

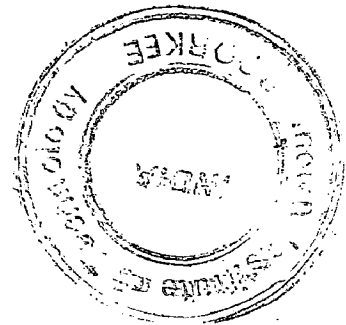
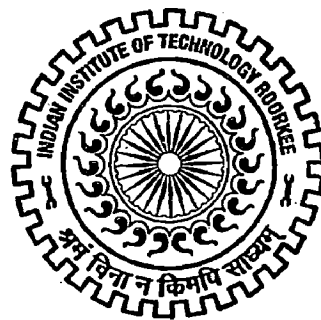
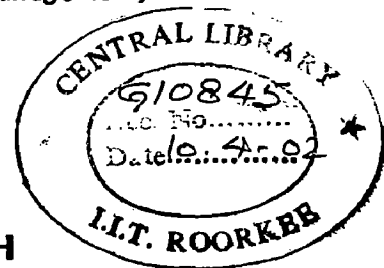
# LOAD FLOW ANALYSIS OF WEAKLY MESHED DISTRIBUTION SYSTEM

## A DISSERTATION

submitted in partial fulfilment of the  
requirements for the award of the degree  
of  
MASTER OF TECHNOLOGY  
in  
WATER RESOURCES DEVELOPMENT  
(Hydro-Electric System Engineering & Management)

By

ARUN KUMAR SINGH



WATER RESOURCES DEVELOPMENT TRAINING CENTRE  
INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE  
ROORKEE - 247 667 (INDIA)

FEBRUARY, 2002

12

## CANDIDATE'S DECLARATION

I do hereby certify that the work which is being presented in the dissertation entitled "**LOAD FLOW ANALYSIS OF WEAKLY MESHED DISTRIBUTION SYSTEM**" in partial fulfillment of the requirements for the award of Degree of Master of Technology in *Hydroelectric System Engineering and Management* submitted in the **Water Resources Development Training Centre, Indian Institute of Technology, Roorkee** is an authentic record of my own work carried out since July, 2001 to February, 2002, under the active guidance and supervision of **Dr. Biswarup Das, Assistant Professor, Deptt. of Electrical Engineering** and **Prof. Devadutta Das, Professor & Head, Water Resources Development Training Centre, Indian Institute of Technology Roorkee, India.**

I have not submitted the matter embodied in this dissertation for award of any other degree or diploma


Place: -Roorkee


Dated: - February 11 , 2002.

  
(ARUN KUMAR SINGH)

---

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

  
(Dr. Biswarup Das)  
Assistant Professor  
Deptt. of Electrical Engg.  
IIT, Roorkee, 247667

  
(Devadutta Das)  
Professor & Head  
W.R.D.T.C  
IIT, Roorkee, 247667

## ACKNOWLEDGEMENT

My foremost & profound gratitude goes to my guides **Dr. Biswarup Das**, *Assistant Professor, Deptt. of Electrical Engineering* and **Shri Devadutta Das**, *Professor and Head, WRDTC, Indian Institute of Technology, Roorkee*, for their proficient & enthusiastic guidance, useful encouragement & immense help. I have deep sense of admiration for their innate goodness & inexhaustible enthusiasm. The valuable hours of discussions and suggestions that I had with them have undoubtedly helped in supplementing my thoughts in the right direction for attaining the desired objective of completing the work in its present form. Working under their guidance will always remain a cherished experience in my memory and I will adore it throughout my life.

My heartfelt gratitude & indebtedness goes to all the faculties of **WRTDC** group who with their encouraging and caring words, constructive criticism & suggestions or by simply doing their duty with sincerity & smile have contributed directly or indirectly in a significant way towards completion of this report.

I am highly grateful to **Dr. (Smt.) Sunita Devi**, in-charge Computer laboratory, **WRDTC, IIT, Roorkee**, for providing the required facilities during the course of study.

I wish to express my gratitude and obligation to the **Authority concern, Deptt. of Power, Govt. of Arunachal Pradesh** for enabling me to take up this course.

I would like to express my indebtedness and gratefulness and want to send my special thanks to my dear friends **Sri R.N. Jha** and **Sri N.R. Sen** for their invaluable service rendered to me.

Special & sincere thanks goes to my batch mates whose support & encouragement has been a constant source of inspiration, guidance and strength.

Over and above, I would like to express my greatest appreciation to my lovely and gracious wife **Mrs. Shalini**, who stands by my side night and day and my children **Master Karan** and **Miss Ankita** for their forbearance during this work.

Place:- Roorkee  
Dated: February {}, 2002

  
(ARUN KUMAR SINGH)

## ABSTRACT

In this thesis, an attempt has been made to develop an algorithm for load flow solution of a weakly meshed (i.e. which contains a few loops) distribution system. In this algorithm, initially the original meshed network is converted into an equivalent radial network by breaking it at appropriate places. The effect of the mesh configuration is taken into account by suitable amount of complex power injections at appropriate buses in the radial network. Finally, the radial network is solved through standard forward/backward sweep algorithm to obtain the load flow solution of the original meshed network. The developed algorithm has been tested on two sample weakly meshed distribution systems.

## CONTENTS

	Page No.
Candidate's Declaration	i
Acknowledgement	ii
Abstract	iii
<b>Chapter 1. Introduction</b>	<b>1-2</b>
<b>Chapter 2. Solution Algorithm of a Meshed Distribution System</b>	<b>3-13</b>
2.1 Basic Concept	3-5
2.2 Branch Numbering Scheme	5-7
2.3 Bus Ordering Scheme	7-8
2.4 Load Flow Equations	8-10
2.5 Power Injections at the LBPs	10-12
2.6 Solution Algorithm	12-13
<b>Chapter 3. Results and Discussion</b>	<b>14-20</b>
<b>Chapter 4. Conclusion</b>	<b>21</b>
<b>References</b>	<b>22-23</b>
<b>Appendix-A</b>	<b>24-25</b>
<b>Appendix-B</b>	<b>26-43</b>

### INTRODUCTION

One of the most fundamental tools for analyzing a power system is load-flow. Essentially, a load-flow solution provides the steady-state operating point of a power system for given a network and load data in the system. The knowledge of the steady-state operating point is necessary for ensuring the secured operation of a power system. If any of the electrical quantities in the system (such as bus voltage, current and power flow over the lines etc.) exceeds its respective limits, certain corrective actions must be taken to bring the out-of-bound quantities within their limits for ensuring secured operation of the system. These corrective actions include switching of shunt capacitors and reactor banks, adjustment of transformer tap settings etc. The nature and magnitude of such corrective actions depend on the magnitudes of violation of the out-of-bound quantities. As the magnitudes of violation are determined from the load-flow solution, the nature and magnitudes of the necessary corrective actions are actually heavily dependent on load-flow solution.

Therefore, for determining accurately the nature and magnitudes of the necessary corrective actions, the load-flow solutions need to be quite accurate. Moreover, as the necessary corrective actions need to be implemented quickly for minimizing the damage to the power system from insecure operation, the load-flow solution needs also to be obtained very quickly. Previously, as the high voltage transmission grid used to carry much more electrical power than a low voltage distribution system, a lot of attention had been given to ensure secured operation of high voltage transmission grid. Towards this end, a number of efficient load-flow algorithms have been developed for the solution of a high voltage transmission grid. Among them prominent are Newton-Raphson Technique [1], Fast Decoupled Load Flow Technique [2] and different improved versions of Fast Decoupled Load Flow Technique [3-5].

However, presently due to increasing load demand, distribution system are also carrying significant amount of electrical power. Consequently, the need for ensuring secured operation of a distribution system has also been felt and towards this goal, the

need for efficient load-flow techniques for distribution system analysis has also been identified. Now, power distribution networks, with their primarily radial configuration and wide-ranging resistance and reactance values of the feeders, are inherently ill-conditioned and hence, the load-flow techniques developed for transmission system analysis are not suitable for the analysis of distribution system. As a result different load-flow algorithms have been reported in the literatures for the analysis of radial distribution system [6-8].

Although the above algorithm [6-8] are quite efficient for solving a radial distribution system, they are not suitable for solving weakly meshed distribution networks (i.e. networks containing loops), which are not uncommon in distribution systems. Consequently, different special algorithm has been developed for solving a meshed distribution system. Goswami, S.K. and Basu, S. K. [9], presented a method based on loop impedance matrix. In [10], a compensation based power-flow method has been proposed for solving both radial and meshed distribution systems. An efficient load-flow method for large weakly meshed distribution system has been presented in [11]. Haque presented [12] a very simple method for solving a meshed distribution network.

In this thesis, a technique for solving a weakly meshed distribution system has been developed. The work presented in this thesis is essentially in the same line as that of [12]. In this method, the meshed network is first converted into an equivalent radial configuration by breaking the loops. In this process, some new dummy buses are added in the system. Next, the power injections at the loop break points (the nodes where the loops in the original system have been broken) are computed by using a reduced order bus impedance matrix. After the calculation of injection powers at the loop break points, the equivalent radial system is solved to obtain the final solution.

This thesis is organized as follows. Chapter-2 discusses the load-flow solution algorithm for analyzing a meshed distribution system in detail. Chapter-3 presents the main results of this work. In this work, the developed technique has been applied to two different distribution systems. Finally, Chapter-4 gives the main conclusions of this work.

---

## SOLUTION ALGORITHM OF A MESHED DISTRIBUTION SYSTEM

In this chapter, algorithm for power flow solution of a meshed distribution system is described in detail.

### 2.1 Basic Concept

Generally a distribution system draws power from a point (substation) and the configuration of the system is normally radial. In a radial distribution system, the number of lines and the number of buses are related as follows,

$$n = n_b + 1 \quad \dots (2.1)$$

Where  $n$  = number of buses and

$n_b$  = number of branches.

However, to increase the efficiency and reliability of the system, a distribution system is sometimes operated in a weakly meshed (i.e. containing a few loops) configuration. In case of a meshed network, number of buses in the system may be less than or equal to the number of branches in the system. The number of loops  $n_{LP}$  of a meshed network is given by,

$$n_{LP} = n_b - n + 1 \quad \dots (2.2)$$

As the power flow solution algorithm of a radial distribution system is well established in this work the power flow solution algorithm of a meshed distribution system is developed on the basis of the power flow solution algorithm of a radial distribution system. In this approach, a meshed distribution system is first converted into a radial system. The effect of the loop configuration is taken into account by calculating complex power injections at appropriate nodes of the radial system. Finally, the radial system obtained is solved to obtain the final solution of the original meshed distribution system.



A meshed network having loops can be converted into an equivalent radial network by opening the loops. When a loop is opened, an extra dummy bus is created. Injecting appropriate amount of complex power at the loop breakpoints (LBPs) preserves the characteristics of the original meshed network. Thus, the number of dummy buses is equal to the number of loop  $n_{LP}$  existing in the original meshed distribution system. An illustration of the above procedure is shown in Fig. 2.1.

Fig. 2.1 shows an example of a meshed network in which the branch between **a** and **b** makes a loop. The equivalent radial network of Fig. 2.1 has been shown in Fig. 2.2 in which the loop of the network is opened by adding a dummy bus **a'**. Thus, a LBP is created at bus **a**.

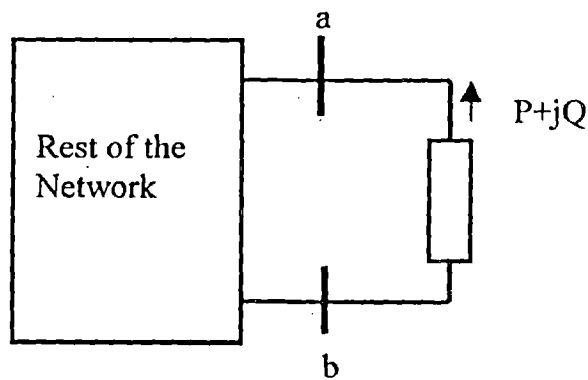


Figure 2.1: A Mesh Network

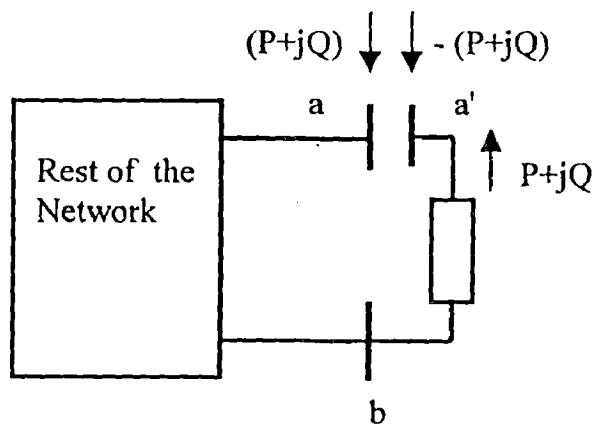


Figure 2.2: Conversion of Mesh to radial Network by adding Dummy bus **a'**

To preserve the characteristic of the network after breaking, a compensatory power is injected to both sides of a break point as shown in Fig 2.2. This also reflects the power circulation in the original loop. The injected power at the LBPs should be equal in magnitude with opposite in sign as shown in Fig. 2.2.

Although the above procedure is quite simple to apply, a systematic procedure must be adopted to identify the nodes at which the loops are to be broken. This is achieved by numbering the branches in a systematic fashion. The procedure for numbering the branches is described in the following section.

## 2.2 Branch Numbering Scheme

For branch numbering procedure, first the tree of the given network must be constructed. The tree consists of several layers and starts at the root node, where the power source is connected. The root bus is also the slack bus of the network. The first layer consists of all the branches that are connected to the root bus. The next (second) layer consists of all the branches that are connected to receiving end bus of the branches in the previous (first) layer and so on. In this fashion all the branches of the network should be considered in the tree and they should appear only once. In this process, if it is observed that the receiving end bus of a newly added branch has already been considered in the tree then the newly added branch makes a loop in the network. The loop can be opened by introducing a dummy bus, which is numbered with a prime sign ('). The newly created dummy bus is assigned the role of receiving end bus of the newly considered branch instead of its original receiving end bus.

A single line diagram of a 9-bus system [12] is shown in Fig. 2.3. In this system, 'o' is root bus or slack bus. The tree diagram of the network is shown in Fig. 2.4. The first layer of the tree consists of the branch o-a. The second layer consists of the branches a-b, a-c, a-f and a-h. The third layer consists of branches that are connected to buses b, c, f, or h but are not yet considered. This layer starts with branch b-c. Since the receiving end bus (bus c) of this branch has already been considered in the tree, a loop is created at this bus. This is opened by adding a dummy bus c'. This dummy bus c' is now assigned the role of receiving end bus of the branch b-c. Thus, the branch between buses b and c is now put between buses b and c'. The corresponding LBP (loop break point, i.e. where the

loop is broken) is  $c-c'$ . Similarly the other LBPs in this layer are  $h-h'$  and  $e-e'$  and the other branches in this layer are  $c-d$ ,  $c-e$ ,  $f-h'$ , and  $f-g$ . By this same process, in the last (fourth) layer another LBP  $g-g'$  is created. Thus, the last layer consists of only one branch  $d-g'$ . Now, the total branches in the system that are numbered are  $o-a$ ,  $a-b$ ,  $a-c$ ,  $a-f$ ,  $a-h$ ,  $b-c'$ ,  $c-d$ ,  $c-e$ ,  $f-h'$ ,  $f-e'$ ,  $f-g$  and  $d-g'$ .

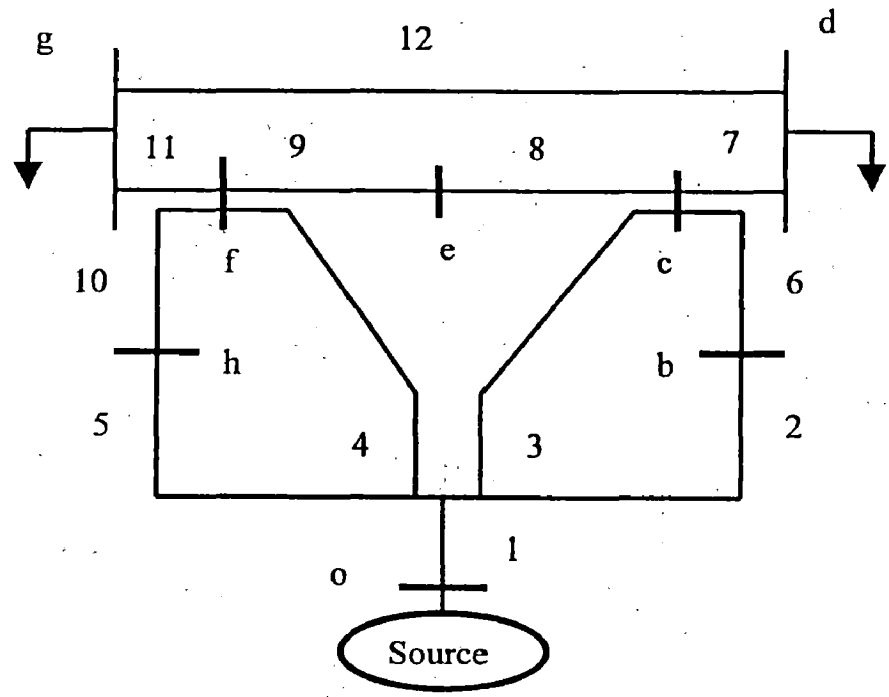


Figure 2.3: Single line diagram of a 9-bus network

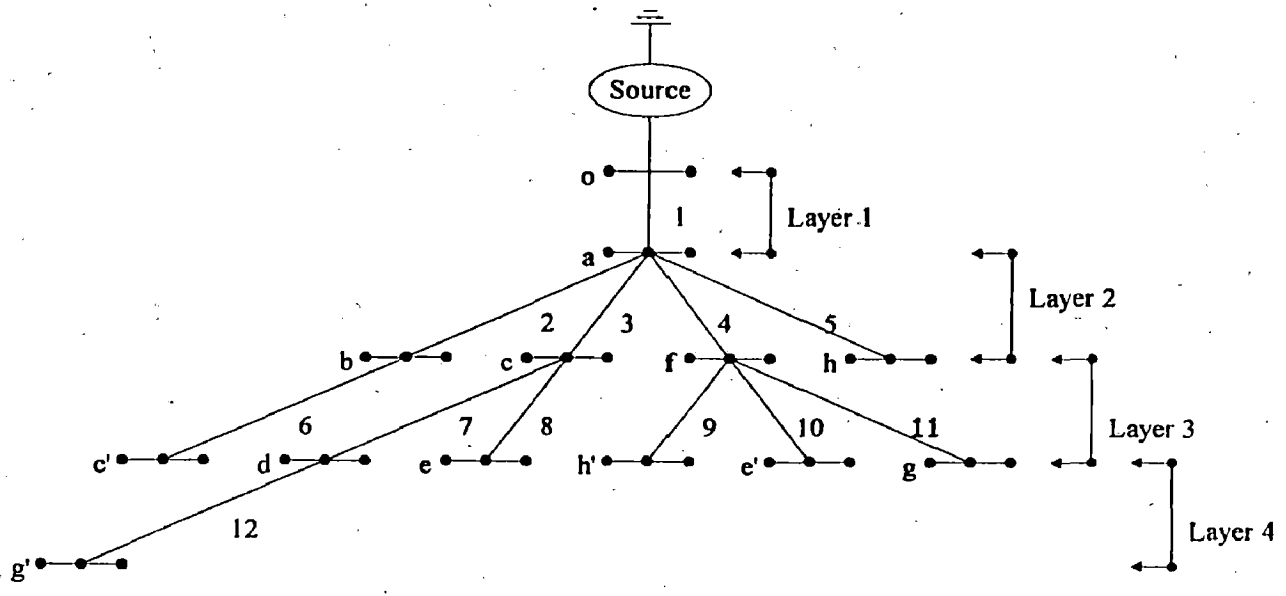


Figure 2.4: Tree diagram of the 9-bus network

Now according to the equation (2.2) there are four loops in the system. The total number of LBPs identified during the tree construction process is also 4 (four) and they are **c-c'**, **h-h'**, **e-e'**, and **g-g'**. After the loops are broken and the original mesh network is converted into radial network, for efficient load flow solution, all the buses (the original buses plus the dummy buses) are ordered in a particular fashion in this work. In the next section, the bus-ordering scheme is described in detail.

### 2.3 Bus Ordering Scheme

Due to the addition of dummy buses, the number of buses in a tree is more than the actual number of buses that existed before breaking the loops. Now, the buses so obtained are divided into three different sets, namely **set-a**, **set-b** and **set-c**.

**Set-a:** In this set, the original buses where LBPs are created during the tree construction process are kept. As it contains the original buses, the number of buses is equal to the number of loops ( $n_{LBP}$ ) in the original system. As in Fig. 2.4, buses **c**, **h**, **e** and **g** belong to this set. The order or sequence of the buses in this set is not important.

**Set-b:** This set consists of all the dummy buses created in the process of mesh to radial conversion of the system by breaking the loops. Therefore, with reference to Fig. 2.4 buses **c'**, **h'**, **e'** and **g'** belong to this set. The order or sequence of the buses in this set should be same as used in **set-a**.

**Set-c:** This set contains all the buses except the root bus that are not considered in **set-a** and **set-b**. In fact, it consists of all the remaining buses except the root bus. Again, the order or sequence of the buses in this set is not important.

It is assumed that the network is so meshed that there is only one loop at any bus. The bus ordering process is essential in order to compute the power injections at the LBPs, to take into account the effect of the meshed network.

Once the original meshed network is converted into an equivalent radial network and the buses are appropriately ordered, the radial network is solved by an iterative backward/forward sweep algorithm. However, to proceed with the backward/forward sweep algorithm, the complex power injections at the LBPs need to be first calculated.

In the next section, the basic backward/forward sweep algorithm and in the subsequent section the procedure for calculating complex power injections at the LBPs are presented.

## 2.4 Load Flow Equations

As it is considered that the network is fed from a single source, the load flow problem can be solved iteratively using two sets of recursive equations. These are known as backward and forward recursive equations. The backward recursive equations are applied to compute the power flows in the branches and forward recursive equations are applied for computing the bus voltages. The procedure is illustrated as follows.

In Fig. 2.5, a branch  $i$  between buses  $k$  and  $m$  is shown. The branch  $i$  is modeled as  $\pi$  (PI) equivalent network as shown in the Fig.2.5

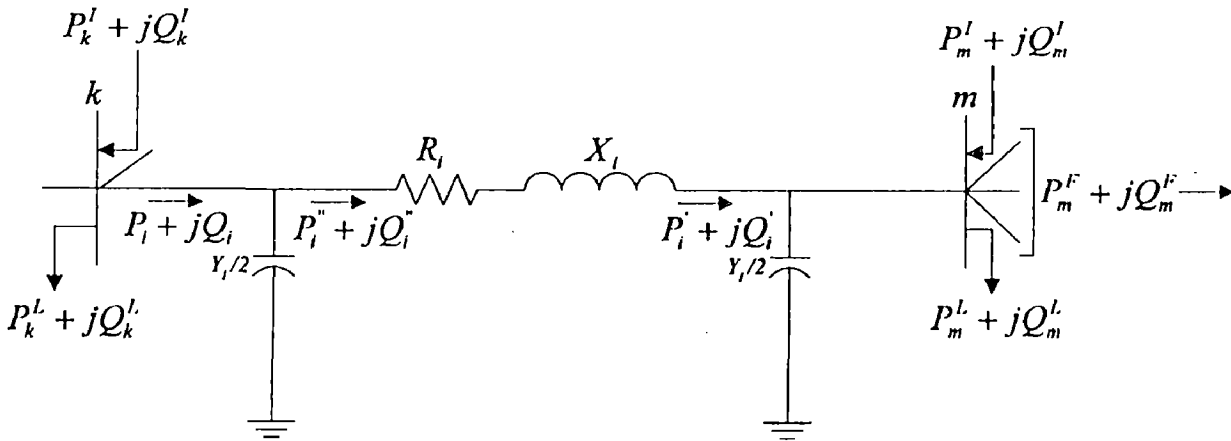


Figure 2.5:  $\pi$  (PI) circuit model of branch.

It is assumed that bus  $k$  is nearer to the root bus. The resistance and reactance of the branch are  $R_i$  &  $X_i$  respectively and the shunt charging admittance is denoted by  $y_i$ . The power flow through the series impedance can be written as,

$$P_i' = P_m^L + P_m^F - P_m^I \quad \dots (2.3)$$

$$Q_i' = Q_m^L + Q_m^F - Q_m^I - V_m^2 y_i / 2 \quad \dots (2.4)$$

Where  $P_i'$  and  $Q_i'$  are real & reactive power flow over the branch respectively. The superscripts  $L$ ,  $F$ , and  $I$  in  $P$  and  $Q$  represent the load, flow, and the injection respectively. The flow  $P_m^F$  ( $Q_m^F$ ) is the sum of active (reactive) power flow through all

the down stream branches that are emanating from the bus **m**. The procedure for finding the power injections ( $P_m^I$  and  $Q_m^I$ ) at the LBPs is described in the next section. The active power ( $P_i$ ) and the reactive power ( $Q_i$ ) flow through the branch at bus **k** can be written as,

$$P_i = P_i' + R_i(P_i'^2 + Q_i'^2)/V_m^2 \quad \dots (2.5)$$

$$Q_i = Q_i' + X_i(P_i'^2 + Q_i'^2)/V_m^2 - V_k^2 y_l/2 \quad \dots (2.6)$$

The above equations (2.5) & (2.6) are applied in a backward direction to compute the power flow through each branch in the tree. The equations are first applied to the last branch of the tree and proceed in reverse direction until the first branch is reached. After computing the power flow through each branch, the voltage magnitude and the angle at each bus is obtained by another set of recursive equations in forward direction.

Assuming that the voltage angle at bus **k** is zero, the voltage at the bus **m** can be written as (refer Fig.2.5):

$$\begin{aligned} V_m &= V_k - I_l Z_l \\ &= V_k - (S_l''/V_k)(R_l + jX_l) \\ &= V_k - (P_l'' - jQ_l'')/V_k (R_l + jX_l) \\ &= (V_k^2 - P_l'' R_l + Q_l'' X_l)/V_k - j(P_l'' X_l - Q_l'' R_l)/V_k \end{aligned} \quad \dots (2.7)$$

Where,  $S_l'' = P_l'' + jQ_l''$

$$P_l'' = P_l$$

$$Q_l'' = Q_l + V_k^2 y_l/2$$

and  $I_l$  is the current flowing through the series impedance ( $R_l + jX_l$ ).

Now, from the equation (2.7), the expression for voltage magnitude at bus **m** can be written as,

$$V_m = \sqrt{V_k^2 - 2(P_l'' R_l + Q_l'' X_l) + (P_l''^2 + Q_l''^2)(R_l^2 + X_l^2)/V_k^2} \quad \dots (2.8)$$

and the expression for voltage angle at bus **m** can be written as,

$$\delta_m = -\tan^{-1}(a_1/a_2) \quad \dots (2.9)$$

Where,  $a_1 = (P_l'' X_l - Q_l'' R_l)/V_k$  and

$$a_2 = V_k - (P_l'' R_l + Q_l'' X_l)/V_k$$

Now, if the voltage at the bus **k** is  $\delta_k$  (instead of zero), the angle  $\delta_m$  becomes,

$$\delta_m = \delta_k - \tan^{-1}(a_1/a_2) \quad \dots (2.10)$$

Now, the equations (2.8) and (2.10) are applied in forward direction i.e. they are applied at the first node and proceeds in the forward direction till the last node is reached.

## 2.5 Power Injections at the LBPs

In this algorithm, the voltage differences at the LBPs are calculated in each iteration. After getting the voltage difference, the current and accordingly the power injections at the LBPs are computed with the help of reduced order bus impedance matrix  $Z_{red}$ . The rank of  $Z_{red}$  is same as the number of loops  $n_{LP}$ . Further, it is assumed that the root bus has constant terminal voltage with negligible internal resistance, is capable enough to supply all the loads and losses in the system. The procedure is explained as follows.

The node equations of the system can be written as,

$$[I] = [Y] [V] \quad \dots(2.11)$$

The root bus is not considered in equation (2.11) as it is connected to the reference bus through a negligible (or zero) impedance. In deriving the admittance matrix, the loads in the system are replaced by constant shunt admittances at a nominal voltage of 1.0. p.u. i.e,

$$Y_{load} = S^*_{load}$$

The power injections to third set of the buses are zero because there is no LBP at these buses and the effect of loads have been considered in the Y-matrix. Hence we can eliminate the third set of the buses by Kron reduction [13] as shown below.

From equation (2.11) we can write,

$$\begin{bmatrix} I_a \\ I_b \\ 0 \end{bmatrix} = \begin{bmatrix} Y_{aa} & Y_{ab} & Y_{ac} \\ Y_{ba} & Y_{bb} & Y_{bc} \\ Y_{ca} & Y_{cb} & Y_{cc} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} \quad \dots(2.12)$$

$$\text{Or,} \quad I_a = Y_{aa}V_a + Y_{ab}V_b + Y_{ac}V_c \quad \dots(2.13)$$

$$I_b = Y_{ba}V_a + Y_{bb}V_b + Y_{bc}V_c \quad \dots(2.14)$$

$$0 = Y_{ca}V_a + Y_{cb}V_b + Y_{cc}V_c \quad \dots(2.15)$$

From equation (2.15), we can write

$$V_c = -Y_{cc}^{-1}(Y_{ca}V_a + Y_{cb}V_b) \quad \dots(2.16)$$

By substituting equation (2.16) in equation (2.13), we get,

$$I_a = Y_{aa}V_a + Y_{ab}V_b - Y_{ac} Y_{cc}^{-1}(Y_{ca}V_a + Y_{cb}V_b) \quad \dots(2.17)$$

$$\begin{aligned} &= (Y_{aa} - Y_{ac} Y_{cc}^{-1} Y_{ca})V_a + (Y_{ab} - Y_{ac} Y_{cc}^{-1} Y_{cb})V_b \\ &= Y_{AA}V_a + Y_{AB}V_b \end{aligned} \quad \dots(2.18)$$

where,  $Y_{AA} = Y_{aa} - Y_{ac} Y_{cc}^{-1} Y_{ca}$  and

$$Y_{AB} = Y_{ab} - Y_{ac} Y_{cc}^{-1} Y_{cb}$$

By substituting equation (2.16) in equation (2.14), we get

$$I_b = Y_{ba}V_a + Y_{bb}V_b - Y_{bc} Y_{cc}^{-1}(Y_{ca}V_a + Y_{cb}V_b) \quad \dots(2.19)$$

$$\begin{aligned} &= (Y_{ba} - Y_{bc} Y_{cc}^{-1} Y_{ca})V_a + (Y_{bb} - Y_{bc} Y_{cc}^{-1} Y_{cb})V_b \\ &= Y_{BA}V_a + Y_{BB}V_b \end{aligned} \quad \dots(2.20)$$

where,  $Y_{BA} = Y_{ba} - Y_{bc} Y_{cc}^{-1} Y_{ca}$  and

$$Y_{BB} = Y_{bb} - Y_{bc} Y_{cc}^{-1} Y_{cb}$$

Finally, the reduced system consists of first two sets of buses (**set-a** and **set-b**) and hence the corresponding node equation becomes,

$$\begin{bmatrix} I_a \\ I_b \end{bmatrix} = \begin{bmatrix} Y_{AA} & Y_{AB} \\ Y_{BA} & Y_{BB} \end{bmatrix} \begin{bmatrix} V_a \\ V_b \end{bmatrix} \quad \dots(2.21)$$

or

$$\begin{bmatrix} V_a \\ V_b \end{bmatrix} = \begin{bmatrix} Z_1 & Z_2 \\ Z_3 & Z_4 \end{bmatrix} \begin{bmatrix} I_a \\ I_b \end{bmatrix} \quad \dots(2.22)$$

Now, we can write the voltage difference  $[V_a - V_b] = [V_{ab}]$

$$\text{as } [V_{ab}] = [Z_1 - Z_3] [I_a] + [Z_2 - Z_4] [I_b] \quad \dots(2.23)$$

The current or power injection at the LBPs (**set-a** and **set-b**) are equal but opposite in sign i.e.

$$[I_a] = -[I_b] \quad \dots(2.24)$$

Therefore, we can write,

$$[V_{ab}] = [Z_1 - Z_3 - Z_2 + Z_4] [I_a]$$

$$\text{Or } [V_{ab}] = [Z_{red}] [I_a] \quad \dots(2.25)$$

where,  $Z_{red} = Z_1 - Z_3 - Z_2 + Z_4$ .



As equation (2.25) is linear in nature, in terms of incremental quantities, equation (2.25) can be written as,

$$[\Delta V_{ab}] = [Z_{red}] [\Delta I_a] \quad \dots(2.26)$$

Now, the incremental voltage difference  $[\Delta V_{ab}]$  is the difference between the specified and the calculated values. The specified voltage difference  $[V_{ab}]$  at the LBPs is always zero. If the value of  $[\Delta V_{ab}]$  is known during the iteration process, we can calculate the value of  $[\Delta I_a]$  from equation (2.26).

Therefore, the changes in power injection at the first set of buses (**set-a**) can be found out by,

$$[\Delta S_a] = [V_a] [\Delta I_a]^* \quad \dots(2.27)$$

where,  $[V_a]$  is a diagonal matrix. After each iteration  $p$ , the active and reactive power injections at loop break points can be updated as follows,

$$P_{(p+1)}^I = P_{(p)}^I + \Re(\Delta S_a) \quad \dots(2.28)$$

$$Q_{(p+1)}^I = Q_{(p)}^I + \Im(\Delta S_a) \quad \dots(2.29)$$

Where,  $\Re(\cdot)$  and  $\Im(\cdot)$  denote the real and imaginary parts of the quantity  $(\cdot)$  respectively. The power injection at buses lying in the second set (**set-b**) can be computed similarly with opposite sign. In the next section the step-by-step solution algorithm is shown in detail.

## 2.6 Solution Algorithm

The steps involved for finding the load flow solution of a single source mesh network are given below:

- Step 1.** Read the system data.
- Step 2.** Create loop break points for mesh to radial conversion. Construct tree network in layers and number the branches.
- Step 3.** Order the buses. Divide the buses into three sets; **set-a**, **set-b** and **set-c**.
- Step 4.** Assume initial voltage for all the buses except the root bus.
- Step 5.** Compute the matrix  $Y_{aa}$ ,  $Y_{ab}$ ,  $Y_{ac}$ ,  $Y_{ba}$ ,  $Y_{bb}$ ,  $Y_{bc}$ ,  $Y_{ca}$ ,  $Y_{cb}$ , and  $Y_{cc}$ .
- Step 6.** Find the matrix  $[Z_{red}]$  using equations (2.21), (2.22) and (2.25).
- Step 7.** Assume initial power injections at the LBPs equal to Zero.

- Step 8.** Set iteration count  $K=1$ .
- Step 9.** Compute the active and reactive power flow through each branches of the tree network by equations (2.5) & (2.6) respectively. It is done in backward direction i.e. the computation starts at the last branch and stops at the first branch.
- Step 10.** Compute the voltage magnitude at the buses by equation (2.8) in forward direction i.e. the computation starts at the first node and stops at the last node. Also, compute the voltage angle in the same fashion by using equation (2.10).
- Step 11.** Compute the incremental voltage difference  $[\Delta V_{ab}]$  at the LBPs. If the value of  $[\Delta V_{ab}]$  is within the specified tolerance, then the load flow has converged. Hence, stop the iterations and print the results. Otherwise go to step 12.
- Step 12.** Update the active and reactive power injections at the LBPs using equations (2.28) & (2.29) respectively. Increment the iteration count by 1 and go back to step 9.

### RESULTS AND DISCUSSION

In order to test the validity of the solution algorithm developed in Chapter-2, two mesh connected systems of different sizes were adopted from [14] and [15], namely, i) 23 kV, 19-bus mesh system and ii) 12.66 kV, 22-bus mesh system. The system data for these two systems are given in Table A.1 and Table A.2 in Appendix-A respectively. For convergence, a tolerance limit of 0.000001 p.u. on the voltage magnitude has been specified. Moreover, a flat voltage profile (i.e. voltage magnitude of 1.0 p.u and the angle of zero degree) has been assumed as the initial value of all the bus voltages.

The results obtained for both the systems i.e. 19-bus network and 22-bus network are summarized and presented in the following sections.

#### 3.1 23 kV, 19-bus mesh network

The single line diagram of this system is shown in Fig. 3.1. Following the bus numbering scheme presented in Chapter-2, it was found that two LBPs are present at bus number 11 and 17. Thus, before solving the load flow problem, the meshed network is first converted into an equivalent radial network by opening the loops at bus number 11 and 17. In this process, two additional dummy buses have been created. These additional dummy buses have been numbered as 19 and 20. The equivalent radial network is shown in Fig. 3.2.

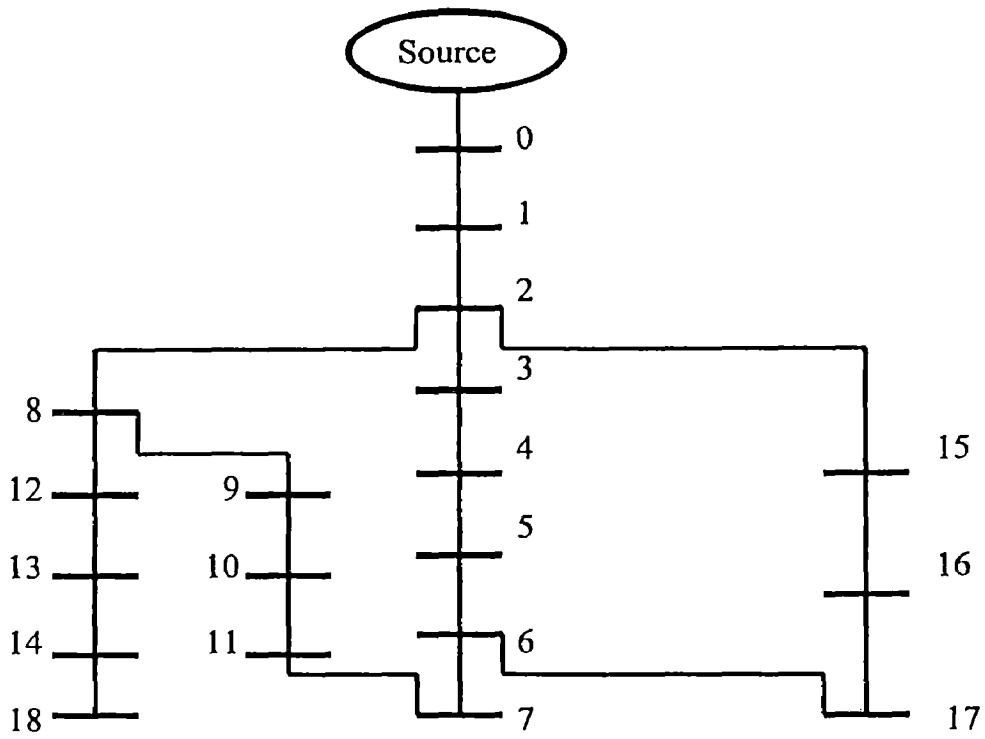


Figure 3.1: Schematic diagram of 19-Bus mesh network

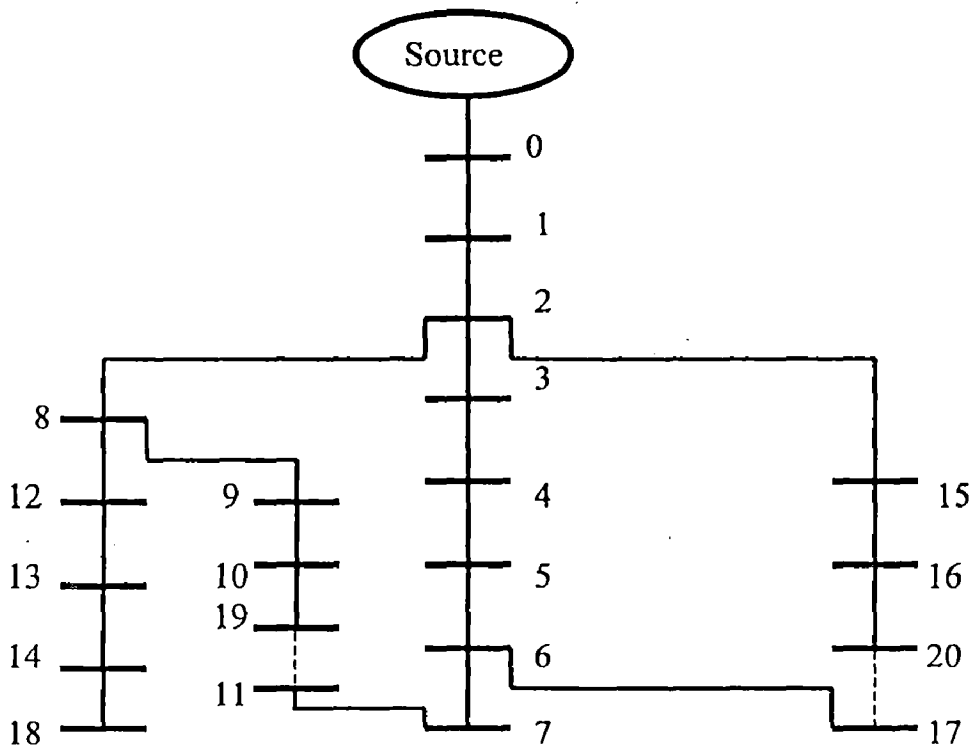


Figure 3.2: Schematic diagram of equivalent radial network of Fig. 3.1

The voltage magnitude at the root bus was considered to be 1.0 p.u. The algorithm took 4 iterations to converge and results are shown Table 3.1. The system data are given in Table A.1 in Appendix-A. Table 3.1 shows the voltage magnitude for the remaining buses of the system. It is expected that the voltages at the two buses representing a LBP would be identical after final solution of the system. From Table 3.1, it is observed that the voltage solutions of bus 11 and 19 (which are representing the LBP at bus 11 in the original meshed network) are identical. Similarly, the voltage solutions of bus 17 and 20 (which are representing the LBP at bus 17 in the original meshed network) are also identical. Thus, the developed algorithm is able to provide satisfactory results for the load flow solution of the meshed network.

**Table 3.1: Load Flow Solution for 19-Bus System**

Bus No.	Bus Voltage	
	Magnitude (p.u.)	Voltage Angle (deg.)
1	1	0
2	0.996716	-0.50663
3	0.995268	-0.50844
4	0.994337	-0.50996
5	0.993682	-0.51147
6	0.993303	-0.51299
7	0.992997	-0.51365
8	0.994301	-0.51077
9	0.993707	-0.51176
10	0.993258	-0.51262
11	0.992946	-0.51351
12	0.992929	-0.51373
13	0.991887	-0.5167
14	0.991369	-0.51846
15	0.994932	-0.51093
16	0.993982	-0.51286
17	0.993419	-0.51346
18	0.991109	-0.51934
19	0.992946	-0.51351
20	0.993419	-0.51346

### **3.2 12.66 kV, 22-bus mesh network**

To verify the validity of the developed algorithm further, a 22-bus meshed network [15] has also been considered. The single line diagram of this system is shown in Fig.3.3. The system data are given in Table A.2 in Appendix-A. The corresponding equivalent radial network obtained after breaking the loops is shown in Fig. 3.4. Loop break points are created at bus number 14 and 21. The dummy buses added are bus number 22 and 23 (refer Fig. 3.4).

The voltage magnitude at the root bus was considered to be 1.0 p.u. The algorithm took 5 iterations to converge and results are shown Table 3.2. From Table 3.2 it is observed that the voltage solutions of buses 14 and 23 (which are representing the LBP at bus 14 in the original meshed network) are identical. Similarly, the voltage solutions of buses 21 and 22 (which are representing the LBP at bus 21 in the original meshed network) are also identical. In this case also the developed algorithm has provided satisfactory results for the load flow solution of the meshed network.

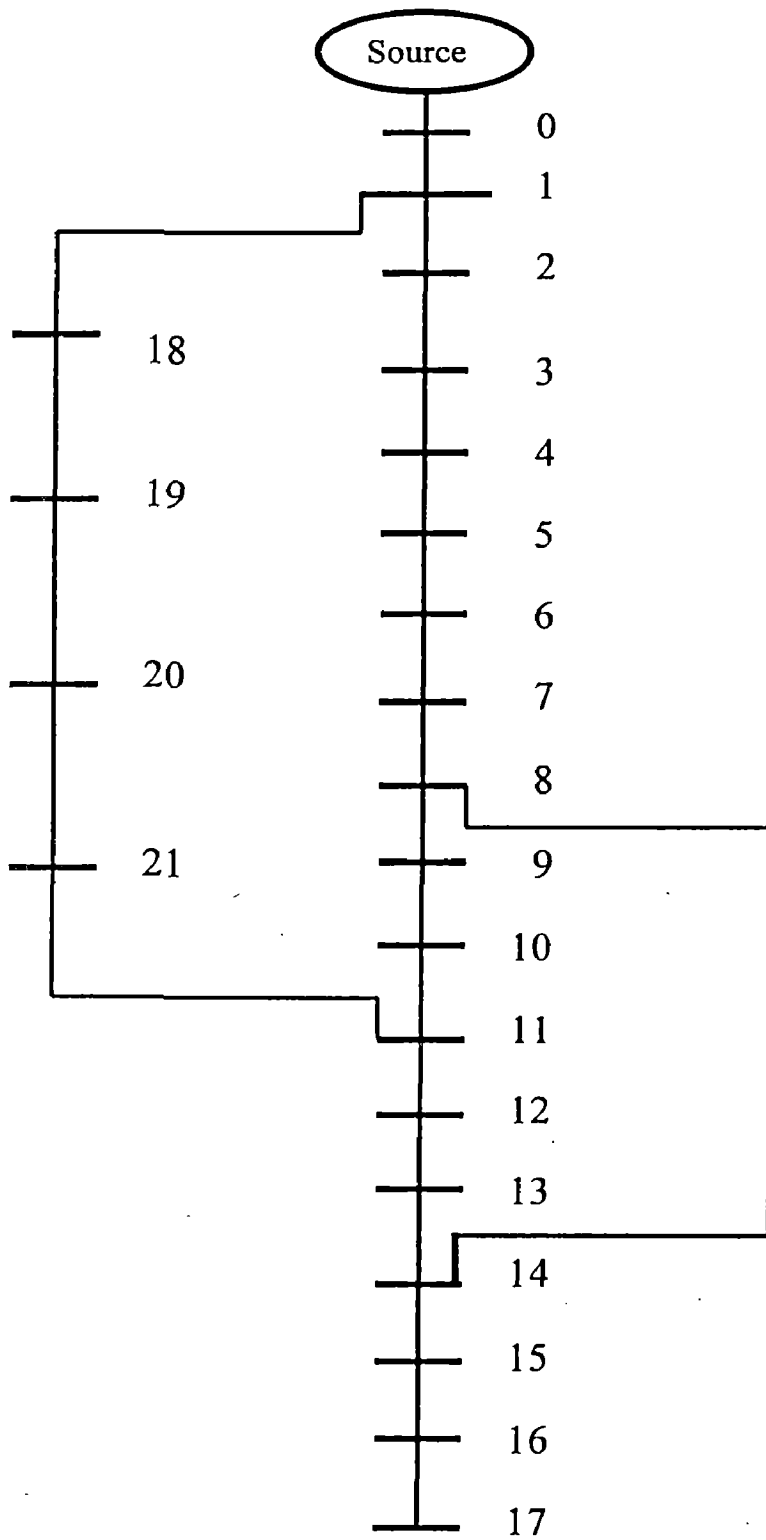


Figure 3.3: Schematic Diagram of 22-Bus mesh network.

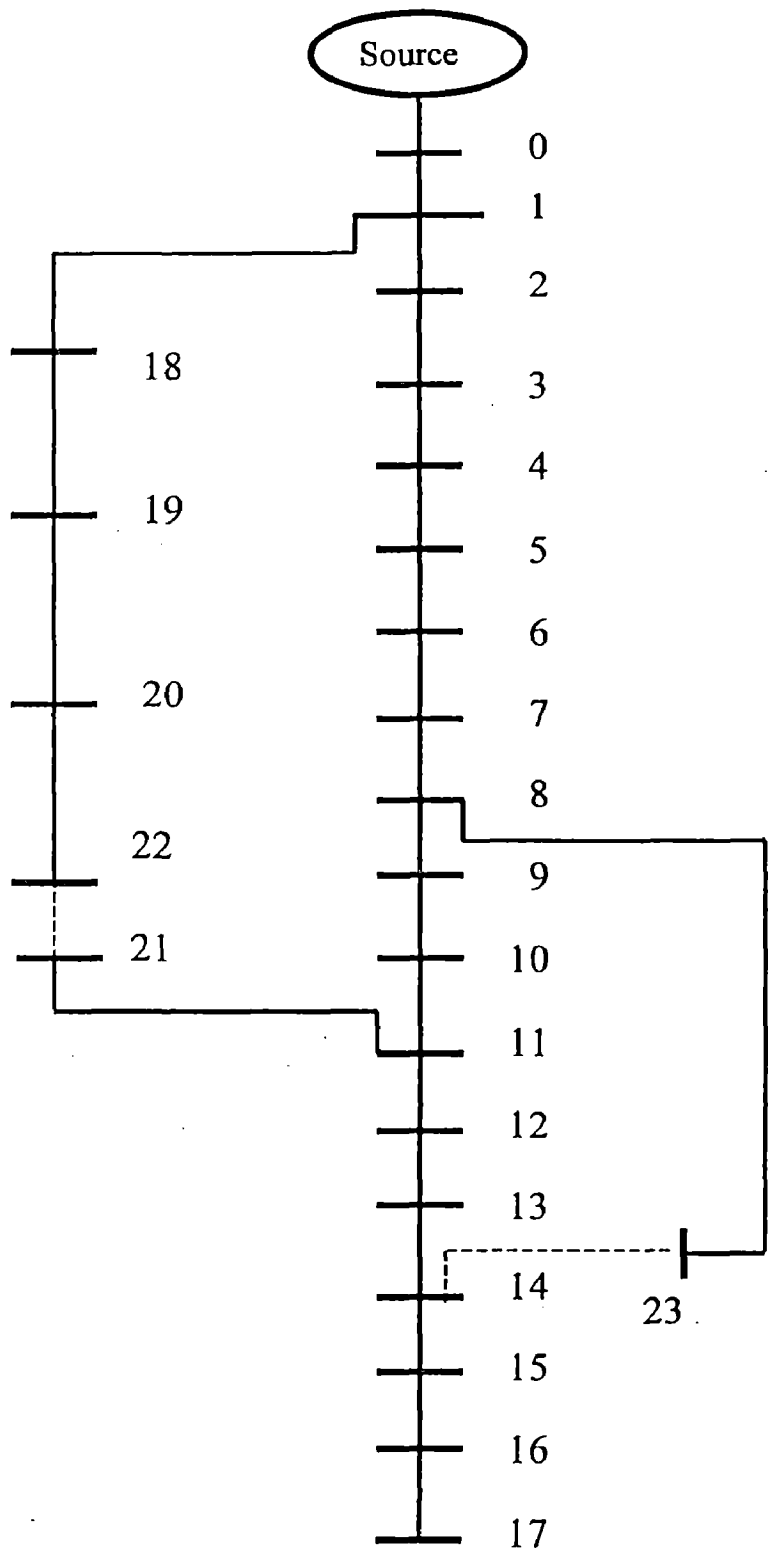


Figure 3.4: Schematic diagram of equivalent radial network of Fig 3.3



**Table 3.2: Load Flow Solution for 22-Bus System**

Bus No.	Bus Voltage	
	Magnitude (p.u.)	Voltage Angle (deg.)
1	0.998636	-0.0017
2	0.994535	-0.0129
3	0.991736	-0.02142
4	0.989208	-0.03297
5	0.983563	-0.14397
6	0.981481	-0.29345
7	0.978717	-0.28722
8	0.975983	-0.34245
9	0.97559	-0.36451
10	0.975706	-0.36811
11	0.976272	-0.35827
12	0.973207	-0.39732
13	0.972234	-0.4283
14	0.972026	-0.43945
15	0.970735	-0.46015
16	0.968824	-0.52888
17	0.968251	-0.53741
18	0.997543	-0.01897
19	0.988896	-0.13601
20	0.986665	-0.18289
21	0.983243	-0.26473
22	0.983243	-0.26473
23	0.972026	-0.43945

---

### CONCLUSION

In this dissertation, an effort has been made to develop an algorithm for load flow analysis of weakly meshed distribution system network. In this algorithm, the original meshed configuration is first converted to an equivalent radial configuration by breaking the loops. In this process, dummy buses are added to the LBPs. To preserve the original characteristics of the network after conversion, a compensatory power is injected to the both sides of a loop break point (LBP). The injected power at the LBPs should be equal in magnitude with opposite sign at the two buses of a LBP. The power injections at the LBPs are computed with the help of a reduced order bus impedance matrix. The solution algorithm has been tested on a 23 kV, 19-bus mesh distribution system and a 12.66 kV, 22-bus mesh distribution system. It has been observed that this method bears an excellent convergence behavior and provides quite satisfactory results.

## REFERENCES

---

1. TINNEY, W.G., and HART, C. E., 'Power flow solutions by Newton's method', *IEEE Transactions on Power Apparatus and System*, Vol-86, 1967, pp: 1449-1457.
2. STOT, B., and ALSAC, O., 'Fast Decoupled Load Flow', *IEEE Transactions on Power Apparatus and System*, Vol-93, 1974, pp: 859-869.
3. VAN AMERONGEN, R. A. M., 'A general purpose version of the Fast Decoupled Load Flow', *IEEE Transactions on Power System*, Vol-4, No-2, 1989, pp: 760-766.
4. RAJICIC, D., BOSE, A., 'A modification to the Fast Decoupled Power Flow for networks with high R/X ratios', *IEEE Transactions on Power System*, Vol-3, No-2, 1988, pp : 743-746.
5. HAQUE, M. H., 'Novel Decoupled Load Flow Method', *IEE Proc.*, Part-C, Vol-140, No-3, 1993, pp: 199-205.
6. DAS, D., NAGI, H.S., and KOTHARI, D. P., 'Novel method for solving radial distribution networks', *IEE Proc.*, Part C, Vol-141, No-4, 1994, pp : 291-298.
7. PAPADOPOULUS, M., HATZIARGYRIOU, N.D., and PAPADAKIS, M.E., 'Graphics aided interactive analysis of radial distribution networks', *IEEE Transactions on Power Delivery*, Vol-2, October 1987, pp : 1297-1302.
8. RENATO, C.G., 'New method for the analysis of distribution networks', *IEEE Transactions on Power Delivery*, Vol-5, No-1, 1990, pp : 391-396.
9. GASWAMI, S.K., and BASU, S.K., 'Direct solution of distribution systems', *IEE Proc.*, Part- C, Vol-138, No-1, 1991, pp: 78-88.
10. SHIRMOHAMMADI, D., HONG, H.W., SEMLYEN, A., and LUO, G. X., 'A compensation-based power flow method for weakly meshed distribution and transmission networks', *IEEE Transactions on Power System* ,Vol-3, No-2, 1988, pp : 753-762.

11. LUO, G. X., and SEMLYEN, A., 'Efficient load flow for large weakly meshed networks', *IEEE Transactions on Power System*, Vol-5, No-4, 1990, pp : 1309-1326.
12. HAQUE, M. H., 'Efficient load flow method for distribution systems with radial or mesh configuration', *IEE Proc., Part-C*, Vol-143, No-2, January 1996, pp :33-38
13. ANDERSON, P.M., and Fouad, A.A., 'Power system control and stability', (Iowa State University Press, Ames, Iowa, 1977).
14. BARAN, M.E., and WU, F. F., 'Network reconfiguration in distribution systems for loss reduction and load balancing', *IEEE Transactions on Power Delivery*, Vol-4, No-2, 1989, pp: 1401-1407.
15. SADHUKHAN, B., 'State Estimation in power distribution system', M. Tech, Dissertation, Water Resources Development Training Centre, IIT, Roorkee, 2001.

## APPENDIX-A

**Table A.1: System Data for 19-Bus System**

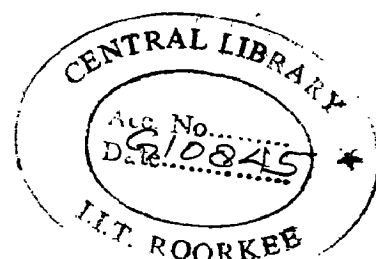
**Base Voltage = 23 kV, Base kVA = 500 kVA**

Line No.	From Bus (i)	To Bus (j)	Line Data		Load at Bus (j)	
			Resistance (ohm)	Reactance (ohm)	Active (p.u.)	Reactive (p.u.)
1	0	1	0.00	0.00	0.0	0.0
2	1	2	0.00	0.55	1.0	0.4
3	2	3	0.30	0.12	1.0	0.4
4	2	8	0.30	0.12	1.0	0.4
5	2	15	0.40	0.16	1.2	0.4
6	3	4	0.25	0.10	1.0	0.4
7	4	5	0.25	0.10	1.0	0.4
8	5	6	0.25	0.10	0.8	0.3
9	6	7	0.25	0.10	0.9	0.3
10	6	17	0.20	0.08	1.2	0.4
11	7	11	0.20	0.08	0.8	0.3
12	8	9	0.25	0.10	0.8	0.3
13	8	12	0.30	0.12	1.0	0.4
14	9	10	0.30	0.12	0.8	0.3
15	10	11	0.50	0.20	0.8	0.3
16	12	13	0.30	0.12	0.8	0.3
17	13	14	0.20	0.08	1.2	0.4
18	14	18	0.20	0.08	1.2	0.4
19	15	16	0.30	0.12	1.2	0.4
20	16	17	0.30	0.12	1.2	0.4

**Table A.2: System-Data for 22-Bus System**

**Base Voltage = 12.66 kV, Base kVA = 100 kVA**

Line No.	From Bus (i)	To Bus (j)	Line data		Load at Bus (j)	
			Resistance (p.u.)	Reactance (p.u.)	Active (p.u.)	Reactive (p.u.)
1	0	1	5.75E-05	5.75E-05	1.00	0.60
2	1	2	0.000308	0.000155	0.90	0.40
3	1	18	0.000102	9.77E-05	0.90	0.40
4	2	3	0.000228	0.000116	1.20	0.80
5	3	4	0.000238	0.000121	0.60	0.30
6	4	5	0.000511	0.000441	0.60	0.20
7	5	6	0.000117	0.000386	2.00	1.00
8	6	7	0.000444	0.000147	2.00	1.00
9	7	8	0.000643	0.000462	0.60	0.20
10	8	9	0.000651	0.000462	0.60	0.20
11	9	10	0.000123	0.000406	0.45	0.30
12	10	11	0.000234	0.000772	0.60	0.35
13	11	12	0.000916	0.000721	0.60	0.35
14	11	21	0.001248	0.001248	0.90	0.40
15	12	13	0.000338	0.000445	1.20	0.80
16	13	14	0.000369	0.000328	0.60	0.10
17	14	15	0.000466	0.00034	0.60	0.20
18	15	16	0.000804	0.001074	0.60	0.20
19	16	17	0.000457	0.000358	0.90	0.40
20	18	19	0.000939	0.000846	0.90	0.40
21	19	20	0.000255	0.000298	0.90	0.40
22	20	21	0.000442	0.000585	0.90	0.40
23	8	14	0.001248	0.001275	0.60	0.10



```
// ***** PROGRAM FOR LOAD FLOW ANALYSIS OF WEAKLY MESHED
DISTRIBUTION SYSTEM ***** //
#include<stdio.h>
#include<alloc.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
#define m1 21
#define N 21
// ***** FUNCTION FOR MATRIX-ADDITION *****
void addition(float A[][N],float B[][N],float C[][N],int p,int q)
{
int i,j;
for(i=0;i<p;i++)
for(j=0;j<q;j++)
C[i][j]=(A[i][j]+B[i][j]);
}
// ***** FUNCTION FOR MATRIX-SUBTRACTION *****
void subtraction(float A[][N],float B[][N],float C[][N],int p,int q)
{
int i,j;
for(i=0;i<p;i++)
for(j=0;j<q;j++)
C[i][j]=(A[i][j]-B[i][j]);
}
// ***** FUNCTION FOR MATRIX-MULTIPLICATION *****
void mul(float A[][N],int n,float B[][N],int p,float C[][N],int q)
{
int i,j,k;
for(i=0;i<n;i++)
for(j=0;j<p;j++)
{
C[i][j]=0.00;
for(k=0;k<q;k++)
C[i][j]+=(A[i][k]*B[k][j]);
}
}
// ***** FUNCTION FOR MATRIX-INVERSION *****
void inverse(float xx[][N],int n)
{
int i,j,k,l;
for(i=0;i<n;i++)
{
```

```

for(j=0;j<n;j++)
for(k=0;k<n;k++)
if((j!=i)&&(k!=i))
xx[j][k] -= xx[j][i]*xx[i][k]/xx[i][i];
xx[i][i] = -(1.0/xx[i][i]);
for(l=0;l<n;l++)
{
if(l==i) continue;
xx[l][i] *= xx[i][i];
xx[i][l] *= xx[i][i];
}
}
for(i=0;i<n;i++)
for(j=0;j<n;j++)
xx[i][j]= -xx[i][j];
}
// ***** FUNCTION FOR COMPLEX MATRIX-ADDITION *****
void complex_addition(float Ar[][N],float Ai[][N],float Br[][N],float Bi[][N],float
Cr[][N],float Ci[][N],int p,int q)
{
int i,j;
for(i=0;i<p;i++)
for(j=0;j<q;j++)
{
Cr[i][j]=(Ar[i][j]+Br[i][j]);
Ci[i][j]=(Ai[i][j]+Bi[i][j]);
}
}
// ***** FUNCTION FOR COMPLEX MATRIX-SUBTRACTION *****
void complex_subtraction(float Ar[][N],float Ai[][N],float Br[][N],float Bi[][N],float
Cr[][N],float Ci[][N],int p,int q)
{
int i,j;
for(i=0;i<p;i++)
for(j=0;j<q;j++)
{
Cr[i][j]=(Ar[i][j]-Br[i][j]);
Ci[i][j]=(Ai[i][j]-Bi[i][j]);
}
}
// ***** FUNCTION FOR COMPLEX MATRIX-MULTIPLICATION *****
void complex_mul(float Ar[][N],float Ai[][N],int n,float Br[][N],float Bi[][N],int p,float
Cr[][N],float Ci[][N],int q)
{
float Dr[N][N],Di[N][N];
mul(Ar,n,Br,p,Dr,q);

```



```

mul(Ai,n,Bi,p,Di,q);
subtraction(Dr,Di,Cr,n,p);
mul(Ar,n,Bi,p,Dr,q);
mul(Ai,n,Br,p,Di,q);
addition(Dr,Di,Ci,n,p);
}
// ***** FUNCTION FOR COMPLEX MATRIX-INVERSION *****
void complex_inverse(float A[][N],float B[][N],float C[][N],float D[][N],int n)
{
int i,j;
float Yb1[N][N],Yc1[N][N],Yc2[N][N];
for(i=0;i<n;i++)
for(j=0;j<n;j++)
Yb1[i][j]=B[i][j];
inverse(Yb1,n);
mul(Yb1,n,A,n,Yc1,n);
for(i=0;i<n;i++)
for(j=0;j<n;j++)
Yc1[i][j]=-Yc1[i][j];
mul(A,n,Yc1,n,Yc2,n);
subtraction(Yc2,B,D,n,n);
inverse(D,n);
mul(Yc1,n,D,n,C,n);
}
// ***** FUNCTION FOR SORTING AN ELEMENT FROM AN ARRAY *****
int check_array(int a,int A[N],int n)
{
int i,k;
i=0;
k=0;
while(i<n)
{
if(a!=A[i])
k++;
i++;
}
if(k==n)
return(a);
else
return(-1);
}
// ***** FUNCTION FOR ARRANGING ARRAY-ELEMENTS IN INCREASING
ORDER *****
void serial(int A[N],int n)
{
int temp,i,j;

```

```

for(i=1;i<n;i++)
for(j=0;j<(n-i);j++)
if(A[j]>A[j+1])
{
temp=A[j];
A[j]=A[j+1];
A[j+1]=temp;
}
}
// ***** FUNCTION FOR CALCULATION OF Y-MATRIX *****
void calculate(float A[][7],int n,int B[N],int b,int C[N],int c,float D[][N],float E[][N])
{
float temp;
int i,j,k,ii,jj,connection[m1];
for(i=0;i<b;i++)
{
for(j=0;j<c;j++)
{
D[i][j]=0.00;
E[i][j]=0.00;
if(B[i]!=C[j])
{
for(k=0;k<n;k++)
if(((A[k][1]==B[i])&&(A[k][2]==C[j]))||((A[k][2]==B[i])&&(A[k][1]==C[j])))
{
temp=pow(A[k][3],2)+pow(A[k][4],2);
D[i][j]=(-A[k][3]/temp);
E[i][j]=(A[k][4]/temp);
}
}
else
{
ii=1;
connection[0]=B[i];
for(k=0;k<n;k++)
{
if((A[k][1]==B[i]))
if(check_array(int(A[k][2]),connection,ii)!=-1)
{
connection[ii]=int(A[k][2]);
ii++;
}
if((A[k][2]==B[i]))
if(check_array(int(A[k][1]),connection,ii)!=-1)
{
connection[ii]=int(A[k][1]);
}
}
}
}
}

```

```

ii++;
}
} //
for(jj=1;jj<ii;jj++)
{
for(k=0;k<n;k++)
if(((A[k][1]==connection[0])&&(A[k][2]==connection[jj]))||((A[k][2]==connection[0])
&&(A[k][1]==connection[jj])))
{
temp=pow(A[k][3],2)+pow(A[k][4],2);
D[i][j]+=(A[k][3]/temp)+A[k][5];
E[i][j]+=(-A[k][4]/temp)+A[k][6];
}
}
}
}
}
}
}
}
}
void check_for_connection(float A[][7],int n,int a,int B[N],int *b)
{
int i,k,ii;
B[0]=a;
for(i=1;i<m1;i++)
B[i]=-1;
ii=1;
for(k=0;k<n;k++)
{
if((A[k][1]==a)
if(check_array(int(A[k][2]),B,ii)!=-1)
{
B[ii]=int(A[k][2]);
ii++;
}
}
*b=ii-1;
}
}
void node_position(float A[][7],int n,int a,int *b)
{
int k,ii;
for(k=0;k<n;k++)
if((A[k][0]==a))
*b=k;
}
void main()
{
float branch_data[m1][7],node_data[m1][7],temp,Y1r[N][N],Y1i[N][N],Y2r[N][N],

```

```

Y2i[N][N],Y3r[N][N],Y3i[N][N];
float Yaar[N][N],Yaai[N][N],Yabr[N][N],Yabi[N][N],Yacr[N][N],Yaci[N][N];
float Ybbr[N][N],Ybbi[N][N],Ybcr[N][N],Ybci[N][N],Yccr[N][N],Ycci[N][N];
float YAAr[N][N],YAAi[N][N];
float Ykronr[N][N],Ykroni[N][N],Zkronr[N][N],Zkroni[N][N];
float Zredr[N][N],Zredi[N][N];
float P_flow,Q_flow,diff_V[m1],pi,max,tolerance=0.000001,Sb,Vb;
int no_branch,no_buses,root,flag[m1],seta[N],setb[N],setc[N],setc1[N],no_seta,no_setb,
no_setc,connection[N],no_fconnection,bconnection1,bconnection2,root_connection,
layer[N][N],no_layer;
int i,j,k,ii,jj,kk,iii,jjj,kkk;
FILE *f1,*f2,*f3;
char s1[15];
clrscr();
pi=4*atan(1);
printf("\n          ***** WELCOME TO MY DISSERTATION *****");
printf("\n This Program is developed assuming Inital Bus Voltages and Bus Loadings are
in p.u. form but you can enter Branch Data either in p.u. form or in actual form");
printf("\n\nIf Branch-data are already in p.u.form then enter 1000 else enter base-kVA:-");
scanf("%f",&Sb);
printf("\n\nIf Branch-data are already in p.u.form then enter 1.00 else enter base-kV:-");
scanf("%f",&Vb);
printf("Enter Input File Name :");
scanf("%s",s1);
f1=fopen(s1,"r");
printf("Enter Output File Name :");
scanf("%s",s1);
f2=fopen(s1,"w");
// ***** DATA INPUTTING FROM FILE *****
fscanf(f1,"%d %d %d",&no_buses,&no_branch,&root);
fprintf(f2,"\t***** LINE TOPOLOGY BEFORE BREAKING THE
LOOPS*****\n");
for(j=0;j<no_branch;j++)
{
fscanf(f1,"%f %f %f %f %f %f
%f",&branch_data[j][0],&branch_data[j][1],&branch_data[j][2],&branch_data[j][3],&bra
nch_data[j][4],&branch_data[j][5],&branch_data[j][6]);
branch_data[j][3]/=(Vb*Vb*1000/Sb);
branch_data[j][4]/=(Vb*Vb*1000/Sb);
branch_data[j][5]*=(Vb*Vb*1000/Sb);
branch_data[j][6]*=(Vb*Vb*1000/Sb);
if(branch_data[j][1]>branch_data[j][2])
{
temp=branch_data[j][1];
branch_data[j][1]=branch_data[j][2];
branch_data[j][2]=temp;

```

```

}
fprintf(f2, "\nNode %d <---Line no %d,Z=%f+j(%f) and Y=%f+j(%f)---> Node
%d,",int(branch_data[j][1]),int(branch_data[j][0]),branch_data[j][3],branch_data[j][4],bra
nch_data[j][5],branch_data[j][6],int(branch_data[j][2]));
}
for(i=1;i<no_buses;i++)
{
fscanf(f1,"%f%f%f%f
%f",&node_data[i][0],&node_data[i][1],&node_data[i][2],&node_data[i][5],&node_data
[i][6]);
node_data[i][2]=pi*node_data[i][2]/180;
node_data[i][3]=0.00;
node_data[i][4]=0.00;
}
flag[0]=-1;
node_data[0][0]=root;
node_data[0][1]=node_data[1][1];
node_data[0][2]=node_data[1][2];
node_data[0][3]=0.00;
node_data[0][4]=0.00;
node_data[0][5]=0.00;
node_data[0][6]=0.00;
for(i=1;i<no_buses;i++)
flag[i]=0;
// ***** CREATION OF LOOP BREAK POINTS,CALCULAITON OF SET-A & SET-
B AND NUMBERING OF DUMMY BUSES *****
k=0;
fprintf(f2, "\n\n");
for(j=0;j<no_branch;j++)
if(flag[branch_data[j][2]]==-1)
{
seta[k]=int(branch_data[j][2]);
setb[k]=k+no_buses;
branch_data[j][2]=float(setb[k]);
node_data[seta[k]][3]=node_data[seta[k]][3]/2;
node_data[seta[k]][4]=node_data[seta[k]][4]/2;
node_data[seta[k]][5]=node_data[seta[k]][5]/2;
node_data[seta[k]][6]=node_data[seta[k]][6]/2;
node_data[k+no_buses][0]=float(setb[k]);
node_data[k+no_buses][1]=node_data[seta[k]][1];
node_data[k+no_buses][2]=node_data[seta[k]][2];
node_data[k+no_buses][3]=node_data[seta[k]][3];
node_data[k+no_buses][4]=node_data[seta[k]][4];
node_data[k+no_buses][5]=node_data[seta[k]][5];
node_data[k+no_buses][6]=node_data[seta[k]][6];
}

```

```

fprintf(f2, "\nLoop breakpoint created at Bus no-%d is Dummy bus no-
%d", seta[k], setb[k]);
k++;
}
else
flag[branch_data[j][2]]=-1;
no_seta=k;
no_setb=k;
fprintf(f2, "\n\n\n\t***** LINE TOPOLOGY AFTER BREAKING THE
LOOPS*****\n");
for(j=0;j<no_branch;j++)
{
fprintf(f2, "\nNode %d <---Line no %d,Z=%f+j(%f) and Ysh=%f+j(%f)---> Node
%d,", int(branch_data[j][1]),int(branch_data[j][0]),branch_data[j][3],branch_data[j][4],bra
nch_data[j][5],branch_data[j][6],int(branch_data[j][2]));
}
// ***** CALCULAITON OF SET-C *****
k=0;
for(j=1;j<no_buses+no_seta;j++)
{
if(check_array(node_data[j][0],seta,no_seta)!=-1)
{
setc1[k]=node_data[j][0];
k++;
}
}
i=0;
for(j=0;j<k;j++)
if(check_array(setc1[j],setb,no_setb)!=-1)
{
setc[i]=setc1[j];
i++;
}
no_setc=i;
fprintf(f2, "\n\nNo of buses in set a=%d", no_seta);
fprintf(f2, "\n\n\t\t***** SET A *****\n");
for(i=0;i<no_seta;i++)
fprintf(f2, "\nSet A[%d]=%d", i+1, seta[i]);
fprintf(f2, "\n\nNo of buses in set b=%d", no_setb);
fprintf(f2, "\n\n\t\t***** SET B *****\n");
for(i=0;i<no_setb;i++)
fprintf(f2, "\nSet B[%d]=%d", i+1, setb[i]);
fprintf(f2, "\n\nNo of buses in set c=%d", no_setc);
fprintf(f2, "\n\n\t\t***** SET C *****\n");
for(i=0;i<no_setc;i++)
fprintf(f2, "\nSet C[%d]=%d", i+1, setc[i]);

```

```

// ***** LAYERS CREATION *****
for(i=0;i<N;i++)
for(j=0;j<N;j++)
layer[i][j]=0;
check_for_connection(branch_data,no_branch,root,connection,&root_connection);
layer[0][0]=root_connection;
for(i=1;i<=layer[0][0];i++)
{
for(j=0;j<no_branch;j++)
if((branch_data[j][1]==root)&&(branch_data[j][2]==connection[i]))
layer[0][i]=branch_data[j][0];
}
ii=layer[0][0];
no_layer=1;
kk=0;
while(ii<no_branch)
{
jj=1;
for(i=1;i<=layer[kk][0];i++)
{
check_for_connection(branch_data,no_branch,int(branch_data[layer[kk][i]-
1][2]),connection,&no_fconnection);
for(k=1;k<=no_fconnection;k++)
{
for(j=0;j<no_branch;j++)
{
if((int(branch_data[layer[kk][i]-
1][2])==branch_data[j][1])&&(connection[k]==branch_data[j][2]))
{
layer[kk+1][jj]=branch_data[j][0];
jj++;
ii++;
} //if
} //j
} //k
layer[kk+1][0]+=no_fconnection;
} //for i
no_layer++;
kk++;
} // end of while
fprintf(f2,"\n\n\t***** LAYERS CREATED *****\n");
fprintf(f2,"\n\t\tNUMBER OF LINES\t\tLINE NUMBERS\n");
for(i=0;i<no_layer;i++)
{
fprintf(f2,"\nLAYER NO.-%d\t\t%d\t\t",i+1,layer[i][0]);
for(j=1;j<=layer[i][0];j++)

```

```

fprintf(f2," %d",layer[i][j]);
}
fprintf(f2,"\n\nNUMBER OF LAYERS CREATED = %d",no_layer);
// ***** CALCULATION OF Y-MATRIX *****
calculate(branch_data,no_branch,seta,no_seta,seta,no_seta,Yaar,Yaai);
calculate(branch_data,no_branch,seta,no_seta,setb,no_setb,Yabr,Yabi);
calculate(branch_data,no_branch,seta,no_seta,setc,no_setc,Yacr,Yaci);
calculate(branch_data,no_branch,setb,no_setb,setb,no_setb,Ybbr,Ybbi);
calculate(branch_data,no_branch,setb,no_setb,setc,no_setc,Ybcr,Ybci);
calculate(branch_data,no_branch,setc,no_setc,setc,no_setc,Yccr,Ycci);
// ***** PRINTING OF Y-MATRIX *****
fprintf(f2,"\n\n\t\t***** Yaa *****\n");
for(i=0;i<no_seta;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_seta;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",Yaar[i][j],Yaai[i][j]);
}
fprintf(f2,"\n\n\t\t***** Yab *****\n");
for(i=0;i<no_seta;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_setb;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",Yabr[i][j],Yabi[i][j]);
}
fprintf(f2,"\n\n\t\t***** Yac *****\n");
for(i=0;i<no_seta;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_setc;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",Yacr[i][j],Yaci[i][j]);
}
fprintf(f2,"\n\n\t\t***** Yba *****\n");
for(i=0;i<no_setb;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_seta;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",Yabr[j][i],Yabi[j][i]);
}
fprintf(f2,"\n\n\t\t***** Ybb *****\n");
for(i=0;i<no_setb;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_setb;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",Ybbr[i][j],Ybbi[i][j]);
}
}

```



```

fprintf(f2, "\n\n\n\t\t***** Ybc *****\n");
for(i=0;i<no_setb;i++)
{
fprintf(f2, "\n");
for(j=0;j<no_setc;j++)
fprintf(f2, "%.4f+(%.4f)j ", Ybcr[i][j], Ybci[i][j]);
}
fprintf(f2, "\n\n\n\t\t***** Yca *****\n");
for(i=0;i<no_setc;i++)
{
fprintf(f2, "\n");
for(j=0;j<no_seta;j++)
fprintf(f2, "%.4f+(%.4f)j ", Yacr[j][i], Yaci[j][i]);
}
fprintf(f2, "\n\n\n\t\t***** Ycb *****\n");
for(i=0;i<no_setc;i++)
{
fprintf(f2, "\n");
for(j=0;j<no_setb;j++)
fprintf(f2, "%.4f+(%.4f)j ", Ybcr[j][i], Ybci[j][i]);
}
fprintf(f2, "\n\n\n\t\t***** Ycc *****\n");
for(i=0;i<no_setc;i++)
{
fprintf(f2, "\n");
for(j=0;j<no_setc;j++)
fprintf(f2, "%.4f+(%.4f)j ", Yccr[i][j], Ycci[i][j]);
}
// ***** CALCULATION AND PRINTING OF Y-KRON *****
complex_inverse(Yccr, Ycci, Y1r, Y1i, no_setc);
complex_mul(Yacr, Yaci, no_seta, Y1r, Y1i, no_setc, Y3r, Y3i, no_setc);
for(i=0;i<no_setc;i++)
for(j=0;j<no_setc;j++)
{
Y1r[i][j]=Yacr[j][i];
Y1i[i][j]=Yaci[j][i];
}
complex_mul(Y3r, Y3i, no_setc, Y1r, Y1i, no_setc, Y2r, Y2i, no_setc);
complex_subtraction(Yaar, Yaai, Y2r, Y2i, YAAr, YAAi, no_setc, no_setc);
for(i=0;i<no_setc;i++)
for(j=0;j<no_setc;j++)
{
Ykronr[i][j]=YAAr[i][j];
Ykroni[i][j]=YAAi[i][j];
}
fprintf(f2, "\n\n\n\t\t***** YAA *****\n");

```

```

for(i=0;i<no_seta;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_seta;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",YAAr[i][j],YAAi[i][j]);
}
complex_inverse(Yccr,Ycci,Y1r,Y1i,no_setc);
complex_mul(Yacr,Yaci,no_seta,Y1r,Y1i,no_setc,Y3r,Y3i,no_setc);
for(i=0;i<no_setc;i++)
for(j=0;j<no_setb;j++)
{
Y1r[i][j]=Ybcr[j][i];
Y1i[i][j]=Ybci[j][i];
}
complex_mul(Y3r,Y3i,no_seta,Y1r,Y1i,no_setb,Y2r,Y2i,no_setc);
complex_subtraction(Yabr,Yabi,Y2r,Y2i,YAAr,YAAi,no_seta,no_setc);
for(i=0;i<no_setb;i++)
for(j=0;j<no_seta;j++)
{
Ykronr[i][j+no_seta]=YAAr[i][j];
Ykroni[i][j+no_seta]=YAAi[i][j];
}
fprintf(f2,"\n\n\n\t\t***** YAB *****\n");
for(i=0;i<no_seta;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_setb;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",YAAr[i][j],YAAi[i][j]);
}
complex_inverse(Yccr,Ycci,Y1r,Y1i,no_setc);
complex_mul(Ybcr,Ybci,no_setb,Y1r,Y1i,no_setc,Y3r,Y3i,no_setc);
for(i=0;i<no_setc;i++)
for(j=0;j<no_seta;j++)
{
Y1r[i][j]=Yacr[j][i];
Y1i[i][j]=Yaci[j][i];
}
complex_mul(Y3r,Y3i,no_setb,Y1r,Y1i,no_seta,Y2r,Y2i,no_setc);
for(i=0;i<no_setb;i++)
for(j=0;j<no_seta;j++)
{
Y1r[i][j]=Yabr[j][i];
Y1i[i][j]=Yabi[j][i];
}
complex_subtraction(Y1r,Y1i,Y2r,Y2i,YAAr,YAAi,no_setb,no_seta);
for(i=0;i<no_setb;i++)

```

```

for(j=0;j<no_setb;j++)
{
Ykronr[i+no_setb][j]=YAAr[i][j];
Ykroni[i+no_setb][j]=YAAi[i][j];
}
fprintf(f2, "\n\n\n\t\t***** YBA *****\n");
for(i=0;i<no_setb;i++)
{
fprintf(f2, "\n");
for(j=0;j<no_setb;j++)
fprintf(f2, "%.4f+(%.4f)j ", YAAr[i][j], YAAi[i][j]);
}
complex_inverse(Yccr, Ycci, Y1r, Y1i, no_setc);
complex_mul(Ybcr, Ybci, no_setb, Y1r, Y1i, no_setc, Y3r, Y3i, no_setc);
for(i=0;i<no_setc;i++)
for(j=0;j<no_setb;j++)
{
Y1r[i][j]=Ybcr[j][i];
Y1i[i][j]=Ybci[j][i];
}
complex_mul(Y3r, Y3i, no_setb, Y1r, Y1i, no_setb, Y2r, Y2i, no_setc);
complex_subtraction(Ybbr, Ybbi, Y2r, Y2i, YAAr, YAAi, no_setb, no_setb);
for(i=0;i<no_setb;i++)
for(j=0;j<no_setb;j++)
{
Ykronr[i+no_setb][j+no_setb]=YAAr[i][j];
Ykroni[i+no_setb][j+no_setb]=YAAi[i][j];
}
fprintf(f2, "\n\n\n\t\t***** YBB *****\n");
for(i=0;i<no_setb;i++)
{
fprintf(f2, "\n");
for(j=0;j<no_setb;j++)
fprintf(f2, "%.4f+(%.4f)j ", YAAr[i][j], YAAi[i][j]);
}
fprintf(f2, "\n\n\n\t\t***** Y Kron *****\n");
for(i=0;i<no_setb+no_setb;i++)
{
fprintf(f2, "\n");
for(j=0;j<no_setb+no_setb;j++)
fprintf(f2, "%.4f+(%.4f)j ", Ykronr[i][j], Ykroni[i][j]);
}
// ***** CALCULATION AND PRINTING OF Z-KRON *****
complex_inverse(Ykronr, Ykroni, Zkronr, Zkroni, (no_setb+no_setb));
fprintf(f2, "\n\n\n\t\t***** Z Kron *****\n");
for(i=0;i<no_setb+no_setb;i++)

```

```

{
fprintf(f2,"\n");
for(j=0;j<no_setb;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",Zkronr[i][j],Zkroni[i][j]);
}
for(i=0;i<no_setb;i++)
for(j=0;j<no_setb;j++)
{
Y1r[i][j]=Zkronr[i][j];
Y1i[i][j]=Zkroni[i][j];
}
for(i=0;i<no_setb;i++)
for(j=0;j<no_setb;j++)
{
Y2r[i][j]=Zkronr[i][j+no_setb];
Y2i[i][j]=Zkroni[i][j+no_setb];
}
complex_subtraction(Y1r,Y1i,Y2r,Y2i,Y3r,Y3i,no_setb,no_setb);
for(i=0;i<no_setb;i++)
for(j=0;j<no_setb;j++)
{
Y1r[i][j]=Zkronr[i+no_setb][j];
Y1i[i][j]=Zkroni[i+no_setb][j];
}
complex_subtraction(Y3r,Y3i,Y1r,Y1i,Y2r,Y2i,no_setb,no_setb);
for(i=0;i<no_setb;i++)
for(j=0;j<no_setb;j++)
{
Y3r[i][j]=Zkronr[i+no_setb][j+no_setb];
Y3i[i][j]=Zkroni[i+no_setb][j+no_setb];
}
complex_addition(Y2r,Y2i,Y3r,Y3i,Zredr,Zredi,no_setb,no_setb);
fprintf(f2,"\n\n\n\t\t***** Z Reduced *****\n");
for(i=0;i<no_setb;i++)
{
fprintf(f2,"\n");
for(j=0;j<no_setb;j++)
fprintf(f2,"%0.4f+(%0.4f)j ",Zredr[i][j],Zredi[i][j]);
}
complex_inverse(Zredr,Zredi,Y2r,Y2i,no_setb);
//
for(i=0;i<no_branch;i++)
{
Ybbr[i][0]=0;
Ybbr[i][1]=0;
Ybbr[i][2]=0;
}

```

```

Ybbr[i][3]=0;
Ybbr[i][4]=0;
Ybbr[i][5]=0;
}
for(j=0;j<no_buses+no_setaj++)
{
Yccr[j][0]=node_data[j][1];
Yccr[j][1]=node_data[j][2];
Yccr[j][2]=node_data[j][1];
Yccr[j][3]=node_data[j][2];
}
kkk=1;
do
{
***** CALCULATION OF POWER FLOWS IN BACKWARD SWEEP *****
fprintf(f2, "\n\n\t ***** RESULT AFTER ITERATION %d *****\n", kkk);
fprintf(f2, "\n\t ***** CALCULATION OF POWER FLOWS IN BACKWARD SWEEP *****\n");
fprintf(f2, "\nFROM TO P1\t Q1\t P\t Q\t P2\t Q2\n");
for(j=(no_layer-1);j>=0;j--)
{
for(i=1;i<=layer[j][0];i++)
{
check_for_connection(branch_data,no_branch,int(branch_data[layer[j][i]-1][2]),connection,&no_fconnection);
if(no_fconnection==0)
{
node_position(node_data,no_buses+no_setaj,
int(branch_data[layer[j][i]-1][2]),&bconnection1);
Ybbr[layer[j][i]-1][2]=node_data[bconnection1][5]-node_data[bconnection1][3]
+pow(Yccr[bconnection1][2],2)*branch_data[layer[j][i]-1][5];
Ybbr[layer[j][i]-1][3]=node_data[bconnection1][6]-node_data[bconnection1][4]
-pow(Yccr[bconnection1][2],2)*branch_data[layer[j][i]-1][6];
node_position(node_data,no_buses+no_setaj,
int(branch_data[layer[j][i]-1][1]),&bconnection2);
Ybbr[layer[j][i]-1][0]=Ybbr[layer[j][i]-1][2]+(pow(Ybbr[layer[j][i]-1][2],2)
+pow(Ybbr[layer[j][i]-1][3],2))*branch_data[layer[j][i]-1][3]
/pow(Yccr[bconnection1][2],2)
+pow(Yccr[bconnection2][2],2)*branch_data[layer[j][i]-1][5];
Ybbr[layer[j][i]-1][1]=Ybbr[layer[j][i]-1][3]+(pow(Ybbr[layer[j][i]-1][2],2)
+pow(Ybbr[layer[j][i]-1][3],2))*branch_data[layer[j][i]-1][4]
/pow(Yccr[bconnection1][2],2)
-pow(Yccr[bconnection2][2],2)*branch_data[layer[j][i]-1][6];
Ybbr[layer[j][i]-1][4]=Ybbr[layer[j][i]-1][0]
-pow(Yccr[bconnection2][2],2)*branch_data[layer[j][i]-1][5];
Ybbr[layer[j][i]-1][5]=Ybbr[layer[j][i]-1][1]

```

```

+pow(Yccr[bconnection2][2],2)*branch_data[layer[j]][i-1][6];
fprintf(f2,"%d\t%d\t%.4f\t%.4f\t%.4f\t%.4f\t%.4f\t%.4f\n",
int(branch_data[layer[j]][i-1][1]),int(branch_data[layer[j]][i-1][2]),
Ybbr[layer[j]][i-1][2],Ybbr[layer[j]][i-1][3],
Ybbr[layer[j]][i-1][0],Ybbr[layer[j]][i-1][1],
Ybbr[layer[j]][i-1][4],Ybbr[layer[j]][i-1][5]);
}
else
{
P_flow=0.00;
Q_flow=0.00;
for(k=1;k<=no_fconnection;k++)
for(kk=0;kk<no_branch;kk++)
if((int(branch_data[layer[j]][i-1][2])==int(branch_data[kk][1]))
&&(connection[k]==int(branch_data[kk][2])))
{
P_flow+=Ybbr[kk][0];
Q_flow+=Ybbr[kk][1];
}
node_position(node_data,no_buses+no_seta,
int(branch_data[layer[j]][i-1][2]),&bconnection1);
Ybbr[layer[j]][i-1][2]=P_flow+node_data[bconnection1][5]
-node_data[bconnection1][3]
+pow(Yccr[bconnection1][2],2)*branch_data[layer[j]][i-1][5];
Ybbr[layer[j]][i-1][3]=Q_flow+node_data[bconnection1][6]
-node_data[bconnection1][4]
-pow(Yccr[bconnection1][2],2)*branch_data[layer[j]][i-1][6];
node_position(node_data,no_buses+no_seta,
int(branch_data[layer[j]][i-1][1]),&bconnection2);
Ybbr[layer[j]][i-1][0]=Ybbr[layer[j]][i-1][2]
+(pow(Ybbr[layer[j]][i-1][2],2)
+pow(Ybbr[layer[j]][i-1][3],2))*branch_data[layer[j]][i-1][3]
/pow(Yccr[bconnection1][2],2)
+pow(Yccr[bconnection2][2],2)*branch_data[layer[j]][i-1][5];
Ybbr[layer[j]][i-1][1]=Ybbr[layer[j]][i-1][3]
+(pow(Ybbr[layer[j]][i-1][2],2)
+pow(Ybbr[layer[j]][i-1][3],2))*branch_data[layer[j]][i-1][4]
/pow(Yccr[bconnection1][2],2)
-pow(Yccr[bconnection2][2],2)*branch_data[layer[j]][i-1][6];
Ybbr[layer[j]][i-1][4]=Ybbr[layer[j]][i-1][0]
-pow(Yccr[bconnection2][2],2)*branch_data[layer[j]][i-1][5];
Ybbr[layer[j]][i-1][5]=Ybbr[layer[j]][i-1][1]
+pow(Yccr[bconnection2][2],2)*branch_data[layer[j]][i-1][6];
fprintf(f2,"%d\t%d\t%.4f\t%.4f\t%.4f\t%.4f\t%.4f\t%.4f\n",
int(branch_data[layer[j]][i-1][1]),int(branch_data[layer[j]][i-1][2]),
Ybbr[layer[j]][i-1][2],Ybbr[layer[j]][i-1][3],Ybbr[layer[j]][i-1][0],

```

```

Ybbr[layer[j][i]-1][1],Ybbr[layer[j][i]-1][4],Ybbr[layer[j][i]-1][5]);
}
}
}
//***** CALCULATION OF NODE-VOLTAGES IN FORWARD-SWEEP *****
max=0.00;
fprintf(f2,"\n\t***** CALCULATION OF NODE-VOLTAGES IN FORWARD-
SWEEP *****\n");
fprintf(f2,"\n\tNODE NO\t MAGNITUDE\t\t ANGLE\n");
for(j=0;j<no_layer;j++)
{
for(i=1;i<=layer[j][0];i++)
{
node_position(node_data,no_buses+no_seta,
int(branch_data[layer[j][i]-1][1]),&bconnection1);
node_position(node_data,no_buses+no_seta,
int(branch_data[layer[j][i]-1][2]),&bconnection2);
Yccr[bconnection2][2]=sqrt((pow(Yccr[bconnection1][2],2))
-2*((Ybbr[layer[j][i]-1][4]*branch_data[layer[j][i]-1][3])
+(Ybbr[layer[j][i]-1][5]*branch_data[layer[j][i]-1][4]))
+((pow(Ybbr[layer[j][i]-1][4],2)
+pow(Ybbr[layer[j][i]-1][5],2))*(pow(branch_data[layer[j][i]-1][3],2)
+pow(branch_data[layer[j][i]-1][4],2))/(pow(Yccr[bconnection1][2],2))));
Yccr[bconnection2][3]=Yccr[bconnection1][3]
-atan2((Ybbr[layer[j][i]-1][4]*branch_data[layer[j][i]-1][4]
-Ybbr[layer[j][i]-1][5]*branch_data[layer[j][i]-1][3])/Yccr[bconnection1][2],
Yccr[bconnection1][2]-(Ybbr[layer[j][i]-1][4]*branch_data[layer[j][i]-1][3]
+Ybbr[layer[j][i]-1][5]*branch_data[layer[j][i]-1][4])/Yccr[bconnection1][2]);
diff_V[bconnection2]=fabs(Yccr[bconnection2][0]-Yccr[bconnection2][2]);
Yccr[bconnection2][0]=Yccr[bconnection2][2];
Yccr[bconnection2][1]=Yccr[bconnection2][3];
if(diff_V[bconnection2]>max)
max=diff_V[bconnection2];
fprintf(f2,"\n\t%d\t %f\t\t %f",int(branch_data[layer[j][i]-1][2]),
Yccr[bconnection2][2],180*Yccr[bconnection2][3]/pi);
}
}
kkk++;
for(i=0;i<no_seta;i++)
{
node_position(node_data,no_buses+no_seta,seta[i],&bconnection1);
node_position(node_data,no_buses+no_seta,setb[i],&bconnection2);
Yaar[i][0]=node_data[bconnection1][1]*cos(node_data[bconnection1][2])
-Yccr[bconnection1][2]*cos(Yccr[bconnection1][3])
-node_data[bconnection2][1]*cos(node_data[bconnection2][2])
+Yccr[bconnection2][2]*cos(Yccr[bconnection2][3]);
}
}
}

```

```

Yaar[i][1]=node_data[bconnection1][1]*sin(node_data[bconnection1][2])
-Yccr[bconnection1][2]*sin(Yccr[bconnection1][3])
-node_data[bconnection2][1]*sin(node_data[bconnection2][2])
+Yccr[bconnection2][2]*sin(Yccr[bconnection2][3]);
}
for(i=0;i<no_set_a;i++)
{
node_position(node_data,no_buses+no_set_a,seta[i],&bconnection1);
node_position(node_data,no_buses+no_set_a,setb[i],&bconnection2);
Yaar[i][2]=0;
Yaar[i][3]=0;
for(j=0;j<no_set_a;j++)
{
Yaar[i][2]+=(Y2r[i][j]*Yaar[j][0]-Y2i[i][j]*Yaar[j][1]);
Yaar[i][3]+=(Y2r[i][j]*Yaar[j][1]+Y2i[i][j]*Yaar[j][0]);
}
Yaar[i][3]=-Yaar[i][3];
Yaar[i][4]=Yaar[i][2]*Yccr[bconnection1][2]*cos(Yccr[bconnection1][3])-
Yaar[i][3]*Yccr[bconnection1][2]*sin(Yccr[bconnection1][3]);
Yaar[i][5]=Yaar[i][2]*Yccr[bconnection1][2]*sin(Yccr[bconnection1][3])+
Yaar[i][3]*Yccr[bconnection1][2]*cos(Yccr[bconnection1][3]);
// ***** POWER INJECTION TO SET-A AND SET-B BUSES *****
node_data[bconnection1][3]+=Yaar[i][4];
node_data[bconnection1][4]+=Yaar[i][5];
node_data[bconnection2][3]-=Yaar[i][4];
node_data[bconnection2][4]-=Yaar[i][5];
}
}
while(max>tolerance);
}

```