EFFICIENT LOAD FLOW FOR LARGE WEAKLY MESHED NETWORKS

G.X. Luo* A. Semlyen

Department of Electrical Engineering
University of Toronto
Toronto, Ontario, Canada M5S 1A4

*At present with Ontario Hydro

Abstract -- A new and efficient method for calculating the load flow solution of weakly meshed transmission and distribution systems is presented. Its essential advantages over a previous approach[1] are the following: (1) It uses active and reactive powers as flow variables rather than complex currents, thus simplifying the treatment of P,V buses and reducing the related computational effort to half; (2) It uses an efficient tree labeling technique which also contributes to the computational efficiency of the procedure; (3) It uses an improved solution strategy, thereby reducing the burden of mismatch calculations which is an important component of the solution process. Results of tests with 30, 243, 1380, and 4130 bus systems are given to illustrate the performance of the proposed method.

Keywords: Load flow, Radial network, Weakly meshed network, Tree labeling.

INTRODUCTION

Several efficient and generally reliable load flow solution techniques have been developed over the last few decades. These include the Gauss-Seidel load flow[3], the Newton-Raphson load flow[4] and the Fast Decoupled Load Flow[5]. Although these classical techniques have been widely used, there are situations when they may experience difficulties or become inefficient, as in the case of

(1) Ill-conditioned or poorly initialized networks[6,7,8]

(2) Special applications[9] or special network structure, e.g. weakly meshed networks[1].

This paper presents a fast and efficient method for obtaining load flow solutions of weakly meshed power systems or distribution systems. It represents a significant improvement over our first approach reported in [1] due to three new features. First, by using powers (P,Q) as variables in the solution process instead of complex currents, it handles the P,V buses in a direct manner as simple loop breakpoints, thereby reducing the related computational effort to half. Second, by applying the tree labeling technique of network flow programming[2] to labeling the radial network (unlike for instance in reference [10] where network flow programming is the underlying method of the load flow calculation), the sensitivity matrix (equivalent to the breakpoint impedance matrix of reference [1]) can be constructed using the network graph which minimizes the effort. Third, based on the finding that in each iteration the CPU time required for obtaining the breakpoint voltage mismatches is dominant, savings are achieved by using single sweeps (instead of converged sets) to calculate the mismatches.

89 SM 677-6 PWRS A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1989 Summer Meeting, Long Beach, California, July 9 - 14, 1989. Manuscript submitted January 10, 1989; made available for printing June 9, 1989.

The proposed method has been programmed and tested in systems of different sizes in order to demonstrate its performance.

To put the problem of weakly meshed networks and their solution method into perspective, we note that they can be viewed as the dual of nodal approaches, as for instance the Fast Decoupled Load Flow. The latter deals with the solution of a general problem with arbitrarily many loops and uses a nodal approach for the solution. Correspondingly, the unknowns are nodal voltages and the mismatches are nodal powers which, incidentally, can be calculated analytically. In the case of weakly meshed networks, the dual approach based on mesh equations represents a viable alternative. Then the unknowns become mesh currents or powers and the mismatches are loop breakpoint voltages. The latter, however, can not be obtained analytically, so that additional calculations (sweeps) are needed in the radial network obtained by breaking up the loops. Clearly, the mesh based approach is specifically tailored for systems where the number of meshes is not too large.

PREVIOUS WORK

To make this paper self-contained, the basic idea of our previous method[1] is briefly reviewed in this section. Note that all variables are complex quantities.

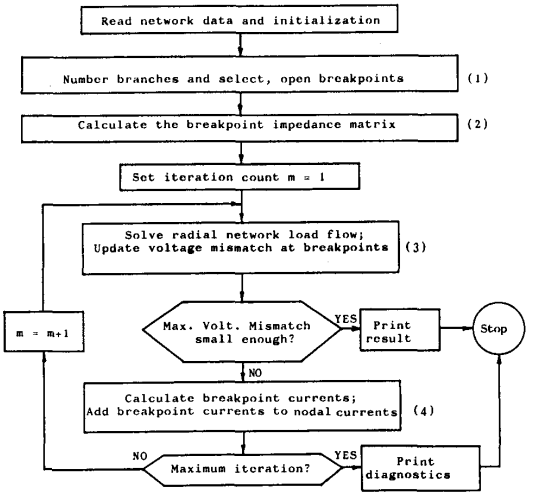


Fig.1 Computational flow chart of reference [1]

0885-8950/90/1100-1309$01.00 © 1990 IEEE

Fig.1 shows the algorithm for weakly meshed networks of reference [1]. The process starts with converting the meshed network into a radial network by means of selecting and opening a number of breakpoints (Block 1). The second step is to calculate the breakpoint impedance matrix based on the multi-port compensation principle[10]. A complete backward-forward sweep must be performed to obtain each column of the matrix. This implies that the same mathematical operation must be performed for every node in the radial network. Blocks 3 and 4 represent the main steps in the iterative process. In Block 3 there is an inner iterative process in which a converged radial network load flow solution is obtained by repeated backward-forward sweeps. To obtain the breakpoint currents, a set of linear equations in the complex domain is solved in Block 4.

Since the P,V buses were treated in a different way than the loop breakpoints, it was shown in reference [1] that the efficiency of the solution method for weakly meshed networks is adversely affected by the introduction of the P,V nodes.

DESCRIPTION OF THE PROPOSED METHOD

General Procedure

In order to efficiently obtain the load flow solution of a weakly meshed network, the network is first converted to a radial network with the slack bus as its root. This conversion is done by breaking all the loops in the meshed network. To compensate for the breaking, proper power injections (active and reactive) must be assigned to both sides of a breakpoint to reflect the power circulating in the original loop. The breakpoint power injections are proper if the voltages at the two sides of the breakpoints in the radial network are equal (both in magnitude and in phase). The solution procedure will start with an initial guess of these power injections. In the process, they will be corrected iteratively until they converge to their final values.

Each iteration of the procedure contains two stages: Mismatch Calculation (Stage 1) and Injection Correction (Stage 2). At Stage 1, breakpoint voltage mismatches (magnitude and angle) are calculated in the radial network with the assigned breakpoint power injections. This is done by a Backward-Forward Sweep along the radial network. At Stage 2, corrections of the breakpoint power injections are calculated by means of a sensitivity matrix, with the breakpoint voltage mismatches obtained at Stage 1 as inputs.

To obtain the load flow solution of the meshed network, the above iteration is repeated until the breakpoint voltage mismatches and/or the corrections of the breakpoint power injections are within the preset error tolerance.

When P,V buses are encountered, each P,V bus is handled similarly to a loop breakpoint, but the computational effort is half. Details of how to treat a P,V bus as a particular type of breakpoint will be discussed later.

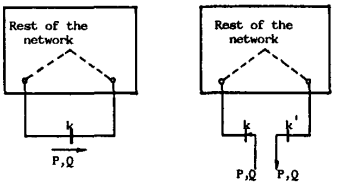
Classification of the Breakpoints

As mentioned before, there are two types of breakpoints in the network: Loop Breakpoint (LBP) and P,V Bus Breakpoint (PVBP).

Loop Breakpoint (LBP)

A LPB breaks a loop in the network. The location of a LBP is normally chosen at a bus so that it breaks the bus into two: the original bus and an artificial bus. These two buses form the two sides of the LBP as shown in Fig.2.

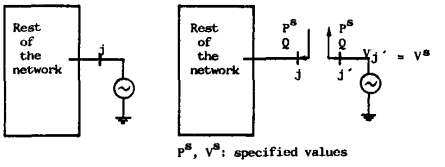
We note that the LBP in this paper is the same as in reference [1], except that the associated variables are different. In the example of Figure 2, the breakpoint voltage mismatches are ΔV ($=V_k - V_k'$) and $\Delta\delta$ ($=\delta_k - \delta_k'$). The power injections are P and Q. The corrections of LBP power injections are not shown in the figure.



(a) Before the loop is broken (b) After the loop is broken
Fig.2 Explanation of a LBP

P,V Bus Breakpoint (PVBP)

Since at a P,V bus P and V are specified, it can be treated as a particular type of breakpoint. This can be seen in Fig.3 where an artificial bus j' serves as the other side of the PVBP. This artificial bus has a fixed voltage magnitude which is equal to the predefined value of the P,V bus voltage. Since the active power P of a P,V bus is specified, the PVBP active power injection P is known from the beginning and the correction of active power injection is always zero. Therefore, the only unknown associated with a PVBP is the reactive power injection Q and the effort of handling a PVBP is correspondingly half of that for a LBP.



(a) Before the P,V bus is broken b) After the P,V bus is broken
Fig.3 P,V bus breakpoint

With this procedure, the mismatch of a PVBP is only the breakpoint voltage magnitude difference ΔV ($=V_j - V_{PV}$) and the correction ΔQ of the reactive power injection is the only quantity sought at each iteration.

Network Flow Tree Labeling

Modified Network Flow Programming tree labeling[2] has been applied to label the radial network. This makes possible the construction of the sensitivity matrix in a very efficient way, as explained later. The labeling includes several functions which are node oriented: an order function $o(\cdot)$, a predecessor function $p(\cdot)$, a distance function $d(\cdot)$, and an arc function $a(\cdot)$. Below are their definitions.

Order $o(\cdot)$:

A sequence pointer puts the nodes in the tree in order. It goes through each node only once in a top to bottom, left to right sequence, starting from the root node.

Precedessor $p(\cdot)$:

Upward pointer. A pointer list which contains for each node w (other than the root) the unique node (say v) above node w which is connected by an arc (i.e. a branch), e.g. $p(w)=v$. By definition $p(\text{root}) = 0$.

Distance $d(\cdot)$:

This function indicates the number of arcs in the predecessor path of the node to the root, not containing the root arc itself.

Arc $a(\cdot)$:

A one to one node-arc correspondence function. This function contains the arc index which connects the node to its predecessor. For the root we define an additional arc so that a(root) = n_a+1, where n_a is the total number of arcs in the tree.

As an example, the tree labeling functions for the tree illustrated in Fig.4 have the following values:

Index	$o(\cdot)$	$p(\cdot)$	$d(\cdot)$	$a(\cdot)$
1	8	0	0	16
2	9	8	1	1
3	1	9	2	3
4	4	1	3	6
5	10	4	4	10
6	6	4	4	9
7	5	6	5	12
8	11	5	6	14
9	15	5	6	13
10	12	9	2	2
11	7	12	3	5
12	13	12	3	4
13	3	13	4	8
14	14	13	4	7
15	2	14	2	11

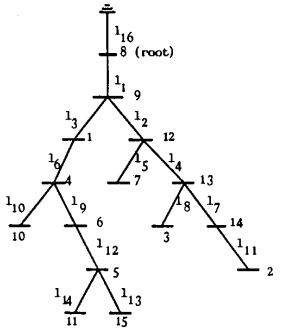


Fig.4 A sample tree

Meshed-to-Radial Network Conversion with Tree Labeling

Breakpoints are first selected for the conversion of a meshed network to a radial network. As noted in reference [1], breakpoint selection does not affect the convergence performance of the solution method in any noticeable manner. Therefore, the algorithm for selection of breakpoints described in [1] could have been adopted with appropriate modifications to fit the tree labeling and the related algorithms described in this paper. To avoid modifications, another algorithm was chosen to do the conversion. Although this algorithm is very similar to that described in [1], it combines the conversion and the Network Flow Tree Labeling into a single process. This has been summarized as Algorithm A given in the Appendix.

Sensitivity Matrix

The Sensitivity Matrix (SM) is defined as the matrix M in the equation

$$M\Delta S = \Delta V \tag{1}$$

where ΔV is the breakpoint voltage mismatch vector (magnitudes and angles) and ΔS is the breakpoint power injection correction vector (real and reactive powers). This matrix gives a measure of how the changes in ΔV depend on ΔS assuming that all nonlinearities are neglected. Based on the compensation theorem[10,11], the Sensitivity Matrix used in (1) can be the breakpoint impedance matrix described in [1] based on the following assumptions:

(1) The loads and other shunt elements are not represented, and

(2) All bus voltages are assumed close to 1.0 p.u. and their phase angles close to zero.

Since the breakpoint impedance matrix has been proven effective as a sensitivity matrix[1], it is also used in the present approach. However, the appearance of the SM employed now is different since we are using real variables (P,Q,V,δ) rather than complex ones $(\bar{V},\bar{I})$ and breakpoints have been generalized to include P,V buses.

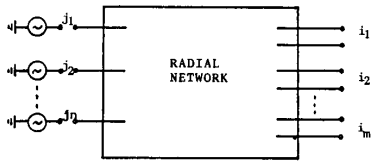


Fig.5 A network with breakpoints

Suppose a network has m LBPs and n PVBPs as shown in Fig.5. When LBPs and PVBPs are not distinguished, the Thevenin equivalent impedance matrix, as seen from the breakpoints, is defined by:

$$\bar{Z}\,\bar{I} = \bar{V} \tag{2}$$

where $\bar{V}$ and $\bar{I}$ are the breakpoint voltage vector and the breakpoint current vector, respectively. Since the radial network is linear, eqn.(2) applies to the incremental quantities as well, so that

$$\bar{Z}\Delta\bar{I} = \Delta\bar{V}\ . \tag{3}$$

With the assumption of all bus voltages being close to 1.0 p.u. and the phase angles small, the following holds

$$\Delta\bar{I} \cong \Delta\bar{S}^* \tag{4}$$

so that

$$\bar{Z}\Delta\bar{S}^* = \Delta\bar{V}\ . \tag{5}$$

By expressing the complex quantities in eqn.(5) in their rectangular form

$$\bar{Z} = R + jX \tag{6a}$$

$$\Delta\bar{S}^* = \Delta P - j\Delta Q \tag{6b}$$

and

$$\Delta\bar{V} = \Delta V + j\Delta\delta \tag{6c}$$

eqn.(5) can be rearranged and expressed as

$$R\Delta P + X\Delta Q = \Delta V \tag{7a}$$

$$X\Delta P - R\Delta Q = \Delta\delta \tag{7b}$$

or in a more compact form

$$\begin{bmatrix} X & R \\ -R & X \end{bmatrix}\begin{bmatrix} \Delta Q \\ \Delta P \end{bmatrix} = \begin{bmatrix} \Delta V \\ \Delta\delta \end{bmatrix}\ . \tag{8}$$

which corresponds to eqn.(1).

Note that the dimension of the vectors ΔP and $\Delta\delta$ is only m since the $\Delta\delta$ for each PVPB is always zero. As a result of this, the dimension of eqn.(8) is equal to $2m+n$.

To illustrate how a sensitivity matrix can be constructed, a simple example is given below. Remember that the diagonal elements in matrix X are the self reactances of the breakpoints and the off-diagonal elements are the mutual reactances of two breakpoints. A self reactance can be obtained by adding the line reactances of the lines which form the loop connecting the two buses of a breakpoint, while a mutual reactance is the sum of the line reactances which are common for two breakpoint loops. The same rule applies to matrix R except that the word 'reactance' should be replaced by 'resistance'.

Let us take the network shown in Fig.6 as an example. This network contains two LBPs (1-1' and 2-2') and one PVBP (3-3'). The line impedances are marked on the figure.

The X and R matrices for this example are

$$X = \begin{bmatrix} 1.4 & -1.0 & 0.4 \\ -1.0 & 1.3 & -0.4 \\ 0.4 & -0.4 & 0.7 \end{bmatrix} \tag{9a}$$

$$R = \begin{bmatrix} 0.7 & -0.5 & 0.2 \\ -0.5 & 0.65 & -0.2 \\ 0.2 & -0.2 & 0.35 \end{bmatrix} \tag{9b}$$

and the Sensitivity Matrix and eqn.(8) become

$$\begin{bmatrix} 1.4 & -1.0 & 0.4 & 0.7 & -0.5 \\ -1.0 & 1.3 & -0.4 & -0.5 & 0.65 \\ 0.4 & -0.4 & 0.7 & 0.2 & -0.2 \\ -0.7 & 0.5 & -0.2 & 1.4 & -1.0 \\ 0.5 & -0.65 & 0.2 & -1.0 & 1.3 \end{bmatrix} \begin{bmatrix} \Delta Q_1 \\ \Delta Q_2 \\ \Delta Q_3 \\ \Delta P_1 \\ \Delta P_2 \end{bmatrix} = \begin{bmatrix} \Delta V_1 \\ \Delta V_2 \\ \Delta V_3 \\ \Delta\delta_1 \\ \Delta\delta_2 \end{bmatrix} \tag{10}$$

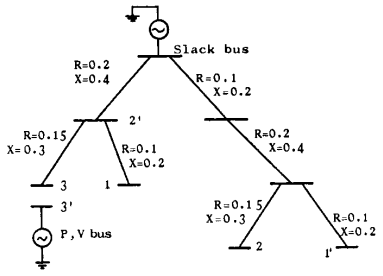


Fig.6 Sample system for illustrating the construction of the sensitivity matrix

Note that the heads (number without prime in Fig.6) and the tails (number with prime in Fig.6) of the breakpoints can be assigned randomly. This assignment will affect the sign of the off-diagonal elements in the X and R matrices of eqns.(9).

Construction of Sensitivity Matrix with Tree Labeling

An algorithm for constructing the elements of SM related to one breakpoint is presented in this section. A complete SM can be formed by repeated calls of the algorithm.

The algorithm consists of three major steps. Step 1 finds the unique path connecting the two side nodes of the breakpoint, which has been divided into a positive path and a negative path in the algorithm. Step 2 first establishes the nodal voltages for the nodes on the path and then the remaining nodal voltages. The elements of SM are then calculated in Step 3.

The two side nodes of a loop breakpoint are labeled by k and k' in the tree, as shown in Fig.7. The details of the corresponding Algorithm B are given in the Appendix.

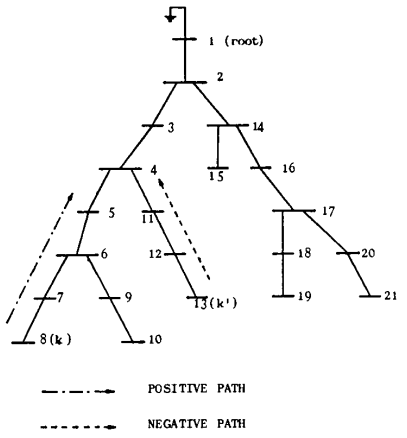


Fig.7 Sample tree for illustration of Algorithm B

Note that the whole algorithm is a graph based approach. For the breakpoint k shown in Fig.7, Step 1 establishes the positive path (8,7,6,5) and the negative path (13,12,11). Step 2 establish the voltages for the nodes on the paths by adding (for positive path) and subtracting (for negative path) the line parameters. Step 3 sets V_9 and V_{10} equal to V_6 as well as δ_9 and δ_{10} equal to δ_6. Note that in this step only a small subset of nodes (node 9 and node 10 in this example) are involved. The voltages of the rest (e.g. nodes 1~4, 14~21) remain at their initialized value (=0). This is why the algorithm is so efficient.

Algorithm B is also applied to PVBP. In this case, the positive path is the path connecting the P,V node to the root and there is no negative path involved.

Calculation of the Breakpoint Voltage Mismatch

The breakpoint voltage mismatches are calculated from the nodal voltages of the radial network. The latter are obtained from a single backward-forward sweep. The backward sweep serves to sum the load powers and the power losses from the end nodes to the root. The forward sweep establishes the nodal voltages from the root to the end nodes based on the power flows obtained in the backward sweep.

To illustrate the mathematical operations involved for one node during the sweep, we have set up a one branch (node) example shown in Fig.8.

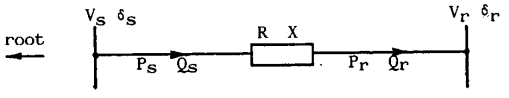


Fig.8 One branch example for explaining the sweeping

In Fig.8, subscripts s and r stand for "sending end" and "receiving end" respectively. P_r and Q_r are the sums of the loads, generations, powers absorbed by shunt components, powers drawn by the downstream branches, and power injections, if any.

In the backward sweep, the sending end power is calculated from

$$P_s = P_r + R\frac{P_r^2+Q_r^2}{V_r^2}$$
$$Q_s = Q_r + X\frac{P_r^2+Q_r^2}{V_r^2}. \quad (11)$$

In the forward sweep, with the sending end voltage known, the receiving end voltage is calculated from

$$\delta_r = \delta_s - \tan^{-1}\left[\frac{\Delta V''}{V_s - \Delta V'}\right]$$
$$V_r = \sqrt{(V_s-\Delta V')^2+\Delta V''^2} \quad (12)$$

where $\Delta V'$ and $\Delta V''$ are the longitudinal and transversal voltage drops respectively, obtained from

$$\Delta V' = \frac{RP_s+XQ_s}{V_s}$$
$$\Delta V'' = \frac{XP_s-RQ_s}{V_s} \quad (13)$$

Applying to each node and branch in the radial network the mathematical operations presented in the above example, the calculations of the breakpoint mismatches can be described as follows.

Backward–Forward Sweeps

(1) Backward Sweep (power summation):
Calculate the sending end powers (P_s, Q_s) in the reverse sequence of the order function. This step employs eqn.(11).

(2) Forward Sweep (voltage calculation):
Calculate the nodal voltage (V, δ) of each node following the sequence defined by the order function. Equations (12) and (13) are employed in this step.

Calculate the loop breakpoint mismatch by taking the voltage difference (magnitude and angle) between the two side nodes. For PVBP the only mismatch is the voltage magnitude difference between the P,V node and the specified value.

Calculation of Breakpoint Power Injection Corrections

Once the breakpoint voltage mismatches have been obtained, the breakpoint power injection corrections can be calculated by solving eqn.(1), presented earlier in this paper. To speed up the process, matrix M is factorized only once.

Computational Flow Chart

To summarize and to make clearer the procedure of the proposed method, a computational flow chart is given in Fig.9. Note that the block updating the breakpoint power injections in the flow chart does this for the $(k+1)$ th iteration by summing up the breakpoint power injections and the power injection corrections at the k th iteration.

NUMERICAL TESTS

The proposed method has been programmed and has been tested in four different systems with the R/X ratio varying from 0.1 to 5.0. Fig.10 shows the descriptions of the four systems.

Sparsity techniques for solving linear equations have been implemented. While this is of little consequence for small size systems (e.g. systems TS1 and TS2), 30% of CPU savings for matrix factorization and 20% of CPU savings for forward-backward substitutions have been achieved in system TS3. The sparsity structure of the sensitivity matrix for this system is given in Fig.11.

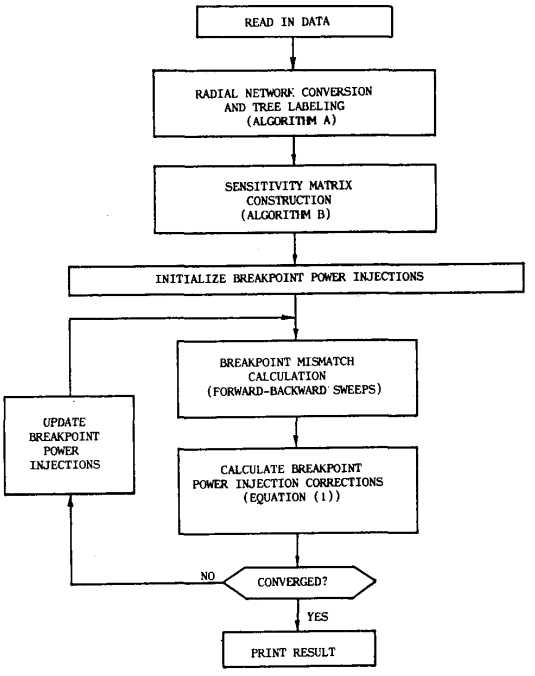


Fig.9 Computation flow chart of the proposed method

Name of system	Number of buses	Number of P,V buses	Number of loops
TS1	30	2	3
TS2	243	3	5
TS3	1380	3	35
TS4	4130	10	105

Fig.10 Description of four test systems

Fig.11 Sparsity structure of the sensitivity matrix of system TS3

The program has been run on a PC/AT compatible microcomputer (with a 287 coprocessor) and on a VAX Micro-2. With an error tolerance of 0.005 p.u., the average CPU requirement and the number of iterations needed are given in Fig.12. In the table, "Prep" means preparation phase which includes the processes of building the tree and the Sensitivity Matrix, and the SM factorization. "Sol" means the solution phase which covers all iterations. Total CPU includes the input/output process.

System	Run on PC			Run on MicroVax	No. of iterations required
	CPU for "Prep"	CPU for "Sol"	Total CPU	Total CPU	
TS1	1s	6s	11s	0.4s	14
TS2	17s	32s	1m 23s	5s	16
TS3	10m 3s	3m 52s	22m	≈ 1m	16
TS4	-	-	-	≈ 10m	40

Fig.12 CPU requirements of the tests

To verify that it is more efficient to use a single backward-forward sweep in the mismatch calculation, comparisons have been made for the two methods: the one using converged sweeps and the proposed method. Fig.13 shows the results of the comparisons. We see that about 60% of CPU savings in the solution phase has been achieved by the proposed method. We note, however, that in some cases using a single sweep may not ensure convergence of the program so that more sweeps may have to be used.

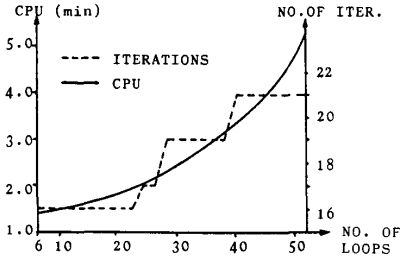


Fig.15 Performance of the proposed method as the number of loops is increased

As already emphasized, this method is designed for weakly meshed networks. We have tested the proposed method in System TS2 by increasing the number of the loops in order to see how the CPU time increases. In the test, starting with a case of 6 loops, the number of loops were increased, by 2 at each step, up to 52. The CPU time and the number of iterations for some steps have been listed in Fig.14. Plots showing the behavior of the proposed method as the number of loops increased are shown in Fig.15. It is clear that, as the number of loops increases, the method becomes less efficient, as the total CPU time increases rapidly.

CONCLUSIONS

This paper presents a fast and efficient method for load flow solutions of weakly meshed networks. The main improvements compared to the work reported in reference [1] are the following.

(1) Using real variables (P,Q) rather than complex ones enables us to handle the P,V buses in an exact manner as loop breakpoints. This not only eliminates the convergence difficulty due to the P,V buses[1], but also reduces to half the computational effort of handling the P,V buses.

(2) Applying Modified Network Tree Labeling makes it possible to construct the sensitivity matrix by a graph based approach. This approach reduces the related computational effort to a minimum.

(3) Based on the fact that the CPU required for backward-forward sweeps is dominant in each iteration of the solution phase, savings of CPU time have been achieved by using a single sweep instead of converged sweeping.

Numerous tests of the proposed method have been conducted. They have shown that the method is fast, efficient, robust, and eminently suitable for large scale systems with a small number of loops.

From the general structure of the proposed method it appears that, with appropriate modifications and extensions, it has the potential of being applied to three phase unbalanced load flow problems in weakly meshed networks.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada. The first author thanks Dr. P. Kundur for his support and encouragement. Useful discussions with Dr. D. Shirmohammadi are much appreciated.

Systems	CPU for sweep	CPU for correction	Converged sweeps (previous method)			Single sweep (proposed method)			CPU savings	
			Sweeps	Corrections	Total CPU	Sweeps	Corrections	Total CPU	Sec.	%
TS2	2s	≈ 0	39	9	78.0s	16	16	32.0s	46s	59
TS3	13.8s	0.7s	48	10	669.4s	16	16	232.0s	437s	65

Fig.13 Comparison of two methods different in sweeping strategy

No. of loops	Preparation phase CPU (s)				Solution phase CPU (s)			Total CPU	Total iter.
	Tree	SM	Factor. SM	Total	Sweep	Correction	Total		
6	9	9	≈ 0	18	2.5	≈ 0	39	1m 26s	16
10	10	8	1	19	2.5	0.5	41	1m 29s	16
20	10	11	7	28	2.5	1	52	1m 49s	16
30	11	13	21	45	3	1	78	2m 32s	19
40	11	19	50	80	3	2	110	3m 40s	21
50	13	21	94	128	3.5	3	137	4m 54s	21

Fig.14 CPU time and number of iterations for increasing number of loops

REFERENCES

[1] D. Shirmohammadi, H.W. Hong, A. Semlyen, and G.X. Luo, "A Compensation-Based Power Flow Method for Weakly Meshed Distribution and Transmission Networks", IEEE Trans. on Power Systems, Vol.3, No.2, May 1988, pp.753-762.

[2] J.L. Kennington and R.V. Helgason, *Algorithms for Network Programming*, John Wiley, 1981.

[3] W.D. Stevenson, *Elements of Power System Analysis*, McGraw-Hill , 1982.

[4] W.G. Tinney and C.E. Hart, "Power Flow Solutions by Newton's Method", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-86, 1967, pp. 1449-1457.

[5] B. Stott and O. Alsac, "Fast Decoupled Load Flow", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-93, No.3, May/June 1974, pp.859-869.

[6] S.C. Tripathy, G.D. Prasad, O.P. Malik, and G.S. Hope, "Load Flow Solutions for Ill-Conditioned Power Systems by a Newton-Like Method", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-101, No.10, October 1982, pp.3648-3657.

[7] S. Iwamoto and Y. Tamura, "A Load Flow Calculation Method for Ill-Conditioned Power Systems", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-100, No.4, April 1981, pp. 1736-1743.

[8] D. Rajicic and A. Bose, "A Modification to the Fast Decoupled Power Flow for Networks with High R/X Ratios", IEEE Trans. on Power Systems, Vol.3, No.2, May 1988, pp.743-746.

[9] K. Behnam-Guilani, "Fast Decoupled Load Flow: The Hybrid Model", IEEE Trans. on Power Systems, Vol.3, No.2, May 1988, pp.734-742.

[10] K. Takahashi, Y. Sekine, and T. Umezu, "Network-Flow Method Applied to Load-Flow Calculation", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-87, No.11, November 1968, pp.1939-1949.

[11] G. Gross and H.W.Hong, "A Two-Step Compensation Method for Solving Short Circuit Problems", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-101, No.6, June 1982, pp.1322-1331.

[12] O. Alsac, B. Stott, and W.F. Tinney, "Sparsity-Oriented Compensation Methods for Modified Solutions", IEEE Trans. on Power Apparatus and Systems, Vol. PAS-102, No.5, May 1983, pp.1050-1060.

APPENDIX

Algorithm A

This algorithm needs two counters: a node counter n (which counts the nodes in the tree) and the dangling node counter k (which counts how many nodes are dangling during the process). It also needs an auxiliary array w to store the dangling buses.

Algorithm A

(A1) Initializations: $n=1$, $k=1$, $n_a=n_b$ (n_b is the total number of buses in the network); $o(n)=$ slack bus number; $a(n)=n_l+1$ (n_l is the total number of branches in the network); $w(k)=n$; $d(n)=0$; $p(n)=0$.

(A2) $D=d(w(k))$; for all the branches and nodes having been used, raise a flag.

(A3) $b_d=o(w(k))$; examine all the remaining branches one by one until a branch (say branch b_c) connected to bus b_d is selected. If such a branch can not be found, then $k=k-1$, go to (A2).

(A4) Breakpoint detection: Check whether the other bus incident (say bus j) to the branch b_c has been used. If not, then $n=n+1$; $a(n)=b_c$; $o(n)=j$. Otherwise, an artificial bus numbered n_a+1 is created and $n_a=n_a+1$; $o(n)=n_a$.

(A5) $d(n)=D$; $p(n)=b_d$; $k=k+1$; $w(k)=o(n)$.

(A6) If $n=n_l+1$, stop; otherwise go to (A2).

Algorithm B

Assuming that the two side nodes of a loop breakpoint are labeled by k and k' in the tree, as shown in Fig.7, Algorithm B can be described as follows.

Algorithm B

(B1) Finding the path:

(1) Initialization: $i_1=1$, $i_2=1$, $u=k$, $v=k'$.

(2) Stop if $u=v$. Otherwise $t_1(i_1)=u$ and $t_2(i_2)=v$;

(3) $d_u=d(u)$, $d_v=d(v)$; $p_u=p(u)$, $p_v=p(v)$;
Stop if $p_u=p_v$. Otherwise

(4) If $d_u \geq d_v$, then $u=p_u$, $i_1=i_1+1$, $t_1(i_1)=u$;
if $d_u \leq dv$, then $v=p_v$, $i_2=i_2+1$, $t_2(i_2)=v$;
go to step (2).

(B2) Establishing the nodal voltages:

(1) Initialize all nodal voltages (magnitude and angle) to zero; set $V_1=0$, $V_2=0$, $\delta_1=0$, $\delta_2=0$.

(2) Do the following for i, starting with $i=i_1$ and decreasing i by 1 each time until $i<1$:
$u=t_1(i)$; $l=a(u)$; $V_1=V_1+X$; $\delta_1=\delta_1+R$;
$V(u)=V_1$; $\delta(u)=\delta_1$.

(3) Do the following for i, starting with $i=i_2$ and decreasing i by 1 each time until $i<1$:
$v=t_2(i)$; $l=a(v)$; $V_2=V_2-X$; $\delta_2=\delta_1-R$;
$V(v)=V_2$; $\delta(v)=\delta_2$.

(4) Do the following for i, starting with $i=1$ and increasing i by 1 each time until $i>i_1$:

(4a) $n=t_1(i)$, $d_u=d(u)$;

(4b) $t=u+1$, $d_t=d(t)$;

(4c) Increase i by 1 and restart from (4a), if $d_t \leq d_u$. Otherwise

(4d) $V(t)=V(u)$, $\delta(t)=\delta(u)$, $u=t$ and go to step (4b).

(5) Do the following for i, starting with $i=1$ and increasing i by 1 each time until $i>i_2$:

(5a) $v=t_2(i)$, $d_v=d(v)$;

(5b) $t=v+1$, $d_t=d(t)$;

(5c) Increase i by 1 and restart from (5a), if $d_t \leq d_v$. Otherwise

(5d) $V(t)=V(v)$, $\delta(t)=\delta(v)$, $v=t$ and go to step (5b).

(B3) Element calculations:

(1) Diagonal elements in Eqn. (9):
$X_k=V(u)-V(v)$
$\delta_k=\delta(u)-\delta(v)$.

(2) Off-diagonal elements in Eqn. (9):
X_{kj} = voltage magnitude difference between the two side nodes of breakpoint j,
δ_{kj} = angle difference between the two side nodes of breakpoint j.

Note that the arrays $t_1(\cdot)$ and $t_2(\cdot)$ in the algorithm are used to store the indices of the nodes on the positive and negative paths respectively.

Guang-Xiang Luo was born in Canton, China on August 22, 1946. He received his B.A.Sc. degree from Wuhan Institute of Hydraulic and Electric Engineering, China, in 1970. From 1970 to 1978, he was employed by Fong-Man Power Station, Northeast China, working at the station control center as a graduate engineer. Since 1980, he has been studying at University of Toronto, Canada, where he received his M.A.Sc. and Ph.D. degrees in electrical engineering in 1983 and 1988, respectively. Recently, he has joined the Ontario Hydro System Planning Division participating in an EPRI project on power system stability.

Adam Semlyen (F'87) was born and educated in Rumania where he obtained a Dipl. Ing. degree and his Ph.D. He started his career with an electric power utility and held an academic position at the Polytechnic Institute of Timisoara, Rumania. In 1969 he joined the University of Toronto where he is a professor in the Department of Electrical Engineering. His research interests include the Steady State and Dynamic Analysis of Power Systems, Electromagnetic Transients, and Power System Optimization.

Discussion

The paper is a considerable advance on the concepts outlined in the original paper [1]. The use of network flow techniques to implement a labelling method of building the sensitivity matrix demonstrates the power of network flow methods to minimize the processing required. Such methods have been used with great success when the problem can be formulated to take advantage of the special structure of electrical power systems and has been particularly successful in identifying optimum power flow configurations.

Distribution systems are typically multi-source. To avoid the representation of a significant proportion of the transmission system, it would be convenient to include all sources in the distribution network. Would the authors comment on whether the methods used in the paper can be adapted for more than one source, each with magnitude and phase angle reference?

The dual nature of the power flow solution to the popular decoupled method is interesting; the latter introduces approximations in the Jacobian to solve for the voltage and angle and has been found to fail to converge on some distribution networks with a wide range of R/X values. The paper states that the loop method does not suffer from similar convergence problems. Because of the loop solution technique, one would expect a better performance. Has the proximity of large and small impedance branches of differing R/X ratios been tried?

In weakly coupled networks the diagonal of the sensitivity matrix is dominant. It may be possible to use only the diagonal elements and employ relaxation techniques to converge on the true solution for the injection powers. This would avoid processing for the mutual loop impedances.

How would the methods described in the paper be adapted to deal with multi-voltage level systems?

G. Dromey, Dromey Design Inc. (non-member)

Reference

[1] D. Shirmohammadi, H. W. Hong, A. Semlyen and G. X. Lou. "A Compensation-Based Power Flow Method for Weakly Meshed Distribution and Transmission Networks". IEEE Trans. on Power Systems, Vol. 3, No. 2, May 1988, pp. 753-762.

The authors have presented a well-written paper addressing the need for alternative power flow methods for large weakly meshed networks. Application of graph-based algorithms and tree labeling methods opens up exciting possibilities for introducing advanced programming methods.

1. Although the break point selection may not affect the convergence performance of the algorithm, total number of break points seem to be non-unique and a larger number increases the computational demands. Is it possible to select the minimum number of loops consistent with the least computations?

2. The PV busses are assumed to have unlimited var supply. How can the algorithm accommodate var limits? What is the best starting value for vars at the PV busses?

3. Use of real variables P, Q, V and δ instead of complex currents have improved the previous algorithm. Real and reactive power losses and voltage magnitudes have been suggested as variables in Reference D1. Can the algorithm be modified to accommodate these 'real world' variables?

4. The example systems are given in terms of 'topological' specifications. It would be interesting to know the total P and Q demands of the various systems.

5. What is the sparsity algorithm used in the solution of the sensitivity equations? The special sparsity structure with diagonal and border bands (Fig. 11 of paper) offer scope for the application of modified techniques.

6. Is the numerical asymmetry of the sparse matrix taken into consideration in a special manner?

The authors are commended for an interesting paper.

Reference

[D1] A. Chandrasekaran and R. P. Broadwater, 'A New Formulation of Load Flow Equations in Balanced Radial Distribution Systems', Canadian Electrical Engineering Journal, Vol. 12, No. 4, 1987, pp. 147-151.

Manuscript received July 25, 1989.

G.X. Luo and A. Semlyen: We would like to thank both discussers for their remarks and questions which give us the opportunity to clarifying some of the problems raised in the paper. Our answers are as follows.

To Dr. A. Chandrasekaran:

1. The number of breakpoints is determined by the number of P,V buses and the topology of the network. We need one breakpoint for each P,V bus and one for each independent loop of the network. The number of the latter is

$$n_{loops}=n_{lines}-n_{nodes}+1 \qquad \text{(a)}$$

since the number of tree branches is $n_{nodes}-1$. It is convenient to select meshes for independent loops, whenever it is possible.

2. In order to account for the VAR limits of P,V buses, the algorithm has to be slightly modified. The modification will convert the P,V bus to a P,Q bus when its VAR limit is violated. The latter is handled similarly to the load buses.

3. We thank the discusser for having brought their pertinent paper (Ref.[D1]) to our attention.

4. The total loads in our test systems, including the real and reactive losses, are:

System	P (MW)	Q (MVAR)
TS1	100	65
TS2	33	32
TS3	20	15
TS4	20	15

5. and 6. Our program did not take advantage of the special sparsity structure of the sensitivity matrices, mentioned by the discusser. Even in large weakly meshed systems these matrices will not be large. We certainly agree with the possibility of further improvements which could be obtained by using more efficient numerical algorithms.

To Dr. G. Dromey:

1. We agree with the discusser that in distribution systems several fixed phasor sources have to be considered simultaneously. This does not present any difficulty in our method. One of the sources will be the root and each of the other sources will be represented as connected to the root by a branch containing a voltage source equal to the (complex) voltage difference of the two sources. This branch must belong to the system tree. Another tree branch will therefore become a link thus increasing the number of independent loops (by one for each of the sources other than the root). When the breakpoint voltages are calculated (as voltage differences at two adjacent points), the voltages of the new voltage sources have to be taken into account. Alternatively, the problem of an additional source point can be solved by adding a new breakpoint for that bus: this will serve to obtain the P, Q injections to the bus with the prescribed complex voltage. Fundamentally, the calculations remain as simple as in the case of a single source.

2. In many power systems that consist of transmission networks the number of lines is of the order of approximately 1.5 times the number of buses. Correspondingly, according to equation (a), the number of independent loops is about half of the number of buses. Clearly, for such cases a load flow method based on loop equations will not be appropriate. Only in the case of networks which are weakly meshed will the use of loop equations become attractive, since then the size of the sensitivity matrix remains small. It is in such situations that the load flow program of the paper becomes computationally superior to conventional load flow methods. Because of the relatively small size of the sensitivity matrix, we did not feel that it would be needed to decouple the matrix as it is done in the well known Fast Decoupled Load Flow. This is why no problems are expected related to the R/X ratio. For the same reason, we did not examine other computational simplifications as, for example, the approximation of the sensitivity matrix by its main diagonal.

3. Regarding the possibility of handling multi-voltage level systems, we believe that this can be done by either a per-unit approach or by reducing different voltage levels to a single common one with proper care to the representation of phase shifting produced by transformer connections.

Manuscript received September 1, 1989.