\theta$] solution. It is also possible to terminate after more than one consecutive [$\Delta\theta$] or [ΔV] solution. If [ΔQ] or [ΔP] respectively donot need converging further.

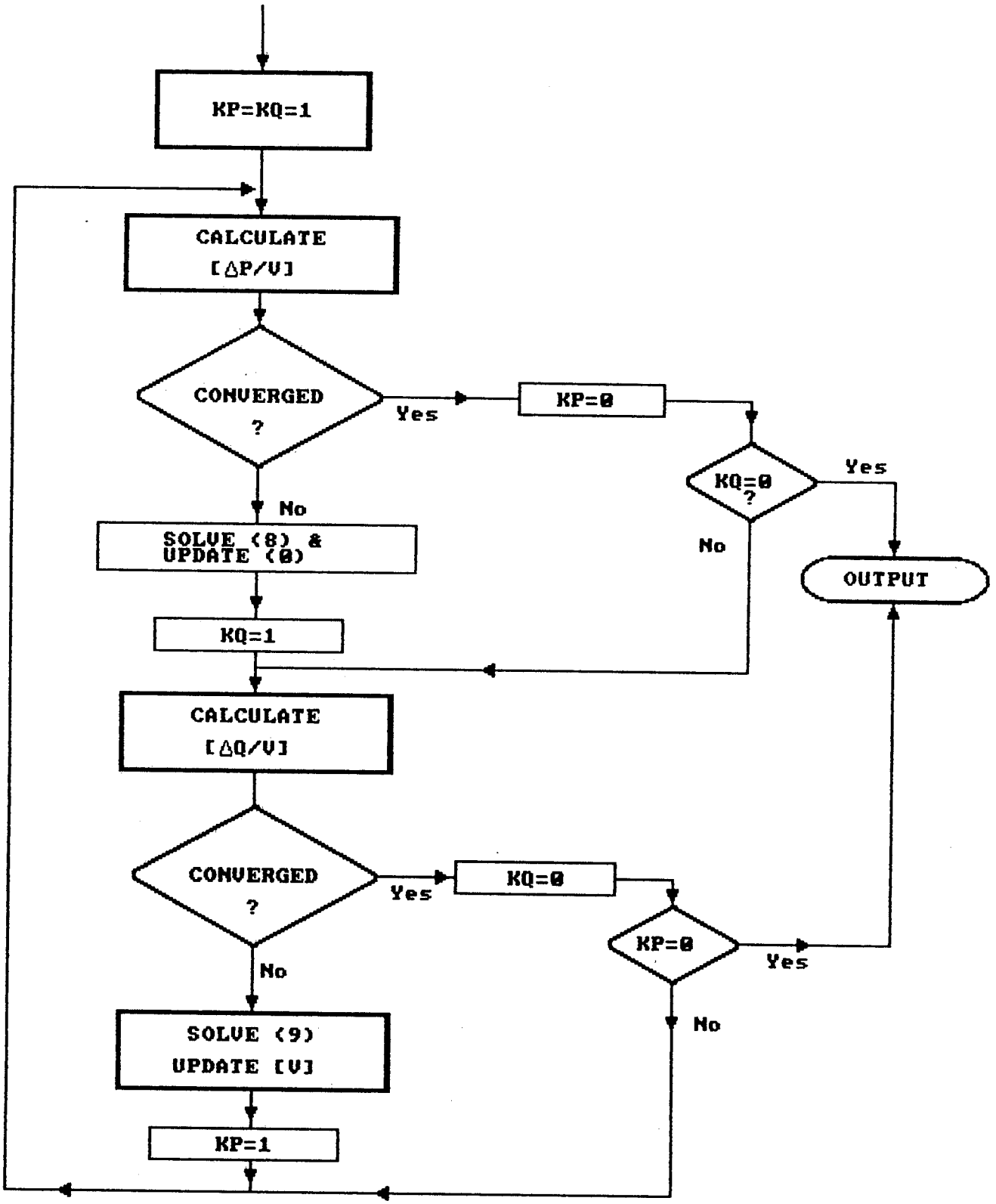


Fig.2.1 Flow chart representing iterative scheme

2.7 CONVERGENCE OF FDLF

The iteration process of the fast decoupled method has three distinct components, each with its own convergence characteristics.

(a). The solution of $[\triangle P/V] = 0$ for using (2.8)

(b). The solution of $[\triangle Q/V] = 0$ for $[V]$ using (2.9), and

(c) The interactive effects of V changes and θ changes on the defining function $[\triangle P/V]$ and $[\triangle Q/V]$ respectively.

Equations (2.8) and (2.9) are both Newton-like except that instead of re-evaluating the true jacobian matrix, tangent-slopes to the left-hand functions at each iteration, fixed approximated tangent slopes $[B']$ and $[B'']$ are used. The algorithms therefore correspond to the generalised 'fixed-tangent' or 'constant slope' method which has geometric convergence. For reasonably-behaved functions, this method is very reliable and if, as in the

present application, the fixed tangent-slopes correspond closely to the Jacobian matrix at the initial point (for flat voltage start), the initial convergence is very rapid. The process does not 'home in' as fast as the quadratic Newton method as the exact solution is approached, but load flow solutions are rarely required to very high accuracy.

The fixed-tangent method is not thrown, of-course, when it encounters 'humps' in the defining functions, whereas Newton's method tends to be mis-directed even to the extent of divergence from the desired solution in such cases. In the decoupled load flow problem, the changing V values during the solution some times have the effect of osscillatory shift in the shapes of multi dimensional surface [Δ P/V] as functions of [$\Delta\theta$] and likewise for θ - values on [Δ Q/V] as functions of [V]. Since doesnot represent MVAR conditions it correspond to fixed tangent directions that take no cognisance of these shifts and are therefore not affected by V-oscillations. Likewise for [B''] in relation to MW conditions.

2.8 CONVERGENCE PATTERN

A typical convergence pattern for the method is shown in fig.(2.2) in terms of the largest mismatches at every half iteration. Each iteration of the program produces rapid reduction of [P] and [Q].

The effect of θ changes on the MVAR flows is to increase max[ΔQ]. The effect of V changes on MW flows is less pronounced because the active loss due to MVAR flows are normally smaller than the reactive loss due to MW flows. Overconverging [ΔP] at any stage produces a severe increase in [Δ Q], thereby slowing down the overall convergence rate. The FDLF method has geometric convergence characteristics.

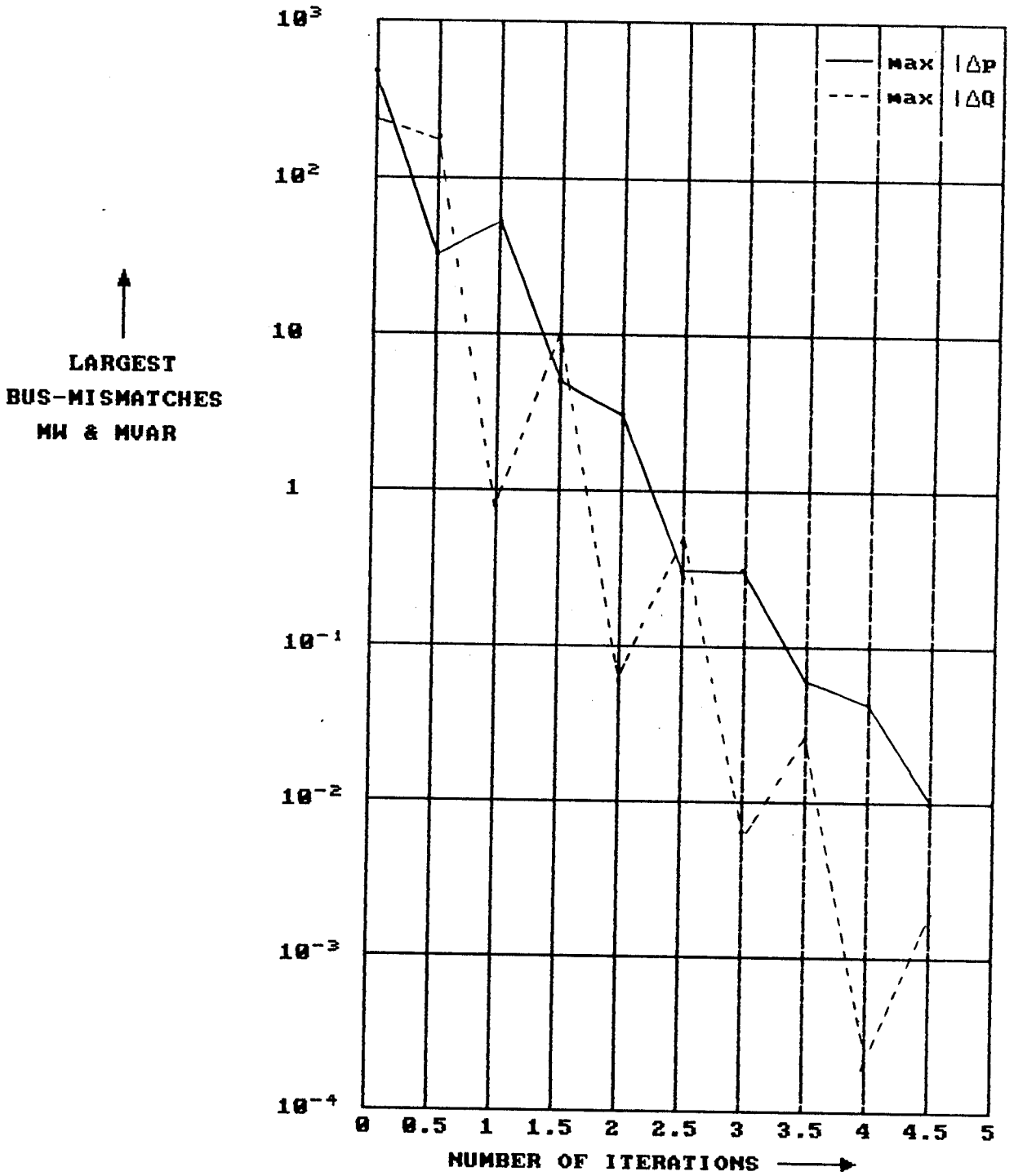


Fig. 2.2 Graph representing Convergence Pattern

2.9 LIMITATIONS OF FAST DECOUPLED METHOD

Though the latest of all the present methods the Fast decoupled load flow method has got its own disadvantages. This method is not as strongly convergent as the Newton Raphson method, and furthermore, the FDLF may diverge for some power systems if transmission line reactances are not dominating. In fact dummy lines or dummy buses may be introduced inorder to achieve the reactance dominated conditions. For the present day large and complex power systems (with 1000 buses or more) it is impossible to invert the [B'] and [B''] matrices as the process would require enormous core memory and computing time.

The remedy may come in through two possible ways. The modern computer languages that are to be developed in future may be equipped well to handle the more complex situations (as the 1000 bus system), or the system engineers are to ponder system algebra for faster, newer and simpler methods to solve the problem.

CHAPTER III

DEVELOPMENT OF SOFTWARE

3.1 CHOICE OF LANGUAGE AND THEIR LIMITATIONS

As the project deals with complex quantities such as voltages, currents and other line parameters, a computer language with good feasibility, flexibility and, most important, with built in library functions that can deal with the complex operations, is idealy suited for the purpose. But the flexibile, improved languages such as C, are not furnished with such library functions such as arithmatic functions in complex numbers. More over, the parameter passing of matrices, comprising of such complex numbers, between the subroutines or functions of the program causes a difficulty as the real and imaginary parts are to be passed seperately. This project uses the FORTRAN77 language, which is considered to be outdated today. Inspite of the lack of flexibility, it comprises the complex operations, which makes it more suitable for complex operations. Here the parameter passing is simple as an 'n' dimentional array of complex numbers can be easily done. The primary aim of the project must be, in any way, to reduce time

consumption, to attain flexible operation and so on. Faster languages such as pascal or C need to be supported with user defined subroutines (or library functions) for such operations. This again will lead to time lag. Parallely, if the 2 dimensional array declaration of real and imaginary matrices are done seperately, it will result in wastage of core space. Even in Fortran, the scientific purpose language, the trigonometric functions being delt with in radians cause problems in determining the exact direction of a vector in the complex (x + j w) plane. A seperate routine had to be developed to determine the modulus and direction of the vector from the reference, from the given number. Ultimately one has to go in for lesser flexibility for want of time and core capacity.

3.2 FLOW CHART

The sequence of operations being performed by the program can be visualised from the flow chart included in this chapter. It clearly describes the logical steps performed, stage by stage,

FLOW CHART

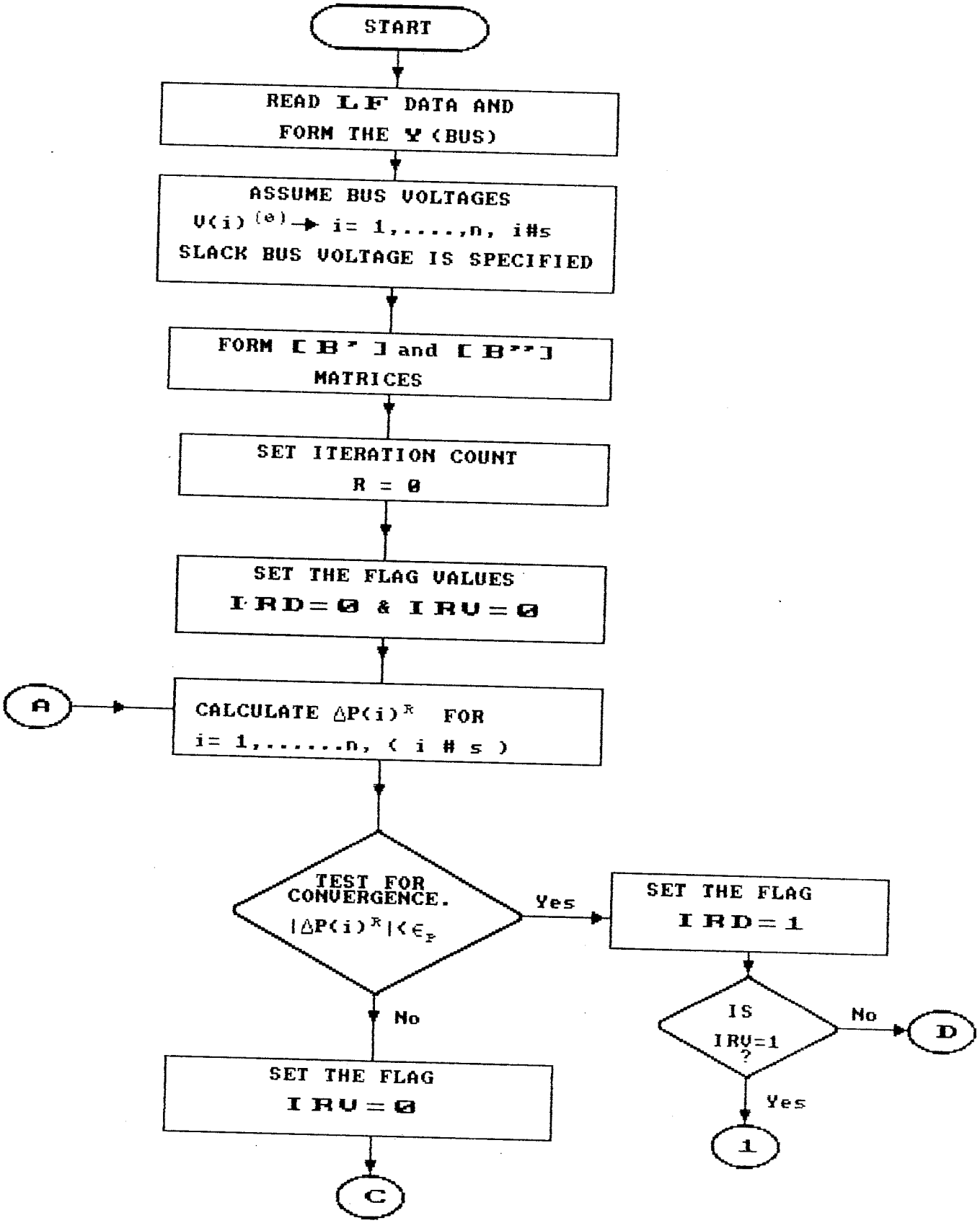
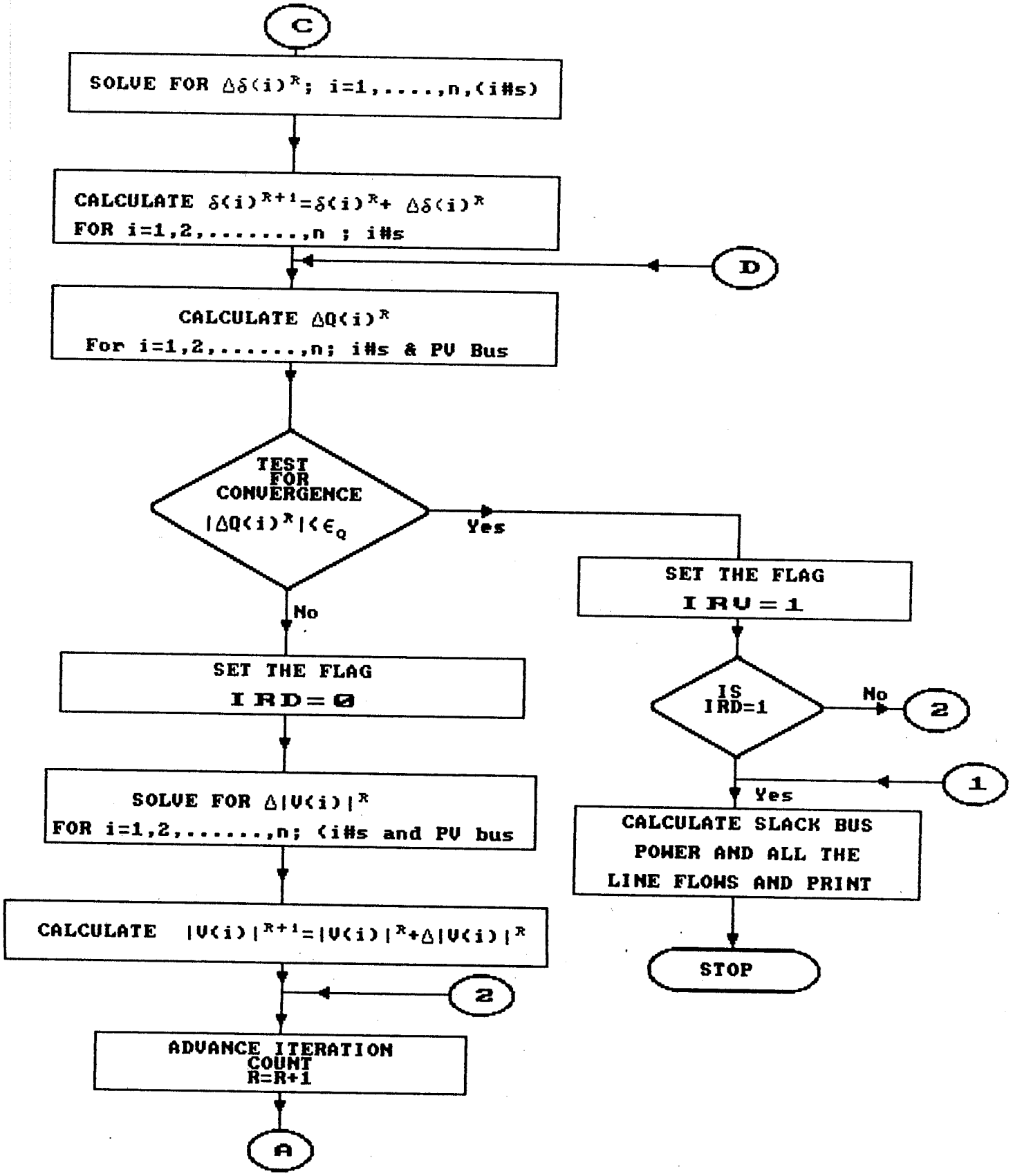


Fig. 3.1. Flow chart for FDLF method


C
SOLVE FOR Δδ(i)^R; i=1,....,n,(i#s)
CALCULATE δ(i)^(R+1)=δ(i)^R+ Δδ(i)^R
FOR i=1,2,.....,n ; i#s
D
CALCULATE ΔQ(i)^R
For i=1,2,.....,n; i#s & PV Bus
TEST FOR CONVERGENCE
|ΔQ(i)^R|<ε_Q
Yes
No
SET THE FLAG
IRD=0
SOLVE FOR Δ|V(i)|^R
FOR i=1,2,.....,n; (i#s and PV bus
CALCULATE |V(i)|^(R+1)=|V(i)|^R+Δ|V(i)|^R
2
ADVANCE ITERATION COUNT
R=R+1
A
SET THE FLAG
IRV=1
IS IRD=1
No
2
1
Yes
CALCULATE SLACK BUS POWER AND ALL THE LINE FLOWS AND PRINT
STOP

3.3 PROGRAM LISTING.

The complete software in FORTRAN77 is included in this chapter. The program requires inputs in per unit quantities. User is free to use either impedances or admittances, for feeding the line parameters. For mixed bus systems the inputs are to be fed, considering the particular bus to be a generator bus. Options are included thereon to feed the corresponding load values. Step by step outputs can be obtained on execution of the program.

```
C       NETWORK PROG.
        DIMENSION B(15,15),BP(15,15),BQ(15,15),BPINV(15,15),BQINV(15,15)
        DIMENSION PMUL(15,15),QMUL(15,15),P(15),Q(15),DEL(15)
        DIMENSION DELV(15),V(15),PS(15),QS(15),DELTA(15),DELTAUD(15)
        DIMENSION DELP(15),DELQ(15),PC(15),QC(15),DELPPP(15),DELQQQ(15)
        DIMENSION PF(15,15),QF(15,15)
        COMPLEX Y(15,15),YB(15,15),YSH(15,15),VRECT(15),XIRECT(15,15)
        COMPLEX LOSS(15,15),YB1(15,15),YSH1(15,15)
        REAL PFLOW(15,15),QFLOW(15,15)
        CHARACTER FL(15),Z,X,X1
        INTEGER COU,COU1
        CALL INPUT(N,YB1,YSH1,ELIMIT,Z,X,X1)
        CALL ALTER(N,YB1,YSH1,Z,X,X1,YB,YSH)
        CALL INPUT2(N,P,Q,V,FL)
        CALL YBUS(N,YB,YSH,Y)
        WRITE(*,*)
        WRITE(*,*)('-',I=1,64)
        WRITE(*,*)'                          MATRIX  OUTPUTS'
        WRITE(*,*)('-',I=1,64)
        WRITE(*,*)'THE Y BUS IS'
        CALL OP1(N,Y)
        CALL BBUS(N,Y,B)
        WRITE(*,*)'THE B BUS IS'
        CALL OP2(N,B)
        CALL SPECI(N,P,Q,PS,QS)
        CALL BPP(N,B,BP)
        WRITE(*,*)'THE BP BUS IS'
        CALL OP2(N-1,BP)
        CALL BQQ(N,FL,B,BQ,K)
        WRITE(*,*)'THE BQ BUS IS'
        CALL OP2(K,BQ)
        CALL INV(N-1,BP,BPINV)
        CALL INV(K,BQ,BQINV)
        PAUSE
        DO 13 I=1,N
        DELTA(I)=0
  13    CONTINUE
        IR=1
        IRD=0
        IRV=0
        WRITE(*,102)IR
  102   FORMAT(5X,I1,'st   ITERATION  RESULTS          ')
        WRITE(*,*)('-',I=1,64)
  66    CALL CALCP(N,V,Y,DELTA,PC)
        CALL DELPP(N,PS,PC,V,DELP,DELPMAX,DELPPP)
        IF(DELPMAX.LT.ELIMIT) THEN
        IRD=1
        IF(IRV.EQ.1)GOTO 95
        GOTO 75
        ELSE
        IRV=0
        ENDIF
        CALL MUL1(N-1,BPINV,DELPPP,DEL)
        DO 20 I=2,N
C       WRITE(*,30)I,DEL(I-1)
  30    FORMAT(1X,'DELDELTA(',I2,')',1X,'IS',1X,F4.2)
```

```
      DELTAUD(I)=DELTA(I)+DEL(I-1)
  20  CONTINUE
  75  CALL CALCQ(N,V,Y,DELTA,QC)
      CALL DELQQ(N,QS,QC,V,FL,DELQ,DELQMAX,DELQQQ)
      IF(DELQMAX.LT.ELIMIT)THEN
      IRV=1
      IF(IRD.EQ.1)GOTO 95
      GOTO 64
      ELSE
      IRD=0
      ENDIF
      CALL MUL1(K,BQINV,DELQQQ,DELV)
      WRITE(*,*)'THE DELV VALUES ARE ...'
      COU=0
      DO 40 I=2,N
      IF(FL(I).EQ.'N')THEN
      COU=COU+1
      WRITE(*,50)I,DELV(COU)
  50  FORMAT(1X,'DELV(',I2,')',1X,'IS',1X,F6.4)
      V(I)=V(I)+DELV(COU)
      ENDIF
  40  CONTINUE
      DO 34 I=1,N
      DELTA(I)=DELTAUD(I)
  34  CONTINUE
  64  WRITE(*,33)IR
  33  FORMAT(1X,'THE DELTA & VOL. AT THE END OF ITERATION ',I2,' ARE')
      CALL OPITR(N,DELTA,V,FL)
      IR=IR+1
      IF(IR.EQ.10) GOTO 95
      GOTO 66
  95  WRITE(*,*)'DELP AND DELQ VALUES CONVERGED'
      CALL PORE(N,V,DELTA,VRECT)
      DO 220 I=1,N
      WRITE(*,*)'VRECT',I,VRECT(I)
 220  CONTINUE
      CALL LINVAL(N,VRECT,Y,YSH,XIRECT,PFLOW,QFLOW,LOSS)
      DO 111 K=1,15
 111  WRITE(*,*)
      WRITE(*,*)('_',I=1,64)
      WRITE(*,*)
      WRITE(*,*)'                                    LOAD FLOW RESULTS'
      WRITE(*,*)('_',I=1,64)
      WRITE(*,*)
      WRITE(*,*)
      DO 150 I=1,N
      DO 150 J=1,N
      IF(I.EQ.J) GOTO 150
      WRITE(*,*)
      WRITE(*,99)I,J
  99  FORMAT(3X,'Bus Code',I2,' --',I2)
      WRITE(*,*)'  ---------------'
```

```
      WRITE(*,*)'Current=',XIRECT(I,J)
      WRITE(*,*)'Real Power flow =',PFLOW(I,J)
      WRITE(*,*)'Reactive power flow =',QFLOW(I,J)
  150 CONTINUE
      WRITE(*,*)('_',I=1,64)
      STOP
      END


C     SUBROUTINE TO RECIEVE LINE VALUES
      SUBROUTINE INPUT(N,Y,YSH,ELIMIT,Z,X,X1)
      DIMENSION B(15,15)
      COMPLEX Y(15,15),YSH(15,15),COM,YT
      CHARACTER Z,X,X1
      WRITE(*,*)'Is the input an Admittance or an Impedance '
      WRITE(*,*)
   98 WRITE(*,*)'Please select your choice  ( A\I ) '
      READ(*,1)Z
    1 FORMAT(A1)
      IF(Z.EQ.'I'.OR.Z.EQ.'i') THEN
      WRITE(*,*)'Give the number of buses'
      READ(*,*)N
      WRITE(*,*)'Enter the Impedances in P.U.'
      DO 30 I=1,N
      DO 30 J=I+1,N
      WRITE(*,10)I,J
   10 FORMAT(1X,'Enter the value of Z ',I2,'-',I2)
      CALL COMIN(Y(I,J))
      IF(Y(I,J).EQ.(0,0)) GOTO 30
      Y(I,J)=1/Y(I,J)
      Y(J,I)=Y(I,J)
   30 CONTINUE
      GOTO 99
      ENDIF
      IF(Z.EQ.'A'.OR.Z.EQ.'a') THEN
      WRITE(*,*)'Give the number of Buses'
      READ(*,*)N
      WRITE(*,*)'Enter the Admittances in P.U.'
      DO 35 I=1,N
      DO 36 J=I+1,N
      WRITE(*,16)I,J
      CALL COMIN(Y(I,J))
      Y(J,I)=Y(I,J)
   36 CONTINUE
   35 CONTINUE
      GOTO 99
      ELSE
      WRITE(*,*)'Your choice is wrong'
      WRITE(*,*)'PLEASE REPEAT '
      GOTO 98
      ENDIF
   16 FORMAT(1X,'Enter the value of Y',I2,'-',I2)
   99 WRITE(*,*)'Do you have half line charging '
      WRITE(*,*)' Admittances (Y\N) '
      READ(*,22)X
```

```
    IF(X.EQ.'Y'.OR.X.EQ.'y') THEN
    WRITE(*,*)'ENTER YOUR CHOICE '
    WRITE(*,*)'  H -  HALF LINE CHARGING ADMITTANCE '
    WRITE(*,*)'  L -  LINE CHARGING ADMITTANCE  '
    READ(*,22)X1
22  FORMAT(A1)
    ELSE
    GOTO 44
    ENDIF
    IF(X1.EQ.'H'.OR.X1.EQ.'h') THEN
    DO Ø8 I =1,N
    DO Ø8 J=I+1,N
    WRITE(*,*)'ENTER THE HALF LINE CHARGING'
    WRITE(*,33)I,J
 33 FORMAT(1X,'ADMITTANCE OF BUS ',2I2)
    CALL COMIN(YSH(I,J))
    YSH(J,I)=YSH(I,J)
 Ø8 CONTINUE
    ELSE
    IF(X1.EQ.'L'.OR.X1.EQ.'l') THEN
    DO Ø9 I =1,N
    DO Ø9 J=I+1,N
    WRITE(*,*)'ENTER THE LINE  CHARGING'
    WRITE(*,33)I,J
    CALL COMIN(YSH(I,J))
    YSH(I,J)=YSH(I,J)/2
    YSH(J,I)=YSH(I,J)
 Ø9 CONTINUE
    ELSE
    WRITE(*,*)'Your choice is wrong'
    WRITE(*,*)'PLEASE REPEAT '
    GOTO 99
    ENDIF
    ENDIF
 44 WRITE(*,*)'Enter the value of Ephsalon "E" '
    READ(*,*)ELIMIT
    RETURN
    STOP
    END


    SUBROUTINE  ALTER(N,YB,YSH,Z1,X,X1,YB1,YSH1)
    COMPLEX YB(15,15),YSH(15,15),YB1(15,15),YSH1(15,15),YT
    CHARACTER Z1,X,X1,X2,C1,C2,C3,C4,C5,C6,C7,C8
    DO Ø3 I=1,2Ø
Ø3  WRITE(*,*)
    WRITE(*,54)
54  FORMAT(7X,'DO YOU WANT TO ALTER ANY LINE PARAMETER')
    WRITE(*,*)'        VALUES  CHOOSE  ( A\ N)'  '
    READ(*,11)X2
    IF(X2.NE.'A'.AND.X2.NE.'a')GOTO 51
    IF(Z1.NE.'I'.AND.Z1.NE.'i')GOTO 52
    WRITE(*,*)'Do you want alter any IMPEDANCE value ? (Y\N)'
```

```
      READ(*,11)C1
      IF(C1.NE.'Y'.AND.C1.NE.'y') GOTO 53
      DO 31 I=1,N
      DO 31 J=I+1,N
      WRITE(*,110)I,J
  110 FORMAT(1X,'The value of Z ',I2,'-',I2,' is ')
      IF(YB(I,J).EQ.0.0) GOTO 2
      YB(I,J)=1/YB(I,J)
    2 WRITE(*,*)YB(I,J)
      WRITE(*,*)'Do you want change this (Y \ N)'
      READ(*,11)C2
   11 FORMAT(1A)
      IF(C2.NE.'Y'.AND.C2.NE.'y')  GOTO 31
      CALL COMIN(YT)
      IF(YT.EQ.0)GOTO 31
      YT=1/YT
      YB(I,J)=YT
      YB(J,I)=YB(I,J)
   31 CONTINUE
   52 IF(Z1.NE.'A'.AND.Z1.NE.'a')GOTO 53
      WRITE(*,*)'Do you want alter any ADMITTANCE value ? (Y\N)'
      READ(*,11)C3
      IF(C3.NE.'Y'.AND.C3.NE.'y') GOTO 53
      DO 32 I=1,N
      DO 32 J=I+1,N
      WRITE(*,111)I,J
  111 FORMAT(1X,'The value of Y ',I2,'-',I2,' is ')
      WRITE(*,*)YB(I,J)
      WRITE(*,*)'Do you want change this (Y \ N)'
      READ(*,11)C4
      IF(C4.NE.'Y'.AND.C4.NE.'y')  GOTO 32
      CALL COMIN(YT)
      YB(I,J)=YT
      YB(J,I)=YB(I,J)
   32 CONTINUE
   53 IF(X.NE.'Y'.AND.Z1.NE.'y')GOTO 51
      IF(X1.NE.'L'.AND.X1.NE.'l')GOTO 57
      WRITE(*,*)
      WRITE(*,*)'          Do you want alter any'
      WRITE(*,*)' LINE CHARGING ADMITTANCE value?(Y\N)'
      READ(*,11)C5
      IF(C5.NE.'Y'.AND.C5.NE.'y') GOTO 51
      DO 34 I=1,N
      DO 34 J=I+1,N
      WRITE(*,112)I,J
  112 FORMAT(1X,'The value of YSH(line charging) ',I2,'-',I2,' is ')
      YSH(I,J)=2*YSH(I,J)
      WRITE(*,*)YSH(I,J)
      WRITE(*,*)'Do you want change this (Y \ N)'
      READ(*,11)C6
      IF(C6.NE.'Y'.AND.C6.NE.'y')  GOTO 34
      CALL COMIN(YT)
```

```
      YSH(I,J)=YT/2
      YSH(J,I)=YSH(I,J)
   34 CONTINUE
   57 IF(X1.NE.'H'.AND.X1.NE.'h')GOTO 51
      WRITE(*,*)
      WRITE(*,*)'                Do you want to alter any'
      WRITE(*,*)'HALF LINE CHARGING ADMITTANCE value?(Y\N)'
      READ(*,11)C7
      IF(C7.NE.'Y'.AND.C7.NE.'y') GOTO 51
      DO 88 I=1,N
      DO 88 J=I+1,N
      WRITE(*,113)I,J
  113 FORMAT(1X,'The value of YSH(half line charging) '
     2,I2,'-',I2,' is ')
      WRITE(*,*)YSH(I,J)
      WRITE(*,*)'Do you want to change this (Y \ N)'
      READ(*,11)C8
      IF(C8.NE.'Y'.AND.C8.NE.'y')  GOTO 88
      CALL COMIN(YT)
      YSH(I,J)=YT
      YSH(J,I)=YSH(I,J)
   88 CONTINUE
      DO 23 I=1,N
   23 WRITE(*,*)(YB(I,J),J=1,N)
   51 DO 81 I=1,N
      DO 81 J=1,N
      YB1(I,J)=YB(I,J)
      YSH1(I,J)=YSH(I,J)
   81 CONTINUE
      RETURN
      STOP
      END

      SUBROUTINE COMIN(X)
      COMPLEX X
      READ(*,*)A,B
      X=CMPLX(A,B)
      RETURN
      STOP
      END


C     SUBROUTINE TO FORM THE Y BUS
      SUBROUTINE YBUS(N,Y,YSH,YB)
      COMPLEX Y(15,15),YSH(15,15),YB(15.15)
      DO 57 I=1,N
      DO 57 J=1,I-1
      Y(I,J)=Y(J,I)
   57 CONTINUE
      DO 55 I=1,N
      Y(I,I)=Ø
      DO 55 J=1,N
```

```
      IF(I.EQ.J)GOTO 55
      Y(I,I)=Y(I,I)+Y(I,J)+YSH(I,J)
   55 CONTINUE
      DO 56 I=1,N
      DO 56 J=I+1,N
      Y(I,J)=-Y(I,J)
      Y(J,I)=Y(I,J)
   56 CONTINUE
      DO 65 I=1,N
      DO 65 J=1,N
      YB(I,J)=Y(I,J)
   65 CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE TO FORM B BUS
      SUBROUTINE BBUS(N,Y,B)
      DIMENSION B(15,15)
      COMPLEX Y(15,15)
      DO 56 I=1,N
      DO 57 J=1,N
      B(I,J)=AIMAG(Y(I,J))
   57 CONTINUE
   56 CONTINUE
      RETURN
      STOP
      END


C     SUBROUTINE TO RECEIVE THE BUS VALUES
      SUBROUTINE INPUT2(N,P,Q,V,FL)
      DIMENSION BUS(15),P(15),Q(15),V(15)
      CHARACTER FL(15),X,X1,X4
      WRITE(*,*)'BUS1 IS ASSUMED TOBE SLACK BUS'
      WRITE(*,*)'Is the voltage 1 P.U. at slack bus?(Y/N)'
      READ(*,18)X1
   18 FORMAT(A1)
      IF(X1.EQ.'Y'.OR.X1.EQ.'y')THEN
      V(1)=1.Ø
      ELSE
      WRITE(*,*)'Then enter its value'
      READ(*,*)V(1)
      ENDIF
      WRITE(*,*)'Does bus1 has any load'
      READ(*,18)X4
      IF(X4.NE.'Y'.AND.X4.NE.'y')GOTO 24
      WRITE(*,*)'Enter the loads P & Q at slack bus'
      READ(*,*)P(1),Q(1)
      P(1)=-P(1)
      Q(1)=-Q(1)
   24 WRITE(*,*)'ENTER THE VALUES OF OTHER BUSES'
      DO 12 I=2,N,1
      WRITE(*,*)'BUS',I
   19 WRITE(*,Ø1)I
   Ø1 FORMAT(1X,'IS BUS',I2,'A GENERATOR BUS?(Y/N)')
```

```
      READ(*,15)FL(I)
   15 FORMAT(A1)
C      IF(FL(I).NE.'Y'.OR.FL(I).NE.'y'.OR.
C     FL(I).NE.'N'.OR.FL(I).NE.'n')GOTO 18
      IF(FL(I).EQ.'Y')THEN
      WRITE(*,*)'Enter the Generator power & voltage in P.U'
      READ(*,*)P(I),V(I)
      Q(I)=Ø
      WRITE(*,*)'Does it have any load attached to it(Y/N)'
      READ(*,18)X
      IF(X.EQ.'Y'.OR.X.EQ.'y')THEN
      WRITE(*,*)'Then enter P & Q'
      READ(*,*)AA,BB
      P(I)=P(I)-AA
      Q(I)=Q(I)-BB
      ENDIF
      ELSE
      WRITE(*,*)'ENTER THE LOAD VALUES (P),(Q)'
      READ(*,*)P(I),Q(I)
      P(I)=-P(I)
      Q(I)=-Q(I)
      V(I)=1.Ø
      ENDIF
   12 CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE SETTING THE SPECIFIED VALUES
      SUBROUTINE SPECI(N,P,Q,PS,QS)
      DIMENSION P(15),Q(15),PS(15),QS(15)
      DO 1Ø I=2,N
      PS(I)=P(I)
      QS(I)=Q(I)
   1Ø CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE TO FORM BP MATRIX
      SUBROUTINE BPP(N,B,BP)
      DIMENSION B(15,15),BP(15,15)
      DO 1Ø I=1,N-1
      DO 1Ø J=1,N-1
      BP(I,J)=-B(I+1,J+1)
   1Ø CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE TO FORM BQ MATRIX
      SUBROUTINE BQQ(N,FL,B,BQ,K)
```

```
      DIMENSION BQ(15,15),B(15,15)
      CHARACTER FL(15)
      WRITE(*,*)(FL(I),I=2,N,1)
      K=Ø
      DO 1Ø I=2,N
      IF(FL(I).NE.'N')GOTO1Ø
      K=K+1
      L=Ø
      DO 2Ø J=2,N
      IF (FL(J).NE.'N')GOTO 2Ø
      L=L+1
      BQ(K,L)=-B(I,J)
 2Ø   CONTINUE
 1Ø   CONTINUE
      RETURN
      STOP
      END

      SUBROUTINE MUL(N,A,B,C)
      DIMENSION A(15,15),B(15,15),C(15,15)
      DO 1Ø I=1,N
      DO 1Ø J=1,N
      C(I,J)=Ø
      DO 1Ø K=1,N
      C(I,J)=C(I,J)+A(I,K)*B(K,J)
 1Ø   CONTINUE
      RETURN
      STOP
      END

      SUBROUTINE OP1(N,A)
      COMPLEX A(15,15)
      DO 1Ø I=1,N
      WRITE(*,*)(A(I,J),J=1,N)
 1Ø   CONTINUE
      RETURN
      STOP
      END

      SUBROUTINE OP2(N,A)
      DIMENSION A(15,15)
      DO 1Ø I=1,N
      WRITE(*,*)(A(I,J),J=1,N)
 1Ø   CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE TO FIND DELP VALUES
      SUBROUTINE DELPP(N,SP,CAL,V,DELP,DELPMAX,DELPPP)
      DIMENSION SP(15),CAL(15),DELP(15),V(15),DELPPP(15)
      DELPMAX=Ø
      DO 1Ø I=2,N
      DELP(I)=SP(I)-CAL(I)
      IF(ABS(DELP(I)).GT.DELPMAX)DELPMAX=ABS(DELP(I))
 1Ø   CONTINUE
      DO 2Ø I=2,N
```

```
      IF(V(I).EQ.Ø)THEN
      GOTO 2Ø
      ELSE
      DELP(I)=DELP(I)/V(I)
      WRITE(*,*)'DELP',I,DELP(I)
      DELPPP(I-1)=DELP(I)
      ENDIF
   2Ø CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE TO FIND DELQ VALUES
      SUBROUTINE DELQQ(N,SP,CAL,V,FL,DELQ,DELQMAX,DELQQQ)
      DIMENSION SP(15),CAL(15),DELQ(15),V(15),DELQQQ(15)
      CHARACTER FL(15)
      DELQMAX=Ø
      DO 1Ø I=2,N
      DELQ(I)=SP(I)-CAL(I)
      IF(ABS(DELQ(I)).GT.DELQMAX)DELQMAX=ABS(DELQ(I))
   1Ø CONTINUE
      L=Ø
      DO 2Ø I=2,N
      IF(V(I).EQ.Ø)THEN
      GOTO 2Ø
      ELSE
      DELQ(I)=DELQ(I)/V(I)
      WRITE(*,*)'DELQ',I,DELQ(I)
      IF(FL(I).EQ.'N'.OR.FL(I).EQ.'n')THEN
      L=L+1
      DELQQQ(L)=DELQ(I)
      ENDIF
      ENDIF
   2Ø CONTINUE
      RETURN
      STOP
      END

      SUBROUTINE MUL1(N,A,B,C)
      DIMENSION A(15,15),B(15),C(15)
      CALL OP2(N,A)
      WRITE(*,*)(B(I),I=1,N)
      DO 2Ø J=1,N
      C(J)=Ø
      DO 1Ø K=1,N
      C(J)=C(J)+A(J,K)*B(K)
   1Ø CONTINUE
   2Ø CONTINUE
      RETURN
      STOP
      END
```

```
C       SUBROUTINE TO FORM THE INV. OF A MATRIX
        SUBROUTINE INV(NX,AA,CC)
        DIMENSION AA(15,15),CC(15,15),BB(15,15),MC1(15),MC2(15),DT(15)
        DO 5I=1,NX
        MC1(I)=Ø
        MC2(I)=Ø
    5   CONTINUE
        MU=NX+NX
        MG=NX+1
        DO 7Ø I=1,NX
        DO 6Ø J=1,NX
        BB(I,J)=AA(I,J)
   6Ø   CONTINUE
   7Ø   CONTINUE
        DO 9Ø I=1,NX
        DO 8Ø J=MG,MU
        BB(I,J)=Ø.Ø
   8Ø   CONTINUE
   9Ø   CONTINUE
        DO 1ØØ I=1,NX
        MH=I+NX
        BB(I,MH)=1.Ø
  1ØØ   CONTINUE
        IC=1
  11Ø   AL=BB(IC,IC)
        LR=IC
        LC=IC
        DO 13Ø I=IC,NX
        DO 12Ø J=IC,NX
        IF(ABS(BB(I,J)).LE.AL)GOTO 12Ø
        AL=BB(I,J)
        LR=I
        LC=J
  12Ø   CONTINUE
  13Ø   CONTINUE
        IF(AL.EQ.Ø.Ø)GOTO 27Ø
        IF(LR.EQ.IC.AND.LC.EQ.IC)GOTO 16Ø
        DO 14Ø I=1,MU
        DT(I)=BB(IC,I)
        BB(IC,I)=BB(LR,I)
        BB(LR,I)=DT(I)
  14Ø   CONTI NUE
        DO 15Ø I=1,NX
        DT(I)=BB(I,IC)
        BB(I,IC)=BB(I,LC)
        BB(I,LC)=DT(I)
  15Ø   CONTINUE
        MC1(IC)=IC
        MC2(IC)=LC
  16Ø   IF(BB(IC,IC).EQ.Ø.Ø)GOTO 27Ø
        PU=BB(IC,IC)
```

```
      IF(PU.EQ.1.Ø)GOTO 175
      DO 17Ø I=1,MU
      BB(IC,I)=BB(IC,I)/PU
  17Ø CONTINUE
  175 DO 19Ø I=1,NX
      IF(I.EQ.IC)GOTO 19Ø
      PU1=BB(I,IC)
      DO 18Ø J=IC,MU
      BB(I,J)=BB(I,J)-BB(IC,J)*PU1
  18Ø CONTINUE
  19Ø CONTINUE
      IF(IC.EQ.NX)GOTO 2ØØ
      IC=IC+1
      GOTO 11Ø
  2ØØ DO 22Ø I=1,NX
      DO 21Ø J=1,NX
      JS1=J+NX
      CC(I,J)=BB(I,JS1)
  21Ø CONTINUE
  22Ø CONTINUE
      DO 24Ø IQ=NX,1,-1
      KS1=MC1(IQ)
      KS2=MC2(IQ)
      IF(KS1.EQ.Ø)GOTO 24Ø
      DO 23Ø J=1,NX
      DT(J)=CC(KS1,J)
      CC(KS1,J)=CC(KS2,J)
      CC(KS2,J)=DT(J)
  23Ø CONTINUE
  24Ø CONTINUE
      GOTO 29Ø
  27Ø WRITE(*,28Ø)
  28Ø FORMAT(5X,'INV. IS NOT EXISTING')
  29Ø CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE TO O/P ITRATED VALUES
      SUBROUTINE OPITR(N,DELTA,V,FL)
      DIMENSION DELTA(15),V(15)
      CHARACTER FL(15)
      INTEGER COU1
      DO 6Ø I=1,N
      WRITE(*,65)I,DELTA(I)
   65 FORMAT(1X,'DELTA',I2,' IS',F6.4)
   6Ø CONTINUE
      DO 8Ø COU1=1,N
      IF(FL(COU1).EQ.'N')THEN
      WRITE(*,75)COU1,V(COU1)
   75 FORMAT(1X,'VOLTAGE AT BUS',I2,1X,'IS',F6.4)
```

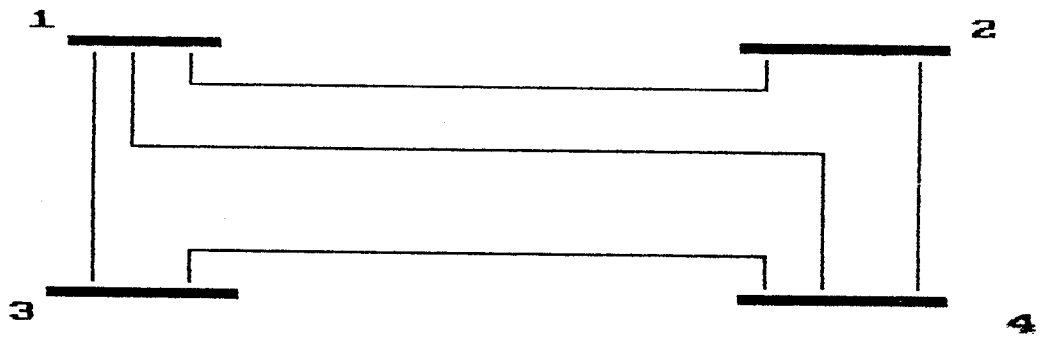
```
      ENDIF
   80 CONTINUE
      RETURN
      STOP
      END


C     THIS SUBROUTINE IS TO FIND MODULUS & ANGLE
      SUBROUTINE MODANG(A,EMOD,ANG)
      COMPLEX A
      B=REAL(A)
      C=AIMAG(A)
      EMOD=SQRT(B**2+C**2)
      IF(B.EQ.0.AND.C.GT.0)THEN
      ANG=1.570796327
      GOTO 40
      ENDIF
      IF(B.EQ.0.AND.C.LT.0)THEN
      ANG=-1.570796327
      GOTO 40
      ENDIF
      IF(B.EQ.0.AND.C.EQ.0)THEN
      ANG=0
      GOTO 40
      ENDIF
      ANG=ATAN(C/B)
      IF(REAL(A).LT.0)THEN
      ANG=3.141592654-ABS(ANG)
      IF(AIMAG(A).LT.0)THEN
      ANG=ANG*(-1)
      ENDIF
      ENDIF
      IF(AIMAG(A).LT.0)THEN
      ANG=(-1)*ABS(ANG)
      ENDIF
      ENDIF
   40 RETURN
      STOP
      END

C     SUBROUTINE TO FIND CALCULATED VALUES - P
      SUBROUTINE CALCP(N,V,Y,DELTA,PC)
      COMPLEX Y(15,15)
      DIMENSION V(15),DELTA(15),PC(15),THETA(15,15)
      REAL MODY(15,15),MODV(15),ANV(15),M
      DO 20 I=2,N
      RRM=0
      DO 10 K=1,N
      CALL MODANG(Y(I,K),MODY(I,K),THETA(I,K))
      AANG=THETA(I,K)+DELTA(K)-DELTA(I)
      RRM=RRM+((ABS(V(K)))*MODY(I,K)*COS(AANG))
   10 CONTINUE
      PC(I)=ABS(V(I))*RRM
      WRITE(*,*)'THE VALUE OF "P" AT THE BUS',I,'=',PC(I)
   20 CONTINUE
      RETURN
```

```
      STOP
      END

C     SUBROUTINE TO FIND CALCULATED VALUES - Q
      SUBROUTINE CALCQ(N,V,Y,DELTA,QC)
      COMPLEX Y(15,15)
      DIMENSION V(15),DELTA(15),QC(15),THETA(15,15)
      REAL MODY(15,15)
      REAL MODV(15)
      DO 20 I=2,N
      RRM=0
      DO 10 K=1,N
      CALL MODANG(Y(I,K),MODY(I,K),THETA(I,K))
      AANG=THETA(I,K)-DELTA(I)+DELTA(K)
      RRM=RRM+(ABS(V(K))*MODY(I,K)*SIN(AANG))
   10 CONTINUE
      QC(I)=-ABS(V(I))*RRM
      WRITE(*,*)'QC',I,QC(I)
   20 CONTINUE
      RETURN
      STOP
      END

      SUBROUTINE LOADFL(N,PC,QC,PF,QF)
      DIMENSION PC(15),QC(15),PF(15,15),QF(15,15)
      DO 120 I=1,N
      DO 120 J=I+1,N
      PF(I,J)=PC(I)-PC(J)
      QF(I,J)=QC(I)-QC(J)
  120 CONTINUE
      RETURN
      STOP
      END

C     SUBROUTINE TO CONVERT POLAR TO RECTANGULAR.
      SUBROUTINE PORE(N,POL,ANG,RECT)
      COMPLEX RECT(15)
      REAL POL(15),ANG(15)
      DO 10 I=1,N
      A=POL(I)*COS(ANG(I))
      B=POL(I)*SIN(ANG(I))
      RECT(I)=CMPLX(A,B)
   10 CONTINUE
      RETURN
      STOP
      END

      SUBROUTINE LINVAL(N,VRECT,Y,YSH,XIRECT,PFLOW,QFLOW,LOSS)
      COMPLEX VRECT(15),Y(15,15),YSH(15,15)
      COMPLEX XIRECT(15,15),LOSS(15,15),PQFL(15,15)
      REAL PFLOW(15,15),QFLOW(15,15)
      S=0.0
      T=0.0
      DO 10 I = 1,N
      DO 10 J = 1,N
      IF(I.EQ.J) THEN
      XIRECT(I,J)=CMPLX(S,T)
      ELSE
      XIRECT(I,J)=(VRECT(I)-VRECT(J))*Y(I,J)+VRECT(I)*(YSH(I,J)/2)
```

```
      WRITE(*,*)'CURRENT',I,J,XIRECT(I,J)
      PQFL(I,J)=CONJG(VRECT(I))*XIRECT(I,J)
      PFLOW(I,J)=REAL(PQFL(I,J))
      QFLOW(I,J)=(-1)*AIMAG(PQFL(I,J))
      ENDIF
10    CONTINUE
      DO 20 I=1,N
      DO 20 J=I+1,N
      LOSS(I,J)=PQFL(I,J)+PQFL(J,I)
      LOSS(J,I)=LOSS(I,J)
      WRITE(*,*)'LOSS',I,J,LOSS(I,J)
20    CONTINUE
      RETURN
      STOP
      END
```

CHAPTER IV

4.1. SYSTEM DATA



- One Line Diagram

LINE DATA :

BUS	IMPEDANCE (p.u)	SHUNT ADMITTANCE (p.u)
1 - 2	0.02066 + j0.1446	0+j7.2315*10^-7
1 - 3	0.022727 + j0.15909	0+j7.9545*10^-7
1 - 4	0.03099 + j0.216942	0+j1.0847*10^-6
2 - 4	0.02066 + j0.1446	0+j7.2315*10^-7
3 - 4	0.0247 + j0.1735	0+j8.677*10^-7

BUS DATA :

BUS	BUS POWER (p.u)		VOLTAGE MAGNITUDE (p.u)	BUS TYPE
	Real	Reactive		
1	unspecified	unspecified	1.02	slack
2	0.95	unspecified	1.01	PV
3	-2.0	-1.0	unspecified	PQ
4	-1.0	-0.2	unspecified	PQ

DATA FED TO THE PROGRAM

MODIN

```
Is the input an Admittance or an Impedance

Please select your choice  ( A\I )
I
Give the number of buses
4
Enter the Impedances in P.U.
Enter the value of Z  1- 2
Ø.Ø2Ø66 Ø.1446
Enter the value of Z  1- 3
Ø.Ø22727 Ø.15ØØ9
Enter the value of Z  1- 4
Ø.Ø3Ø99 Ø.216942
Enter the value of Z  2- 3
Ø Ø
Enter the value of Z  2- 4
Ø.Ø2Ø66 Ø.1446
Enter the value of Z  3- 4
Ø.Ø247 Ø.1735
Do you have half line charging
 Admittances (Y\N)
Y
ENTER YOUR CHOICE
  H -  HALF LINE CHARGING ADMITTANCE
  L -  LINE CHARGING ADMITTANCE
L
ENTER THE LINE  CHARGING
ADMITTANCE OF BUS  1 2
Ø Ø.ØØØØØØ72314
ENTER THE LINE  CHARGING
ADMITTANCE OF BUS  1 3
Ø Ø.ØØØØØØ79545
ENTER THE LINE  CHARGING
ADMITTANCE OF BUS  1 4
Ø Ø.ØØØØØ1Ø847
ENTER THE LINE  CHARGING
ADMITTANCE OF BUS  2 3
Ø Ø
ENTER THE LINE  CHARGING
ADMITTANCE OF BUS  2 4
Ø Ø.ØØØØØØ72314
ENTER THE LINE  CHARGING
ADMITTANCE OF BUS  3 4
Ø Ø.ØØØØØØ8677
Enter the value of Ephsalon "E"
Ø.Ø1
```

```
       DO YOU WANT TO ALTER ANY LINE PARAMETER
        VALUES  CHOOSE  ( A\ N)
 N
 BUS1 IS ASSUMED TOBE SLACK BUS
 Is the voltage 1 P.U. at slack bus?(Y/N)
 N
 Then enter its value
 1.Ø2
 Does bus1 has any load
 N
 ENTER THE VALUES OF OTHER BUSES
 BUS               2
 IS BUS 2A GENERATOR BUS?(Y/N)
 Y
 Enter the Generator power & voltage in P.U
 Ø.95 1.Ø1
 Does it have any load attached to it(Y/N)
 N
 BUS               3
 IS BUS 3A GENERATOR BUS?(Y/N)
 N
 ENTER THE LOAD VALUES (P),(Q)
 2 1
 BUS               4
 IS BUS 4A GENERATOR BUS?(Y/N)
 N
 ENTER THE LOAD VALUES (P),(Q)
 1 Ø.2

 ------------------------------------------------------------------
                      MATRIX  OUTPUTS
 ------------------------------------------------------------------
 THE Y BUS IS
  (2.4936160,-17.4546600) (-9.683166E-Ø1,6.7772790) (-8.799998E-Ø1,6.160037
  (-6.452996E-Ø1,4.5173460)
  (-9.683166E-Ø1,6.7772790) (1.9366330,-13.5545600) (.0000000,.0000000)
  (-9.683166E-Ø1,6.7772790)
  (-8.799998E-Ø1,6.1600370) (.0000000,.0000000) (1.6842370,-11.8092300)
  (-8.Ø42370E-Ø1,5.6491950)
  (-6.452996E-Ø1,4.5173460) (-9.683166E-Ø1,6.7772790)
  (-8.Ø42370E-Ø1,5.6491950) (2.4178530,-16.9438200)
 THE B BUS IS
        -17.4546600          6.7772790          6.1600370          4.5173460
          6.7772790        -13.5545600           .0000000          6.7772790
          6.1600370          .0000000        -11.8092300          5.6491950
          4.5173460          6.7772790          5.6491950        -16.9438200
 THE BP BUS IS
         13.5545600           .0000000         -6.7772790
           .0000000         11.8092300         -5.6491950
         -6.7772790         -5.6491950         16.9438200
 YNN
 THE BQ BUS IS
         11.8092300         -5.6491950
         -5.6491950         16.9438200
 Pause.
 Please press <return> to continue.
```

```
  Bus Code 1 -- 2
  ---------------
Current= (-1.482258E-002,6.704108E-002)
Real Power flow = -1.511903E-002
Reactive power flow = -6.838190E-002

  Bus Code 1 -- 3
  ---------------
Current= (-1.4852830,1.2028130)
Real Power flow =          -1.5149890
Reactive power flow =          -1.2268690

  Bus Code 1 -- 4
  ---------------
Current= (-6.464336E-001,4.096638E-001)
Real Power flow = -6.593623E-001
Reactive power flow = -4.178571E-001

  Bus Code 2 -- 1
  ---------------
Current= (1.482258E-002,-6.704070E-002)
Real Power flow =  1.502164E-002
Reactive power flow =  6.769985E-002

  Bus Code 2 -- 3
  ---------------
Current= (.0000000,.0000000)
Real Power flow =             .0000000
Reactive power flow =             .0000000

  Bus Code 2 -- 4
  ---------------
Current= (-9.550285E-001,5.475449E-001)
Real Power flow = -9.649936E-001
Reactive power flow = -5.522960E-001

  Bus Code 3 -- 1
  ---------------
Current= (1.4852830,-1.2028130)
Real Power flow =          1.4319710
Reactive power flow =  6.457410E-001

  Bus Code 3 -- 2
  ---------------
Current= (.0000000,.0000000)
Real Power flow =             .0000000
Reactive power flow =             .0000000

  Bus Code 3 -- 4
  ---------------
Current= (5.533815E-001,-5.909898E-001)
Real Power flow =  5.633683E-001
Reactive power flow =  3.541378E-001
```

```
  Bus Code 4 -- 1
  ---------------
Current= (6.464337E-Ø1,-4.Ø96633E-Ø1)
Real Power flow =  6.412114E-Ø1
Reactive power flow =  2.9Ø7935E-Ø1

  Bus Code 4 -- 2
  ---------------
Current= (9.55Ø285E-Ø1,-5.475445E-Ø1)
Real Power flow =  9.398560E-Ø1
Reactive power flow =  3.77Ø571E-Ø1

  Bus Code 4 -- 3
  ---------------
Current= (-5.533814E-Ø1,5.9Ø9Ø1E-Ø1)
Real Power flow = -5.795591E-Ø1
Reactive power flow = -4.678674E-Ø1
```

```
Stop - Program terminated.

F:\IVEEE\89EEE28\PRO>LOGOUT
89EEE28 logged out from server KCT connection 1Ø.
Login time:   Thursday  March  26, 1992  11:49 am
                                5, 1992  12:16 am
```

CHAPTER V

CONCLUSION

The fast decoupled method offers a uniquely attractive combination of advantages over the established methods, including Newton's, interms of speed, reliability, simplicity and storage, for conventional load flow solutions. The basic algorithm remains unchanged for variety of different applications. Given a set of good ordered elimination routines, the basic program is easy to code efficiently, and its speed and storage requirements are roughly proportional to system size. A useful feature of the method is the ability to reduce its core-storage requirements to approximately those of the Gauss-Seidel method with a small number of core-disk block transfers. The method performs well with conventional adjustment algorithms and solves network outage security-check cases usually in one or two iterations. It is computationally suitable for optimal load-flow calculations and scope for developments in this area are very much. The software developed proves to be flexible and gives feasible results for small networks. As described earlier in this report the

REFERENCES

1. I.J.NAGRATH,D.P.KOTHARI,"MODERN POWER SYSTEM ANALYSIS", TATA McGRAW-HILL PUBLISHING COMPANY Ltd., NEW DELHI,1989.

2. M.A.PAI, "COMPUTER TECHNIQUES IN POWER SYSTEM ANALYSIS", TATA McGRAW-HILL PUBLISHING COMPANY Ltd., NEW DELHI,1979.

3.STOTT,B., and O.ALSAC, "FAST DECOUPLED LOAD FLOW", IEEE Trans., 1974, PAS-93:859.

4.GEORGE L.KUSIC, "COMPUTER AIDED POWER SYSTEM ANALYSIS", PRENTICE-HALL OF INDIA PRIVATE LIMITED, NEW DELHI, 1989.

5.GROSS,C.A.,"POWER SYSTEM ANALYSIS",JOHN WILEY, NEW YORK,1986.

6.WEEDY,B.M.,"ELECTRICAL POWER SYSTEMS",JOHN WILEY, NEW YORK, 1979.