

**THE LADDER LOAD-FLOW METHOD EXTENDED TO  
DISTRIBUTION NETWORKS**

by

Mikle Val Ellis

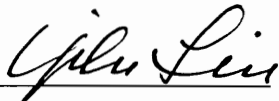
Dissertation submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY  
in  
ELECTRICAL ENGINEERING

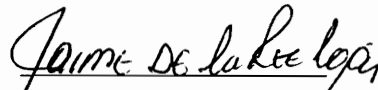
APPROVED:



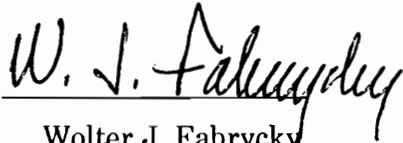
Robert P. Broadwater, Chairman



Yilu Liu



Jaime De La Ree Lopez



Wolter J. Fabrycky



Terry L. Herdman

July, 1994

Blacksburg, Virginia

# THE LADDER LOAD-FLOW METHOD EXTENDED TO DISTRIBUTION NETWORKS

by

Mikle Val Ellis

Robert P. Broadwater, Chairman

ELECTRICAL ENGINEERING

(ABSTRACT)

The rigorous load-flow analysis of Distribution Networks requires the modeling of mutual coupling, multiphase possibilities and unbalanced loading. The Ladder Load-Flow Method meets these requirements, but is limited to radial systems. The Switch Compensation Method presented here, extends the Ladder Load-Flow Method to solve Distribution Networks.

A network analysis program has been developed using the Electric Power Research Distribution Engineering Workstation (DEWorkstation). The program uses the pointers provided by the DEWorkstation to represent the connectivity of the system. The use of these pointers increases the speed of execution while naturally handling the sparsity of the system. The user is also provided with "on line" attribute and topological data maintenance.

The extension technique is based on Thevenin's Theorem. This approach allows the accurate determination of the equivalent impedance for networks which contain series and shunt elements. The equivalent impedance can be changed during the iterative solution process. This allows the algorithm to simulate the operation of control devices which significantly alter the impedance of the network. A transformer model for the forward and reverse trace of the Ladder Method is also presented.

## ACKNOWLEDGMENTS

The completion of this research would not have been possible without the support of many people. I would like to express my appreciation to the Electric Power Research Institute which provided the financial support for this research. I want to express my most sincere thanks and admiration to my wife, Sharie Sue. The completion of this work would not have been possible without her support and guidance. She has sacrificed more than words will ever be able to express. I will always be in her debt.

I want to thank my advisor Dr. Robert Broadwater for his patience and help. He has always been willing to forfeit his time to provide assistance. I will always consider Robert a true mentor and friend.

I would like to dedicate this work to my three children, Caroline, Robin and Jacqueline.

## TABLE OF CONTENTS

### Chapter

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Statement	5
1.2	Scope of Dissertation	7
<b>2</b>	<b>Literature Review</b>	<b>9</b>
2.1	Overview	9
2.2	The Ladder Load-Flow Method	12
2.3	The Use of Linked Lists	15
2.4	The Work of Others	17
2.5	Construction of the Thevenin Matrix	19
2.6	Contributions	20
<b>3</b>	<b>DEWorkstation</b>	<b>23</b>
3.1	Description	23
3.2	The DEWorkstation Architecture	24
3.3	Pointers and Load-Flow Algorithm	26
3.4	Pointers Used By Load-Flow	27
3.5	Radial Load-Flow	29
3.6	Validation	31
<b>4</b>	<b>The Switch Compensation Method</b>	<b>36</b>
4.1	Theory	36
4.2	Formation of Sensitivity Matrix	38
4.3	Current Injection	42
4.4	Voltage Correction	42
4.5	Implementation	46
<b>5</b>	<b>Calculation Example and Results</b>	<b>50</b>
5.1	Determination of Sensitivity Matrix	50
5.2	Test Network	54
5.3	Implementation	59
5.3	Results	62

<b>6</b>	<b>Conclusions and Recommendations</b>	<b>75</b>
6.1	Comparisons of Solution Times	75
6.2	Calculation of Sensitivity Matrix	77
6.3	Updating the Sensitivity Matrix	78
6.4	Absence of Matrix Operations	81
6.5	Voltage Correction	82
6.6	Recommendations	83
6.7	Future Considerations	84
<b>Appendix</b>		
<b>A</b>	<b>Unbalanced Three-Phase Transformer Modeling</b>	<b>85</b>
A.1	Current Reverse Trace	86
A.2	Voltage Forward Trace	92
A.3	Phase Shift of Wye-Delta	94
A.4	Computer Code	96
<b>B</b>	<b>Network Load-Flow Driver and Modules</b>	<b>102</b>
B.1	Main Driver	104
B.2	Determination of the Sensitivity Matrix	107
B.3	Initialize Co-tree Currents	109
B.4	Determine Current Change	110
B.5	Voltage Change	111
B.6	Network Convergence	112
	<b>References</b>	<b>114</b>
	<b>Vita</b>	<b>117</b>

## List of Figures

Figure 1.1.	Circuit Model for Load-Flow .....	4
Figure 1.2.	Load Flow Solution .....	5
Figure 2.1.	Circuit For Illustrating the Ladder Method .....	12
Figure 2.2.	Currents for Ladder Method .....	13
Figure 2.3.	Circuit for Illustrating Reverse Trace Pointers .....	16
Figure 2.4.	Circuit for Illustrating Forward Trace Pointers .....	16
Figure 2.5.	Circuit for Illustrating Loop Compensation .....	18
Figure 2.6.	Resulting Circuit .....	18
Figure 3.1.	DEWorkstation Architecture .....	25
Figure 3.2.	IEEE 34 Bus Radial Distribution Test Results .....	32
Figure 3.3.	Test Circuit Connection Diagram .....	34
Figure 4.1.	Mutually Coupled Line Model .....	37
Figure 4.2.	Graph Illustrating Two Methods To Calculate Resistance .....	39
Figure 4.3.	Determination of Thevenin Resistance .....	40
Figure 4.4.	Example System .....	43
Figure 4.5.	Convergence Characteristics .....	44
Figure 4.6.	Flow Chart of Initialization of Sensitivity Matrix .....	48
Figure 4.7.	Flow Chart of Predictor/Corrector Iterative Scheme .....	49
Figure 5.1.	Thevenin Impedance Test Circuit .....	51
Figure 5.2.	Mutually Coupled Impedance Test Circuit .....	53
Figure 5.3.	Resultant Sensitivity Matrix .....	54
Figure 5.4.	Test Network .....	56
Figure 5.5.	Typical Cable Impedance and Admittance Values .....	57
Figure 5.6.	Test Network Loads .....	59

Figure 5.7.	Radial Part of Test Network .....	60
Figure 5.8.	Typical Voltage Magnitude Values .....	63
Figure 5.9.	Typical Current Magnitude Values .....	64
Figure 5.10.	Convergence of Complete Algorithm .....	65
Figure 5.11.	Convergence without Updating Sensitivity Matrix .....	66
Figure 5.12.	Convergence with changing all three co-tree currents .....	67
Figure 5.13.	Test Case Comparison .....	68
Figure 5.14.	Test Network with Control Devices .....	70
Figure 5.15.	Test Network with Control Devices Voltages .....	71
Figure 5.16.	Test Network with Control Devices Currents/Power Factor .....	72
Figure 5.17.	Typical Voltage Magnitude Values for Heavily Loaded Circuit .....	74
Figure A-1	Example Delta-Wye Configuration .....	87
Figure A-2.	Example Wye-Delta Configuration .....	88
Figure A-3.	Superposition Used To Solve Wye-Delta Configuration .....	89
Figure A-4.	Wye Connection with Single Phase Load .....	91
Figure A-5.	Thevenin Equivalent Circuit .....	92
Figure A-6.	Voltage Relationships for Wye-Delta Configuration .....	93
Figure A-7.	Delta-wye Phase Transformation .....	95

## Chapter 1

### INTRODUCTION

Electrical utilities in the United States are expected to supply their customers with continuous electrical power. Electrical power is produced at generation plants by three phase generators. Since the generation plants are typically located some distance from the customers, the power must be transported across a set of three transmission lines. The power system can be divided into two distinct subsystems: the "Transmission System" and the "Distribution System".

Large amounts of power are transmitted over long distances at high voltages (135 kV up to 765 kV) from generation plants to sites near the customer called substations. This portion of the system is typically referred to as the Transmission System. High voltages are used to reduce losses. Several generation sites frequently supply the same substation. This increases the reliability of the system by connecting the substations into a looped or networked system. The power flowing on each of the three transmission lines is typically the same. Consequently, the transmission system is characterized as a "balanced" system.

At the substation the voltage is reduced and the power is dispersed in smaller amounts to the customers. This portion of the system is referred to as



the Distribution System. One substation will typically supply many customers with power. Therefore, the number of separate transmission lines in the distribution system is many times that of the transmission system. Individual customers are generally supplied from one substation. However, in cases where high reliability is required, customers are supplied by several substations. These "Distribution Networks" are typically located in the downtown areas of large cities. Most customer loads connect to only one of the three transmission lines in the distribution system. Therefore, the power flow on each of the transmission lines is typically not the same and the system is "unbalanced". Unbalanced systems are not as efficient as balanced systems [1].

A "load-flow" study is a computer simulation of the power system which predicts the amount of power flowing on each transmission line connecting the generation plants to the customers. "Load studies are essential in the planning the future development of the system because satisfactory operation of the system depends on knowing the effects of interconnections with other power systems, new loads, new generation stations, and new transmission lines before they are installed [2]".

The load-flow solutions provide valuable information to the planner, designer and operator of the system. System planners are interested in studying the power system 10 or 20 years in the future. A power company

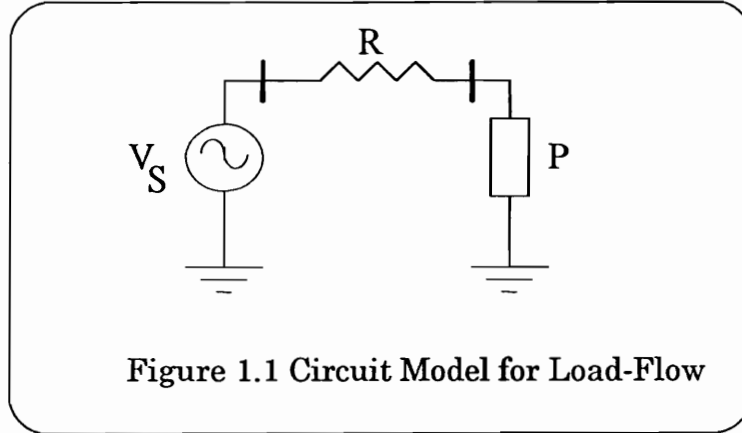
must know far in advance problems associated with the load exceeding either the transmission or generation capacity. More than 10 years can elapse between initiating the plans for a new power plant or transmission line and bringing it on the line.

The designer needs to evaluate quickly and economically design alternatives for complex systems. The load-flow program provides the designer with the ability to quickly simulate the operation of the system for many mutually exclusive design alternatives.

The operator is interested in the reliable and efficient operation of the existing system. The load-flow program can predict the losses of different system configurations which are available to the operator. The fast, efficient, and accurate operation of this program has been the topic of research for several decades.

The solution of the load-flow problem requires matching the characteristics of the source and load through the circuit laws imposed by the transmission lines. Generation plants are regulated to maintain a constant voltage output. Therefore, power sources are often modeled as constant voltage sources in the load-flow program. Customer loads are modeled in a variety of ways. The most common load model is a constant power load. The transmission lines are modeled as constant impedances. Figure 1.1 illustrates

a simplified circuit model for a single customer load connected to a generator through a transmission line.



The laws of circuit theory require that the voltage across the transmission line plus the voltage across the load to be equal to the source voltage. The voltage across the transmission line can be expressed as the product of the current through the line multiplied by the line impedance (Ohm's Law). This results in the following linear relationship:

$$V_S = IR + V_L$$

where  $V_S$  is a constant.

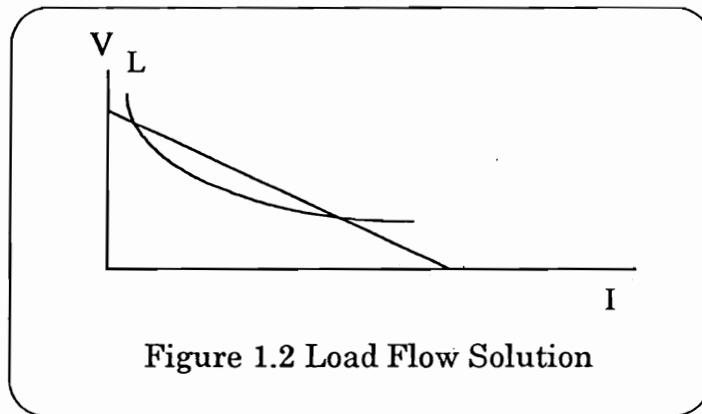
A constant power load may be expressed by the following relationship:

$$P = V_L \times I$$

where  $P$  is a constant.

The solution of these two relationships for the model shown in Figure 1.1 is illustrated graphically in Figure 2.1. It should be noted that this simple example can be solved in closed form. However, as more generators, loads and transmission lines are added to the system a numerical method will be required to achieve a solution.

The solution of the load-flow problem is the intersection of the constant power curve and the linear relationship resulting from the circuit equations. The curves may intersect at more than one point as illustrated in Figure 1.2. However, only the point of the highest load voltage is a practical solution for a power system. The curves may not intersect, in which case no solution is possible.



### 1.1 Problem Statement

It is becoming apparent that current Transmission System load-flow algorithms are inadequate for Distribution Systems. The main differences are the number of different types of devices, multiphase possibilities, and widely varying types of loads in Distribution Systems. Distribution Networks are a good example of all of these differences.

Portions of the network are unbalanced due to single-phase loads and radial spurs. When the system is unbalanced, the mutual coupling between phases must be considered. Extremely large networks which consist of over

12,000 different devices are not uncommon. Consequently, the memory usage and computation time required to achieve a solution become the important performance criterion. It is the intention of this research to develop an algorithm that meets the requirements of rigorous load-flow analyses on Distribution Networks.

Notwithstanding these differences, an important lesson can be learned from several decades of software development for Transmission Systems [3]. Load-flow programs require topological data and attribute data. The topological data represents the interconnections among the components in the system. The attribute data represents the transmission line impedances and load data. Traditionally, topological and attribute data were not directly available to the load-flow program. Instead, this data was combined before it was placed in the database (i.e.,  $Y_{BUS}$ ). Consequently, adding, removing, or switching of components was an off-line maintenance task.

Today the goal is to provide "on line" attribute and topological data maintenance. Graphical User Interfaces (GUI) provide the user with the capability to insert, delete and switch components "on line". However, the GUI data must be converted into a form required by the load-flow program. This can consume considerable CPU time and memory for a large Distribution Systems. If the data is in the same form as required by application programs, the overall system will be faster and use less memory. This is one of the

design goals of the Distribution Engineering Workstation (DEWorkstation) currently under development by the Electric Power Research Institute (EPRI).

## 1.2 Scope of Dissertation

Load flow algorithms exist which solve large unbalanced and mutually coupled radial distribution circuits. One such method is the Ladder Load-Flow Method. The primary contribution of this research is the extension of the Ladder Load-Flow Method to solve looped systems.

The Ladder Load-Flow Method has several advantages over algorithms developed primarily for transmission analysis. The most significant advantage is that the Ladder Method can be formulated without the use of matrix operations. Matrix operations consume considerable time and memory for large systems. This implementation of the Ladder Load-Flow Method uses "pointers" to represent the connectivity of the system. Such an approach increases the speed of execution while naturally handling the sparsity of the system.

This method will be applied to the solution of distribution networks. The key elements of this new approach are as follows:

1. The extensive use of pointers for reduced processing time and memory usage.
2. The representation of mutual coupling among transmission lines.
3. On line attribute and topological data maintenance.

4. The representation of controlled devices such as Tap Changing Under Load Transformers and Capacitors.
5. Run on a personal computer platform.

## Chapter 2

### LITERATURE REVIEW

A literature search has been conducted in the area of unbalanced and mutually coupled load-flow algorithms. The application of these algorithms to large distribution systems was also investigated. Each algorithm is derived from the circuit law equations and the load characteristics. However, the algorithms differ in the mathematical formation of the load-flow problem.

#### 2.1 Overview

The Newton-Raphson load-flow algorithm has demonstrated excellent convergence on Transmission Systems for more than two decades. To apply this method to the solution of load-flow equations the real and imaginary power at each bus are expressed as a function of the complex voltage and line admittances. The change in power at each bus is approximated by a Taylor series expansion of these equations. The higher order terms are generally neglected and the resulting jacobian consists of the partial derivatives of P and Q with respect to the change in voltage at each bus. Mutual coupling can be represented by modifying the line admittance equations [4,5,6 ].

The Newton-Raphson algorithm has been used successfully on relatively small Distribution Systems. However, as the size of the system increases the memory requirements and computation time of this method



become excessive [7]. The size and unbalanced nature of Distribution Networks makes the Newton-Raphson approach unattractive [8,9,10,11]. "The major short coming of this method is the requirement that the jacobian matrix, with rank approximately four times that of  $Y_{BUS}$  be recalculated for each iteration" [11 ].

The Gauss-Seidel method is another familiar load-flow algorithm. The current entering and leaving each node is expressed as a function of the bus voltage and the real and imaginary power at each bus. This results in a set of linear algebraic equations which can be solved by the Gauss iterative method. Mutual coupling can be represented by modifying the line impedances in the same manner as the Newton-Raphson Method. The Gauss-Seidel method generally requires more computation time than the Newton-Raphson method. The method often has convergence difficulties on distribution systems [11,12].

A derivative of the Gauss-Seidel method known as the  $Z_{BUS}$  Gauss approach has been used to solve Distribution Networks [11]. The voltage at each bus is expressed as a function of line impedance and the load current. If the loads are modeled as constant currents the resulting equations are linear. The principle of superposition is used to determine the voltage at each bus resulting from the voltage source and the load currents. The impedance matrix is formed by factoring the  $Y_{BUS}$  matrix which includes the mutual coupling effects. Optimal ordering and L-U decomposition are employed to

reduce memory usage and processor time. No comparison of processor time and memory usage are available.

Several recent methods iterate using circuit models to solve Distribution Systems [8,10,12]. These methods are characterized by the absence of matrix operations. Circuits are solved using "traces". The data can be stored in a dynamically allocated linked-list [13]. The method begins with an approximation of the voltage at each load bus. The load current is then determined using either circuit theory or conservation of energy principles. The resulting circuit is linear. The solution of this linear circuit will result in new values for the voltage at each load bus. The process is iterative and repeated until the bus voltages converge.

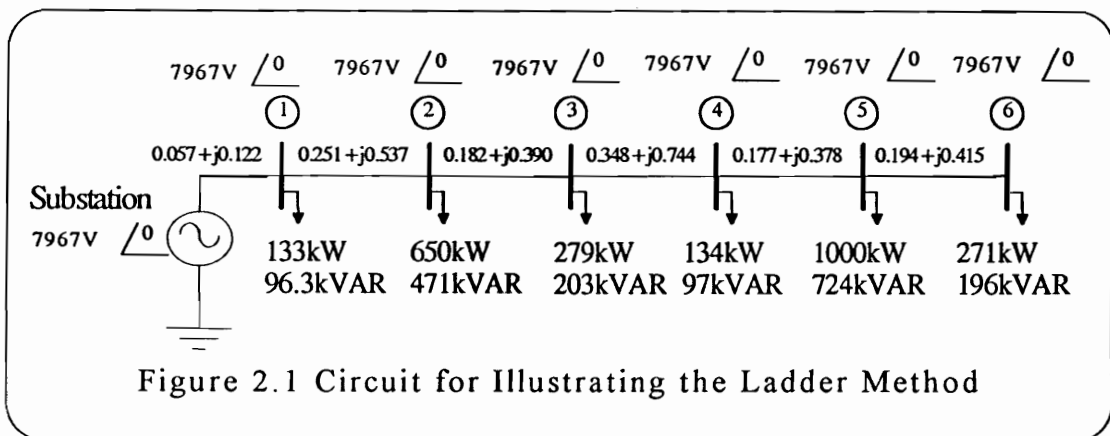
Mutual coupling can be modeled in the circuit as current controlled voltage sources. These methods were developed for radial systems but can be extended to networks using the principle of superposition [14]. The method used on radial systems is competitive with other load-flow methods in terms of total processor time and memory usage for large systems [7]. Modeling errors can also be detected using circuit principles [8,15].

Another method linearizes the load-flow problem about the balanced operating point [16]. This method uses "distribution factors" to relate the change in the line loading to a generation. The distribution factors can be approximated with elements of the Z matrix. These distribution factors can

then be used to model any imbalance or mutual coupling in the network. This method has mainly been employed on Transmission Systems. No comparisons are available on memory usage or computation time.

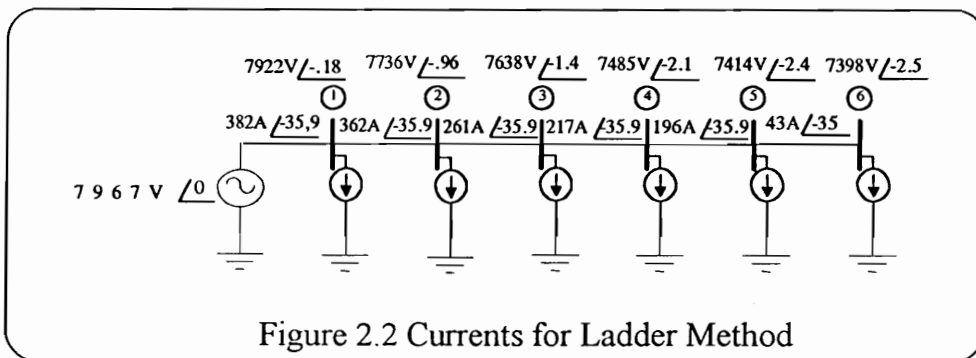
## 2.2 The Ladder Load-Flow Method

Only the trace methods can allow "on line " data attribute and topological data maintenance for the Load-Flow algorithm. Trace methods are also well suited for the representation of controlled devices. The other methods require the construction of the  $Y_{BUS}$  matrix. Changes to either the attribute , topological or control data requires the reconstruction of this matrix. Based on this consideration The Ladder Load-Flow algorithm was selected for the solution of distribution networks. The Ladder Load-Flow algorithm is a trace method. The algorithm can represent mutual coupling. It is competitive with other methods in both memory usage and total processor time. The basic ladder method will be explained with the aid of the circuit shown in Figure 2.1.



In Figure 2.1 bus voltage magnitudes are shown in volts, angles in degrees, line impedances in ohms, and bus loads in kilowatts and kilovars. The voltage magnitude and angle are specified at the sending-end or substation. The iterative algorithm begins by estimating the voltage at each load bus. The substation voltage may be used as an initial estimate as shown in Figure 1. With these voltages it is possible to compute the resulting load currents as  $I = (S/V)^*$ .

Once the load currents are determined the resulting line currents can also be determined as shown in Figure 2.2. The voltage drop across each line can then be calculated as the product of the line current and line impedance. This results in a new estimate of the voltage at each bus shown in Figure 2.2. The process is iterative and repeated until the bus voltages converge.



In principle, the ladder method is based on a circuit approach first proposed in 1967 [26]. The method begins by converting the nonlinear kVA loads into linear load currents. This results in a linear circuit. This linear circuit is then solved which results in new estimates for the load currents.

The process is very similar to a Gauss-Seidel iterative method of solving algebraic equations. However, the iterations take advantage of the circuit structure.

This circuit structure approach allows the algorithm to use circuit principles to determine convergence. The currents are summed back to the source at each iteration. An indication of convergence can be obtained by comparing the source currents at each iteration. If the source current is different from that of the prior iteration the algorithm has not reached convergence. If the source current is the same as that of the prior iteration then all the voltages in the circuit can be compared to verify convergence.

The detection of bad data may also be determined using circuit principles. Line impedances which are too large are determined by comparing the computed voltages to the nominal value. Load data which is likewise too large can be detected by comparing the estimated line currents with the rated values.

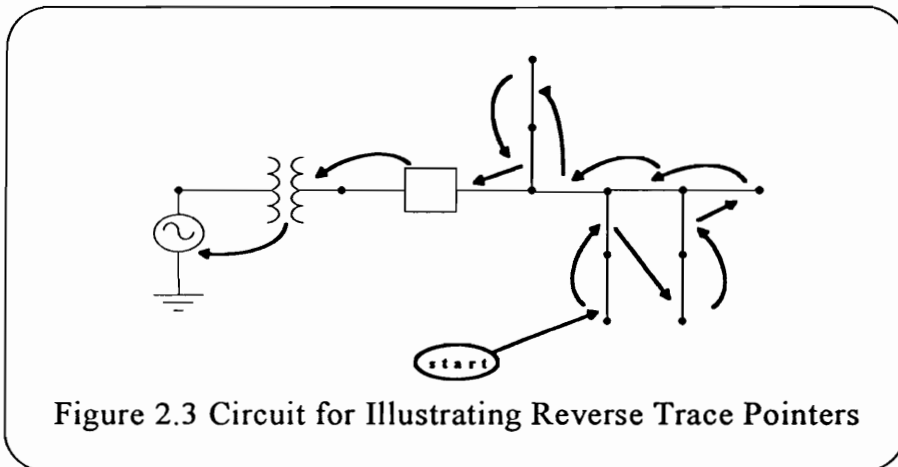
These techniques of checking for convergence and bad data are not mentioned in the literature. However, these simple modifications have proven to be very valuable. The algorithm's speed is enhanced by eliminating the need to compare all the voltages in the system from one iteration to the next. The bad data checks detect the two most common sources of error in data provided to the load flow program.

### 2.3 The Use of Linked Lists

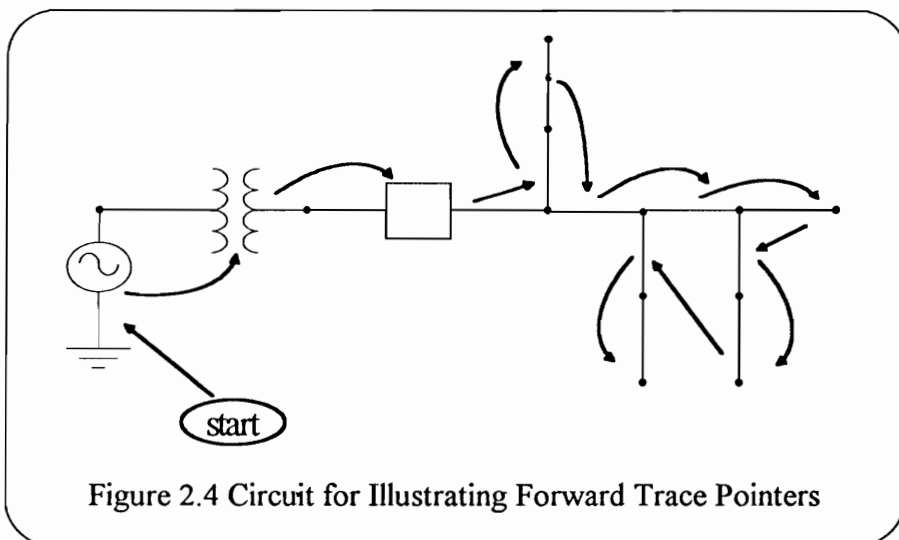
The implementation of this method into a load-flow program uses "traces" to perform the two basic operations required by the method: A reverse trace and a forward trace. A circuit trace is the order in which an algorithm process the data in the component linked list. The reverse trace sums the estimated currents back to the source. The forward trace distributes the voltages out to the ending nodes.

In a linked list each circuit component has a data structure attached to it. This data structure is accessed by pointing to a memory location which is the start of the data structure. Components are linked together by allowing the data structure of one component to contain pointers to another component's data structure. This links the data structures together.

The linked list of reverse trace pointers links the components together in such a way as to facilitate the addition of currents back to the source. This can be illustrated with Figure 2.3. The only requirement in forming this linked list is that for a given component all the downstream components attached to a component must be processed before the given component is processed. It is possible that more than one linked list may satisfy these requirements for a circuit.



The formation of the forward linked list requires all upstream components of a given component be processed before the given component. This is illustrated in Figure 2.4.

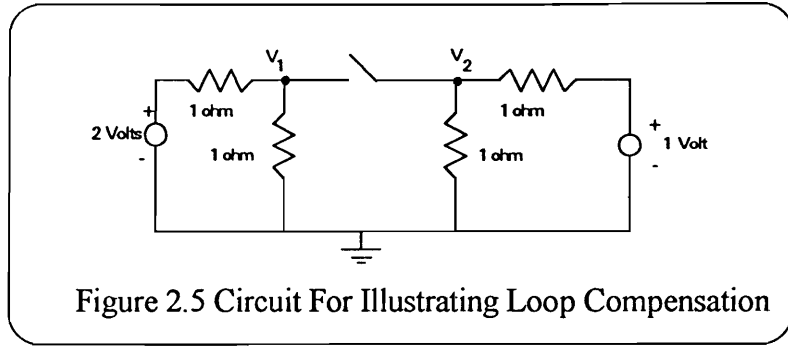


## 2.4 The Work of Others

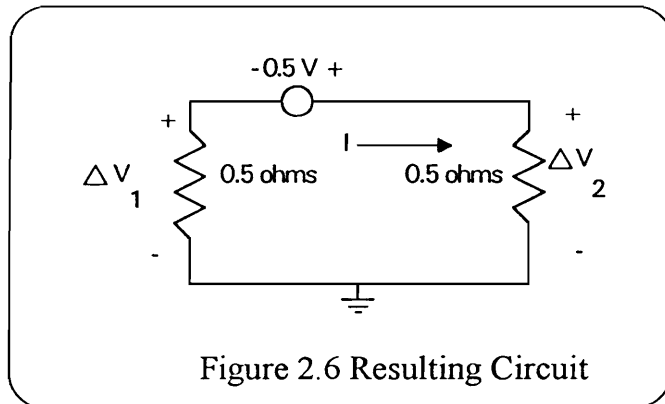
Several authors have explored extending radial trace methods to solve looped systems. The basic technique used is the principle of superposition. This same technique has also been used to determine the switching response of circuits [27].

Consider the two linear radial circuits separated by a switch as shown in Figure 2.5. The solution of each radial circuit will result in a voltage difference across the open switch. If this voltage difference is zero, the switch can be closed without affecting any of the voltages and currents in each of the radial circuits. Consequently, if the voltage difference is zero, the solution of the looped system has been achieved. This provides the motivation behind the technique which is simply to determine a current which when added to the load current at the connection points will make the voltages equal at these same points. Borrowing terminology from graph theory this switch will be referred to hereafter as a "co-tree component". A "co-tree component" is a component which if removed from the system breaks a loop. Removing all co-tree components results in a radially connected system.





The current that would flow through the closed switch is determined by applying a fictitious voltage source at the co-tree component which is equal in magnitude but opposite in sign to the voltage difference across the co-tree component. In accordance with the principle of superposition all other active voltage sources are shorted and active current sources are opened. The Thevenin impedance is determined on both sides of the co-tree component. The current that would flow through the co-tree component is determined by solving this resulting circuit shown in Figure 2.6.



The desired currents at the co-tree components are determined using the Thevenin equivalent circuit shown in Figure 2.6. These steps are

repeated until the voltage difference across the co-tree component is within prescribed limits.

## 2.5 Construction of the Thevenin Matrix

Several different approaches have been proposed for determining the Thevenin impedance seen by the co-tree component. All of the approaches are based on the formation of a matrix which is dimensioned by the number of co-tree components in the system. D. Shirmohammadi determines the impedance on each side of the co-tree component by injecting a current at each co-tree component with the source and all loads removed [14]. The ratio of the voltage and current at the co-tree component is the Thevenin impedance. The impedance values are then stored in a matrix. As the size of the system increases this "Preparation" phase which includes the process of building the linked lists and the Thevenin equivalent matrix requires nearly half of the total CPU time.

G.X. Luo uses a graph based approach in the formation of this matrix [9]. The first step in this approach is to find the unique path connecting the two side nodes of the co-tree component 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. Finally the elements of the matrix are calculated.

These approaches assume that the shunt components in a power system such as capacitors and shunt admittance can be neglected. This is usually a good assumption for the normal operation of a power system. They also assume that transformer turns ratio remains constant throughout the solution process. This assumption is violated when the algorithm must simulate the automatic control of tap changing under load transformers (TCUL) and other similar devices.

## 2.6 Contributions

There are three major differences between the algorithms used by others and the Switch Compensation algorithm proposed here. The first is the manner in which the Thevenin equivalent circuit shown in Figure 2.6 is determined and used by the algorithm. The second is the application of the principle of superposition to change the voltage at the co-tree components as well as the current. Third the algorithm uses pointers to process the topology of the network.

The Switch Compensation method uses the forward and backward traces of the Ladder method to determine a sensitivity matrix at each co-tree component. The determination of this Sensitivity matrix is also based on the principle of superposition. Assume the voltage between two terminals in a linear network is known. If a current is injected at these same two points the voltage will change. This change in voltage divided by the injected currents

determines the Thevenin impedance of the network from the two terminals specified.

It should be noted this current injection could be the desired current determined from the calculation of the circuit shown in Figure 2.6. Consequently, once the process is started the determination of the Thevenin impedance is inherent in the method. At each iteration a change in current is determined at each co-tree component. Each radial circuit is converged with this current injected at the co-tree component. This will result in a change in voltage at the same co-tree component. This change in voltage divided by the change in current provides the approximation of the Thevenin impedance at the co-tree component for the next calculation of current. Control devices which may change between iterations can be solved in this manner.

The principle of superposition may also be applied to determine the resulting change in voltage at a co-tree component. The calculation of this change in voltage due to the injection of current at co-tree components proceeds directly from the circuit shown in Figure 2.6:

$$\Delta V_1 = -.25 \times I \quad \Delta V_2 = .25 \times I$$

This calculation provides an estimate as to the change in voltage at the co-tree components, thereby, increasing the rate at which the algorithm will converge on each radial circuit. The use of circuit principles to determine convergence and detect bad data is another unique feature of this algorithm.

Load-flow algorithms designed for radial systems which use pointers to define the topology of the system appear in the literature. The extension of these algorithms to looped systems is achieved by resorting back to the use of matrix definitions for the system. Many of the advantages of using pointers to define the topology are lost with this approach. The goal of this work is to formulate a load-flow algorithm for looped systems using pointers to process the topology of the system.

## Chapter 3

### DEWORKSTATION

The Electric Power Research Institute Distribution Engineering Workstation is a software package which provides a data environment designed to integrate the analysis, planning, design, and operation needs of distribution engineers. DEWorkstation has a novel Application Programmer Interface [17]. The "on line" features of the Application Programmer Interface provide the motivation behind the design of the load flow application.

#### 3.1 Description

Several years ago, the Electric Power Research Institute (EPRI) began development of a software package designed to aid distribution engineers. A survey was conducted of the needs of distribution engineers to determine the desired characteristics of the software package. Among other items the survey determined that users should be allowed to easily experiment with alternative solutions to a design. The EPRI Distribution Engineering Workstation (DEWorkstation) exhibits this characteristic and meets the design and analysis needs of the distribution engineer [18].

Traditional transmission system software architecture (as used in EMS and SCADA systems) do not meet the needs of the distribution engineer [3].

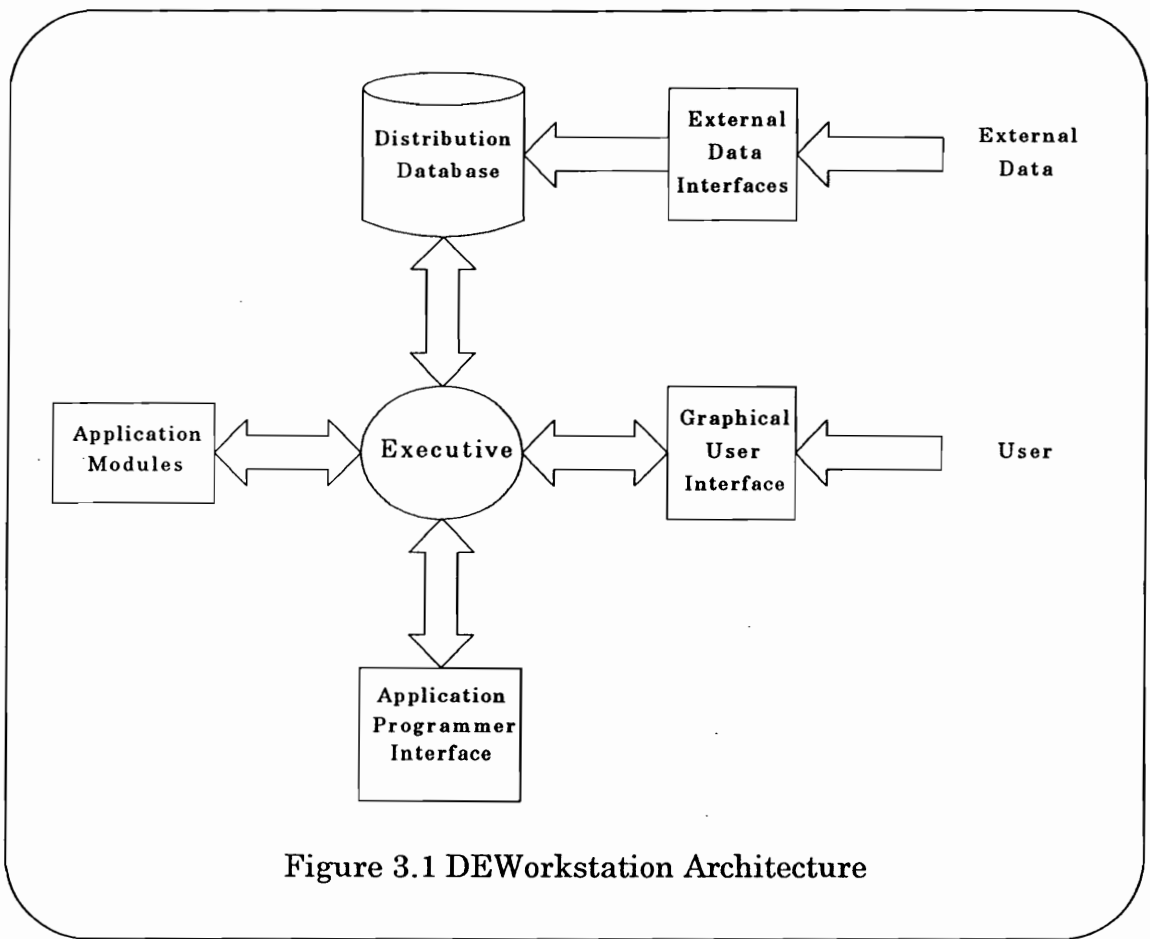
The primary difference is that the architecture must support a database which is many times larger than the typical transmission system.

The easy evaluation of alternative solutions requires that system components can be added and removed "on-line". Adding or removing power system components has usually been an inconvenient, off-line maintenance task. Adding or removing components generally requires that an admittance matrix be reconstructed. This task requires considerable CPU time and memory.

Another shortcoming of present design practices is that they are generally not extensible. Utilities prefer to invest in large software products in stages. This requires that the architecture provide a framework for gradual evolution of the system over an indefinite period of time.

### **3.2 The DEWorkstation Architecture**

DEWorkstation consists of an Executive, core features, and applications. The executive manages the user and application interfaces with the core. The workstation core consists of the data schema and the graphical user interface (GUI) functions. Finally, applications are provided by software developers. Figure 3.1 exhibits the general features and interactions of the DEWorkstation.



The DEWorkstation is being developed as an open architecture system. The term open architecture implies that programmers who wish to develop application modules for DEWorkstation have a well defined executive and core features on which to build their applications.

A key design feature of DEWorkstation is the separation of topology, attribute, and application data. Attribute data is stored with "Parts" objects. Application data is stored with application objects. Topological data is stored



with system, circuit, and component objects. This separation of data allows "on line" data maintenance and makes the workstation extensible.

Pointers are used to manage the topological relationships among workstation objects. They are maintained by the executive and are used to implement such graphical operations as addition, insertion and deletion of components. Operations and manipulations can be performed extremely fast. The addition, insertion and deletion of components is achieved by modifying four pointers in a linked-list. To further increase workstation speed and to provide flexibility, all data object pointers are made available to application programmers.

### **3.3 Pointers and Load-Flow Algorithm**

The use of pointers to define topology has several advantages for the load-flow application. Pointers eliminate the need to define and maintain a system admittance matrix. Consequently, the use of sparsity techniques is not required. This makes the algorithm simpler and faster.

The use of pointers to define topology can only be fully exploited if the applications employ the pointers maintained by the executive. The performance of the workstation is degraded if applications must preprocess the data provided by the executive. The load-flow algorithm in this environment must be implemented using pointers.

### 3.4 Pointers Used By Load-Flow

The topology of a given circuit is maintained by pointers to single, two, or three phase component objects. These pointers are stored with the component objects themselves. A doubly linked list uses forward and backward pointers to link components for each circuit.

System objects point to substations and circuits. Substations and circuit objects contain pointers to starting and ending components. Each circuit object also maintains a linked list of switches which have been closed to create loops.

Two additional pointers are defined that determine physical connectivity within a single circuit. These pointers are referred to as the feeder path and brother pointers. For a system of radial circuits, the feeder path pointer of a given component is the next component toward the substation that feeds the given component. The brother component for a given component gives the first component that occurs in the forward trace that is not fed by the given component. Feeder path pointers are used in a linked list, where as brother pointers are not.

The load-flow algorithm uses the linked list of backward trace pointers and the feeder path pointers provided by the core of the DEWorkstation to sum the currents back to the source. The currents from the previous iteration are removed. A pointer is provided by the workstation to the first component

in the backward trace. This is the first component in the backward linked list of components.

The algorithm then determines the type of component which is to be processed. In the case of lines and cables the load current is first added to given line or cable. The load current is then added to the feeder component. This component is determined by examining the feeder path pointer of the given line or cable.

The next component to be processed is determined by examining the backward trace pointer of the present component. The end of the linked list is indicated by a NULL pointer.

The load-flow algorithm uses the linked list of forward trace pointers and the feeder path pointers provided by the core of the DEWorkstation distribute voltage drops from the source to the ending components. A pointer is provided by the workstation to the first component in the forward trace. This is the first component in the forward linked list of components.

The algorithm then determines the type of component which is to be processed. In the case of lines and cables voltage drop is calculated by the product of the current through the line and the impedance of the line or cable. The resulting voltage at the end of the line or cable is then determined by subtracting this voltage drop from the voltage of the feeder component.

The next component to be processed is determined by examining the forward trace pointer of the present component. The end of the linked list is indicated again by a NULL pointer.

At an open switch that closes into another radial circuit, a pointer to the component in the adjacent circuit is set. If a switch is closed to create a loop, then adjacent component pointers (co-tree pointers) are set to the closed switch, and the switch is referred to as a co-tree component. The number of co-tree components defines the number of loops in the system.

### 3.5 Radial Load-Flow

The load-flow algorithm in DEWorkstation is a derivative of the Ladder-Method. This method is implemented directly with the pointers maintained by the executive. Consequently the user may add , insert, or delete components on line. The translation of data, such as forming the  $Y_{BUS}$  is not required.

The Load Flow program simulates the automatic control of tap changing transformers and controlled capacitors. The transformer controller allows the user to control the customer voltage at a selected point in the circuit. A load drop compensator is also simulated by the algorithm. Capacitor control can be time, power factor or voltage control. Power factor control is available with a voltage override feature.

The algorithm uses the circuit principles discussed in Section 2.2 to simulate automatic control. During the solution process the control variables in the circuit will change from one iteration to the next iteration. If the controller is turned on throughout this solution process the control actions will result from the iterative nature of the algorithm. Controllers which are allowed to move throughout the iterative process are likely to oscillate between their maximum and minimum set points. This will increase the number of iterations required to reach a solution. It is also possible that the simulation of controllers in this manner will cause the algorithm to diverge.

The current provided by the power source of a radial circuit can be used to indicate when controllers should be turned on during the iterative solution process. In a radial circuit, this current provides an indication of convergence. Once the change in source current between iterations is below the threshold of the controllers, the controllers can be turned on in the circuit.

The data used by the load-flow algorithm is arranged by the executive in memory and is immediately available to the application. Consequently, when the load-flow application is executed no hard disk accesses occur. Therefore, the application runs entirely in memory, and thus as quick as possible.

### 3.6 Validation

The DEWorkstation load-flow algorithm is based on the DANE or the Distribution Analysis and Economic Evaluation Workstation developed at Virginia Tech. DANE has been used effectively by several utilities and they have considered it a suitable tool for their work. Arkansas Power and Light has used the DANE load-flow to analyze 3000 bus systems.

The features of the DEWorkstation radial load-flow have been independently verified on over 35 different circuits [19]. The results have also been compared to other distribution analysis programs. Figure 3.2 compares the customer level voltage of the DEWorkstation Load-Flow with RDAP Version 2.2W. RDAP is a commercial product of WH Power Consultants of Las Cruces New Mexico. The buses are listed in Figure 3.2 in order of increasing distance from the substation. Bus 1890 is 34.754 miles from the substation.

The column labeled "%D" in Figure 3.2 is the normalized Euclidean between the phasor voltages of the DEWorkstation and RDAP. The calculation is performed as follows [20]:

$$\%D = \frac{|V_1 \angle \theta_1 - V_2 \angle \theta_2|}{123.8} \times 100$$

The voltage of 123.8 volts is the nominal customer level voltage at the substation.

Bus	PHASE A-N					PHASE B-N					PHASE C-N				
	RDAP		DEW		%D	RDAP		DEW		%D	RDAP		DEW		%D
	MAG	ANG	MAG	ANG		MAG	ANG	MAG	ANG		MAG	ANG	MAG	ANG	
800	123.6	0	123.8	0	0.16	123.6	-120	123.8	-120	0.16	123.6	120	123.8	120	0.16
802	123.3	-0.06	123.5	-0.06	0.14	123.4	-120.1	123.6	-120.1	0.17	123.4	119.9	123.5	119.9	0.14
806	123.1	-0.1	123.3	-0.1	0.13	123.3	-120.1	123.5	-120.1	0.16	123.2	119.92	123.4	119.9	0.16
808	119.6	-0.91	119.9	-0.9	0.24	121.4	-120.77	121.5	-120.8	0.1	120.4	119.35	120.6	119.3	0.18
810						121.4	-120.77	121.5	-120.8	0.1					
812	115.6	-1.9	115.9	-1.86	0.25	119.3	-121.55	119.3	-121.6	0.08	117	118.68	117.3	118.6	0.28
814	112.3	-2.72	112.7	-2.65	0.34	117.7	-122.16	117.6	-122.2	0.11	114.3	118.14	114.7	118	0.39
850	123.6	-2.72	124	-2.65	0.35	126.5	-122.16	126.4	-122.2	0.11	125.7	118.14	126.2	118	0.47
816	123.5	-2.72	124	-2.65	0.42	126.5	-122.17	126.4	-122.2	0.1	125.7	118.14	126.1	118	0.41
818	123.4	-2.73	123.9	-2.65	0.43										
820	120.6	-2.78	121.2	-2.68	0.51										
822	120.2	-2.78	120.9	-2.68	0.59										
824	123	-2.91	123.4	-2.84	0.35	125.9	-122.34	125.8	-122.4	0.13	124.9	117.93	125.3	117.8	0.4
826						125.8	-122.34	125.7	-122.4	0.13					
828	123	-2.93	123.4	-2.86	0.35	125.8	-122.35	125.7	-122.4	0.12	124.8	117.91	125.3	117.8	0.45
854	121.9	-3.34	122.3	-3.27	0.35	124.7	-122.67	124.6	-122.7	0.1	123.2	117.48	123.6	117.3	0.45
856						124.7	-122.68	124.6	-122.7	0.09					
852	120.1	-4.05	120.5	-3.98	0.34	122.7	-123.21	122.7	-123.3	0.16	120.2	116.71	120.7	116.5	0.54
832	125.4	-4.05	125.7	-3.98	0.27	124.3	-123.21	124.2	-123.3	0.18	125.4	116.71	126	116.6	0.52
858	125.2	-4.14	125.5	-4.07	0.27	124	-123.29	124	-123.4	0.19	125.1	116.61	125.7	116.5	0.52
864	125.2	-4.14	125.5	-4.07	0.27										
834	124.9	-4.24	125.3	-4.18	0.34	123.8	-123.38	123.7	-123.5	0.22	124.7	116.49	125.3	116.3	0.59
860	124.9	-4.24	125.3	-4.18	0.34	123.8	-123.38	123.7	-123.5	0.22	124.6	116.5	125.2	116.3	0.6
836	124.8	-4.25	125.2	-4.18	0.35	123.7	-123.38	123.7	-123.5	0.21	124.6	116.5	125.2	116.4	0.52
840	124.8	-4.25	125.2	-4.18	0.35	123.7	-123.38	123.7	-123.5	0.21	124.6	116.5	125.2	116.4	0.52
862	124.8	-4.25	125.2	-4.18	0.35	123.7	-123.38	123.7	-123.5	0.21	124.6	116.5	125.2	116.4	0.52
838	124.8	-4.25	125.2	-4.18	0.35										
842	124.9	-4.25	125.3	-4.18	0.35	123.8	-123.38	123.7	-123.5	0.22	124.7	116.49	125.3	116.3	0.59
844	124.9	-4.27	125.3	-4.2	0.35	123.7	-123.41	123.7	-123.4	0.02	124.7	116.46	125.3	116.3	0.56
846	124.9	-4.32	125.3	-4.25	0.35	123.7	-123.45	123.7	-123.6	0.26	124.7	116.41	125.3	116.3	0.52
848	125	-4.32	125.3	-4.26	0.26	123.7	-123.46	123.7	-123.6	0.24	124.7	116.4	125.3	116.3	0.52
888	122	-4.81	122.4	-4.74	0.35	120.9	-123.99	120.8	-124.1	0.2	122.1	115.95	122.7	115.8	0.55
890	120	-4.73	120.3	-4.66	0.27	118.9	-124.01	118.8	-124.1	0.17	120	115.99	120.5	115.8	0.51

Figure 3.2 IEEE 34 BUS RADIAL DISTRIBUTION TEST RESULTS

The circuit which was used for this load flow comparison is the IEEE 34 Bus Radial Distribution Feeder [21]. A connection diagram of this circuit is shown in Figure 3.3. This feeder is characterized by:

1. Load Types

- a. Spot and distributed loads
- b. All wye connected
- c. All constant kW,kVAR

2. Line Types

- a. Three-phase overhead
- b. Single-phase overhead

3. Voltage Regulators - single phase regulators wye connected

4. Shunt Capacitors - balanced three-phase

5. In-line auto transformers 24.9 kV to 4.16 kV



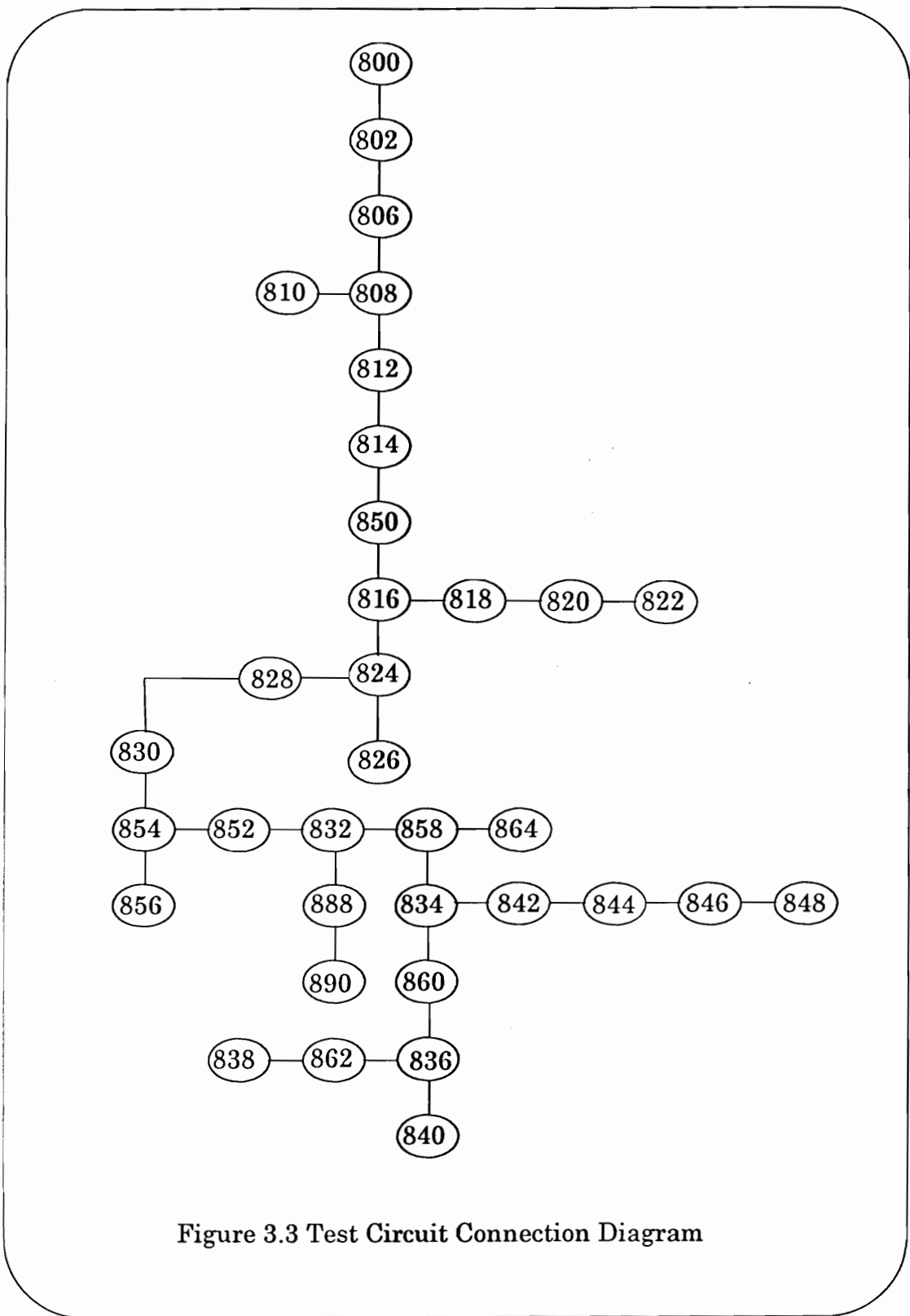


Figure 3.3 Test Circuit Connection Diagram

There are several modeling differences between the two load-flow algorithms. The RDAP voltage angles are referenced to Bus 800. The DEWorkstation voltage angles are referenced to the substation. This explains the difference in the results at the first bus in the circuits (Bus 800).

The loads in the RDAP are placed in the center of the transmission line. While the DEWorkstation places the loads at the end of the transmission line. The shunt charging reactance of the line is split in half by the RDAP model. One half of this reactance is placed at the start of the transmission line and the other half at the end of the line. The DEWorkstation places the total reactance at the far end of the line.

Despite these modeling differences, the percentage difference between the two algorithms is less than 0.6 percent. Some of this difference can be attributed to round-off error. DEWorkstation displays numbers to only two decimal places of accuracy.

## Chapter 4

### THE SWITCH COMPENSATION METHOD

The primary reason the Ladder Load-Flow method is competitive with other methods in terms of CPU time and memory usage is the limited use of matrix operation. In particular, the elimination of the need to form and invert a matrix dimensioned by the size of the system. The intent of the Switch Compensation Method is to use the information provided by the basic Ladder Method to extend the method to looped systems without the use of matrix operations.

#### 4.1 Theory

The ladder method iterates on a radial circuit structure. If the system is looped, it may be separated into radial circuits by inserting open switches into some of the transmission lines. At the open switches the effect of the remaining network may be simulated by a fictitious load current which makes the voltage across the switch equal to zero.

The load currents of all three phases at the switch point must be considered when the circuit is mutually coupled. Figure 4.1 represents the model of a single phase of two mutually coupled three phase transmission lines connected through a switch.

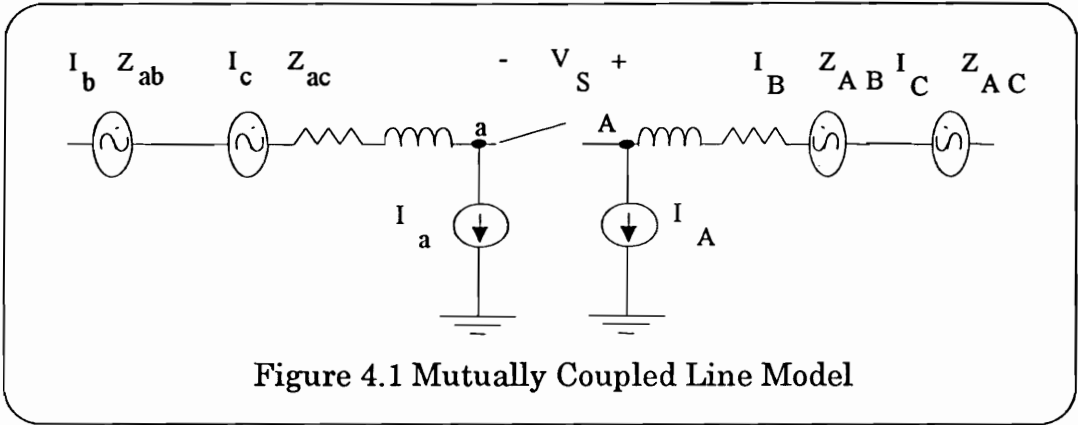


Figure 4.1 Mutually Coupled Line Model

It is desired to make the voltages on opposite sides of the switch equal to one another. Applying the principle of superposition this condition can be achieved by adding a change in voltage to each voltage at the switch.

$$V_{an} + \Delta V_{an} = V_{AN} + \Delta V_{AN} \quad (1)$$

The difference in the change in voltage is equal to the voltage across the switch.

$$\Delta V_{an} - \Delta V_{AN} = V_{AN} - V_{an} = V_S \quad (2)$$

The change in voltage at the switch point may be related to the change in current at the same point by the sensitivity matrix.

$$[\Delta V]_{ABC} = [S]_{ABC}[\Delta I]_{ABC} \quad (3)$$

$$[\Delta V_{abc}] = [S]_{abc}[\Delta I]_{abc} \quad (4)$$

The currents must be equal and opposite to correctly model the current flow through the closed switch. This also implies that the an incremental increase in current on one side of the switch is followed by an incremental decrease in current on the other side of the switch.

$$[\Delta I]_{abc} = -[\Delta I]_{ABC} \quad (5)$$

Combining the equations 3, 4, and 5 results in the following expression.

$$[\Delta V] = [V_S] = [V]_{ABC} - [V]_{abc} = ([S]_{abc} + [S]_{ABC})[\Delta I] \quad (6)$$

This equation relates the voltage at the switch points to the desired change in current

#### 4.2 Formation of Sensitivity Matrix

The implementation of Equation 6 requires the computation of the elements in the 3 -by-3 sensitivity matrix. The elements of the matrix relate a change in current to a corresponding change in voltage:

$$S_{xy} = \Delta V_x / \Delta I_y \quad (7)$$

If the system is linear (no mutual coupling) the sensitivity matrix will be the Thevenin Impedance Matrix. In a circuit which contains mutual coupling and non linear loads the sensitivity matrix elements represent the change in current to the change in voltage at a particular point in the circuit. These elements represent a linear approximation at a particular point of operation.

The voltage versus current graph of a linear resistor shown in Figure 4.2 illustrates the calculation of impedance in this manner. Notice that the value of the resistor may be determined by the ratio of the voltage and current at a point on the line. This is Ohm's law. The same result can be determined by computing the slope of the line. This is Thevenin's theorem.

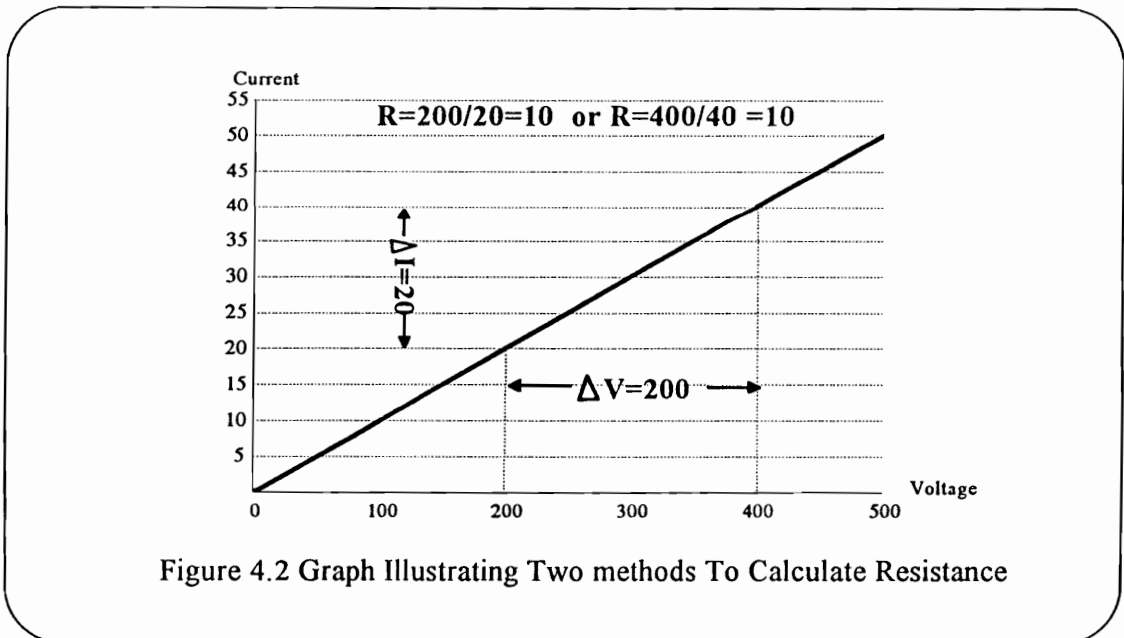
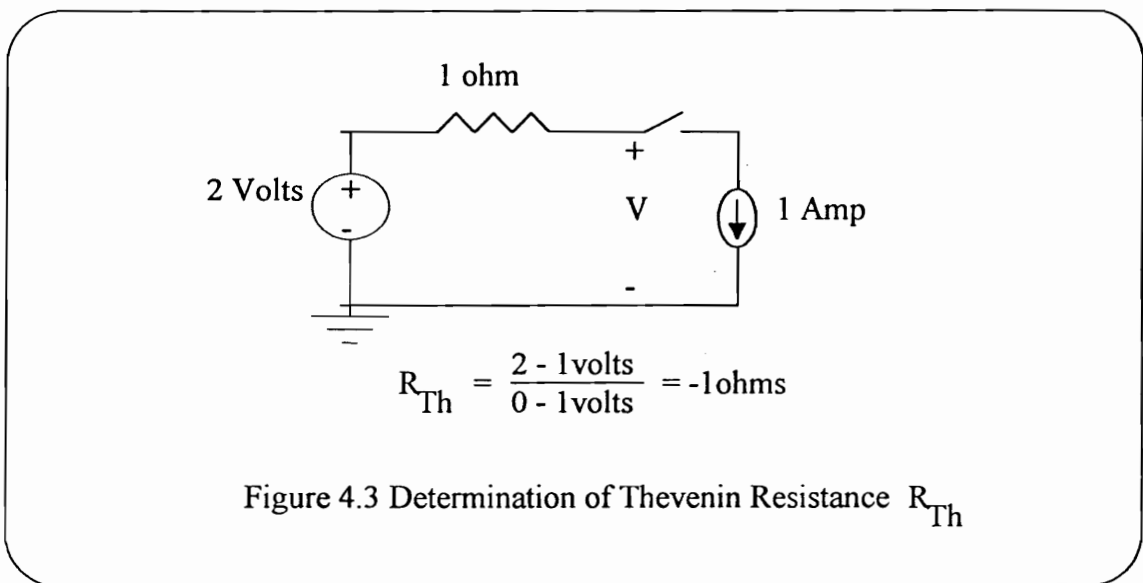


Figure 4.2 Graph Illustrating Two methods To Calculate Resistance

The determination of the Thevenin impedance as the  $\Delta V/\Delta I$  will result in the correct magnitude of the Thevenin impedance of any linear network.

The sign of the impedance may be reversed. Consider the example shown in Figure 4.3. The voltage  $V$  will be 2 volts when the switch is open. The voltage will then drop to 1 volt when the switch is closed and one Amp of load is applied. The calculation of the impedance using the change in voltage and change in current at the terminals would result in a negative value.



The resulting sign of the Thevenin Impedance can be corrected by assuming the resulting impedance must have a positive real value. This implies that the impedances of the network are passive i.e., the energy delivered to any impedance in the network is nonnegative for any excitation. This is certainly the case for most distribution networks.

Previous work in this area has employed Ohm's law to calculate the equivalent impedance of the network. The use of the injection of one amp at

the co-tree component to determine the impedance of the network is an example of the application of Ohm's law. A current is injected, the ratio of resulting voltage and the injected current is used to determine the impedance of the circuit. The Switch Compensation algorithm uses the change in voltage divided by the change in current to determine the equivalent impedance of the circuit.

This relationship between a change in current and a change in voltage can easily be determined from the simulated connection scheme previously described. Consider the following example. At a connection point a desired change in current is determined using equation 6. This current is added to any existing current at the connection point to simulate the closing of the switch. The radial circuit is solved using the ladder method. This will result in a new approximation to the voltage at the connection point. This resulting voltage subtracted from the initial voltage at the connection point represents a change in voltage. The corresponding current change is added to the injected currents to simulate the closing of the switch.

Thus the sensitivity matrix elements are computed as part of the Ladder-Load Flow algorithm. The information from one iteration is used to determine the sensitivity matrix elements for the next iteration. The only other requirement is some way to begin this iterative process.



### 4.3 Current Injection

In order to initialize the iterative process the sensitivity matrix must be calculated. Traditionally this has been done by simply summing the impedance matrices back to source. This approach is limited to systems which contain components that can be modeled as constant series impedances. Therefore, the current injection method to determine the sensitivity matrix has been used in order to be able to model different types of power system components. This modification to the ladder method can be implemented in computer code as a separate module, thus avoiding revisions to working code.

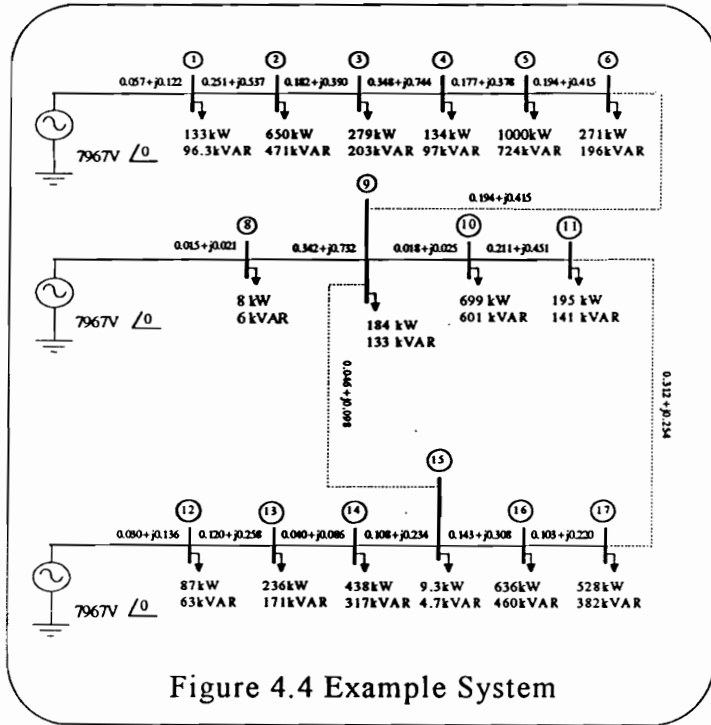
### 4.4 Voltage Correction

The principle of superposition may also be applied to determine a voltage correction at each co-tree component. The desired change in the co-tree currents is determined from solution of Equation 6. These currents may then be substituted into Equations 3 and 4 to obtain the corresponding change in voltage at the co-tree components.

The rate of convergence of a looped system can be increased by applying this correction at each co-tree component. This is not an unexpected result considering this was the basic algorithm of the original ladder method.

This can be illustrated with aid of the example system shown in Figure 4.4. The transmission lines which are shown as dashed lines in the Figure

are selected as co-tree components. The loads are converted to constant impedances by computing the ratio of the voltage and current at each load bus at the end of each iteration. The Thevenin impedance at each connection point is then determined.



The dependence of the looped system on the voltage correction at each co-tree component is illustrated for three different cases in Figure 4.5. In each case the percent convergence shown in Figure 4.5 is computed as follows:

$$\frac{|\text{Voltage at Iteration} - \text{True Voltage}|}{\text{True Voltage}} \times 100$$

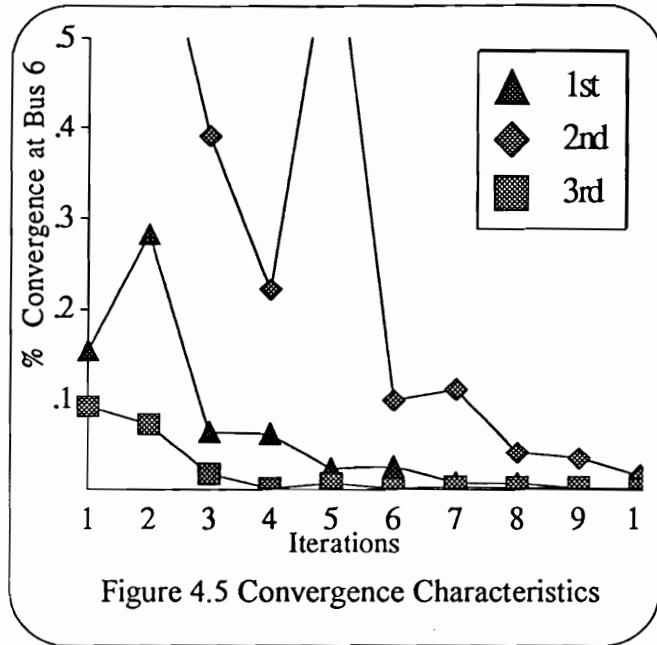


Figure 4.5 Convergence Characteristics

In each of the cases, the algorithm was considered to have converged when the difference was less than 0.02 percent.

The 1st case in Figure 4.5 illustrates the convergence of the iterative scheme with voltage correction at each co-tree component. These results were recorded for bus number 6 in the system. The results are typical of every co-tree component in the system. Convergence is achieved in 8 iterations.

The 2nd case illustrates the convergence characteristics of the method without voltage correction. The algorithm takes over 40 iterations to converge.

The determination of the Thevenin impedance at a bus will be accurate only if the co-tree components are not removed from the system. The method will converge as shown in the 3rd case of Figure 4.5 if this is considered. For instance, once the current flow is approximated for the co-tree component

between buses 6 and 9 , the Thevenin impedance at bus 9 may be corrected before current at the other co-tree component is determined. Conversely, the resulting impedances and currents at other connection points will also affect the approximation of the line current between buses 6 and 9. It should also be mentioned that case 3 requires considerably more computations than case 1.

Several conclusions can be induced about the behavior of the algorithm from this illustration. The first and most obvious conclusion is that voltage correction increase the rate of convergence. Figure 4.5 illustrates how the calculated co-tree current can be in the wrong direction when voltage correction is not applied. This behavior can be explained.

It is important to recognize that the co-tree current is calculated from the voltages at the connecting buses. Consequently, if the voltages at the connecting buses are vastly different from the true values, the co-tree current can also be expected to be in error. This may be the case when one co-tree component serves as a connection point to more than two radial circuit. This is the case for bus 9 in Figure 4.5.

The second case illustrated in Figure 4.5 corresponds to the compensation method which has gained some popularity in the literature. The co-tree currents are determined without considering the effect these currents will have on the voltage at co-tree buses. The calculation of each

co-tree current proceeds without changing the voltage at the co-tree buses. Thus the co-tree currents are in error for the first several iterations.

The first case in Figure 4.5 illustrates the convergence characteristics of the algorithm when the change in voltage at each co-tree bus is partially considered. The co-tree current between buses 6 and 9 is used to determine a new value of voltage at each bus. This "new" bus 9 voltage is used to determine the co-tree current between bus 9 and 15. Each co-tree bus is processed once in this manner.

The third case in Figure 4.5 illustrates the convergence characteristics of the algorithm when the change in voltage at each co-tree bus is considered more fully. This case makes two passes over the co-tree buses. The first pass is identical to case one. The second pass repeats the same processing using the voltages and currents prescribed by the first pass. The co-tree current between bus 9 and 15 will also result in a change in voltage at bus 9. This change in voltage will effect the calculation of the co-tree current at bus 6. This nullifies the order in which co-tree components are processed in the first pass.

#### **4.5 Implementation**

The Switch Compensation method is implemented based on the results of this experiment. The resulting scheme is a predictor/corrector type of iterative scheme.

1. Converge each radial circuit using the Ladder Load-Flow Method.
2. Inject one amp into one phase of the radial circuits at each connection point.
3. Converge each radial circuit using the Ladder Load-Flow Method.
4. Calculate the Sensitivity Matrix elements as the change in voltage divided by the one amp current injection. (Repeat steps 2 through 4 for all phases )
5. Calculate for a phase the change current required at each connection point.
6. Calculate the expected change in voltage for all phases and add this to the connection points.
7. Converge each of the connecting circuits.
8. Update the Sensitivity Matrix elements.
9. Repeat steps 5 through 8 for all phases and until the method converges.

Figure 4.6 is a flow chart of initialization of the sensitivity matrix (steps 1 through 4). Figure 4.7 is a flow chart of the iterative predictor/corrector algorithm. A more detailed explanation of the algorithm can be found in Appendix B.

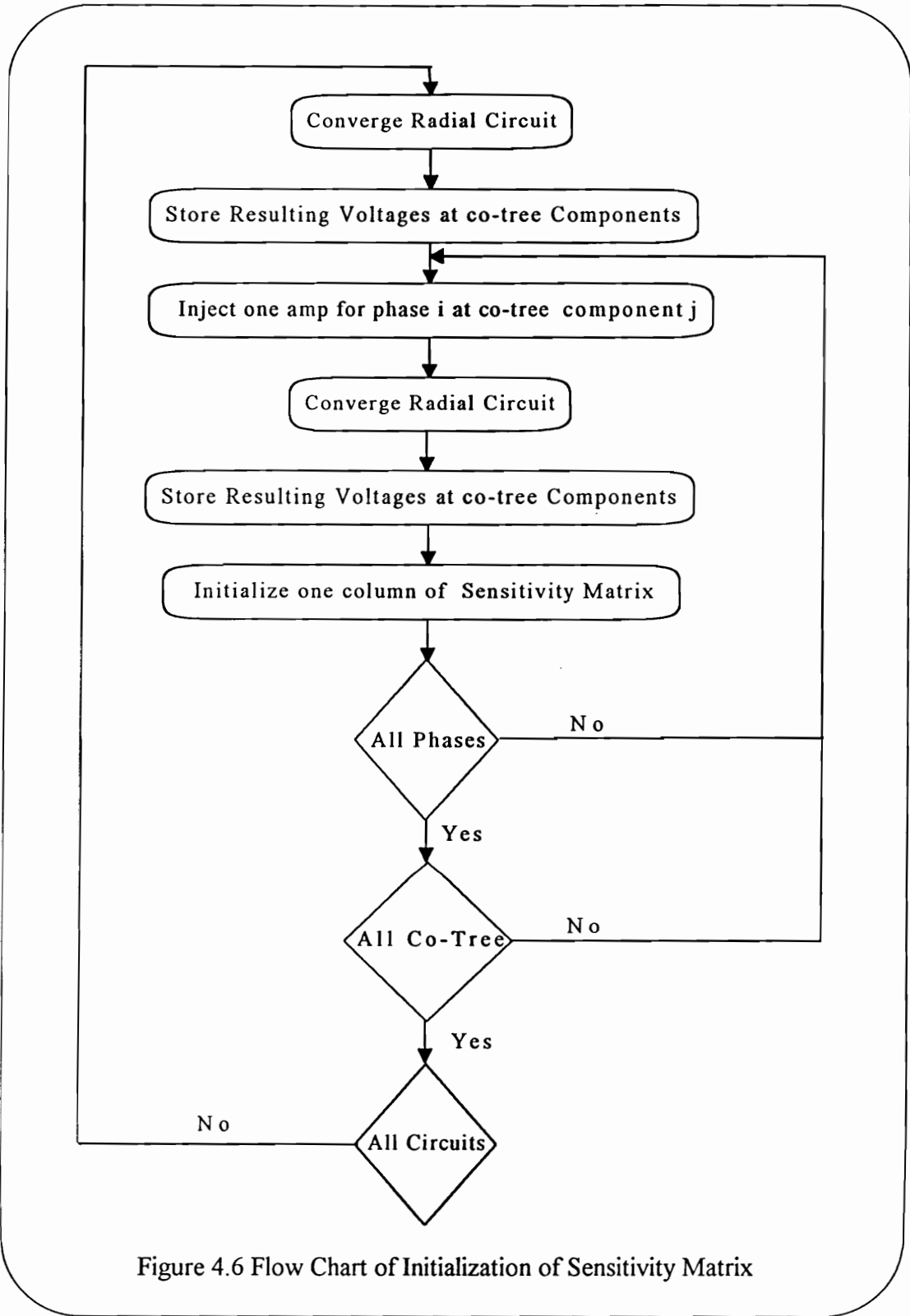


Figure 4.6 Flow Chart of Initialization of Sensitivity Matrix

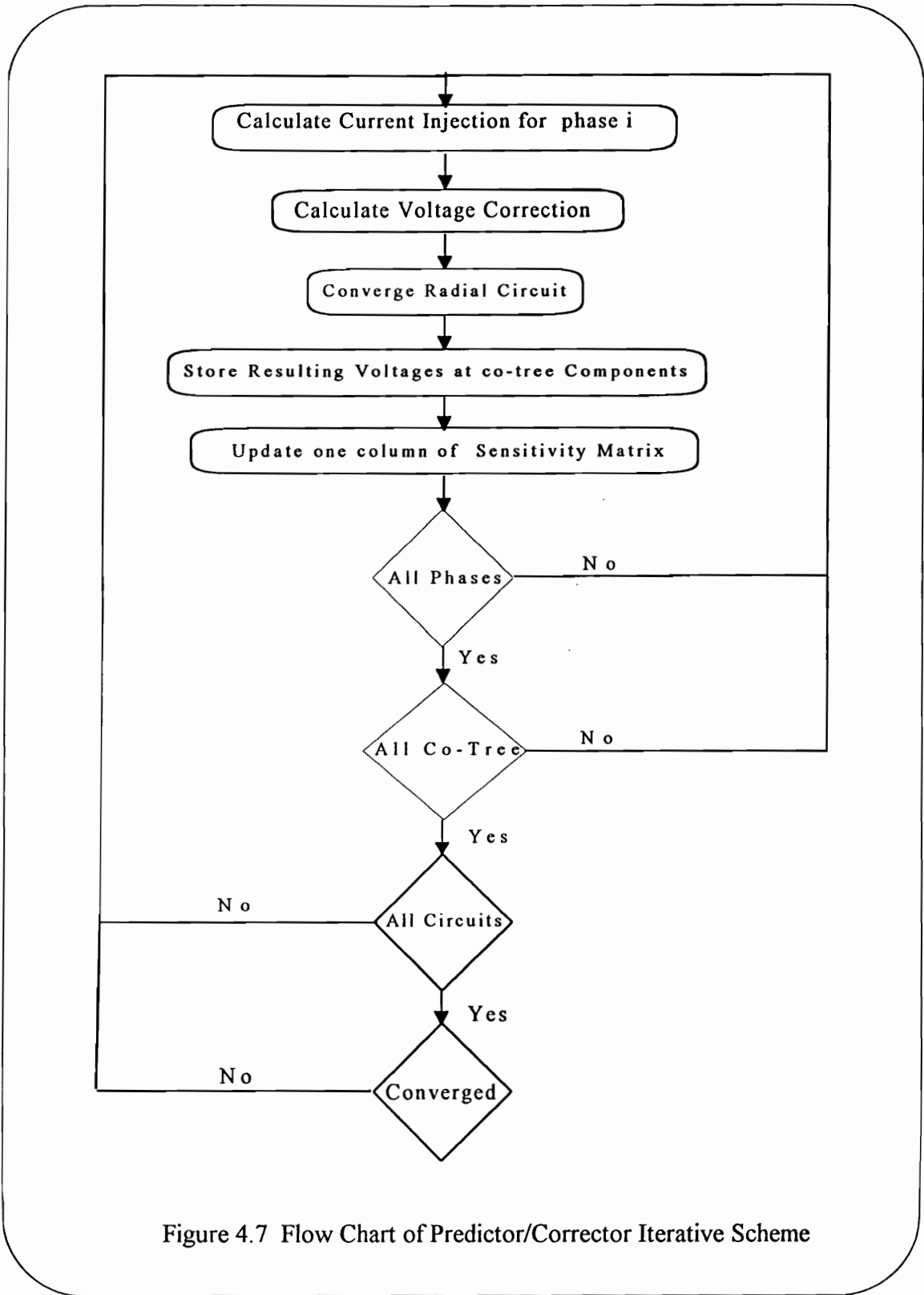


Figure 4.7 Flow Chart of Predictor/Corrector Iterative Scheme



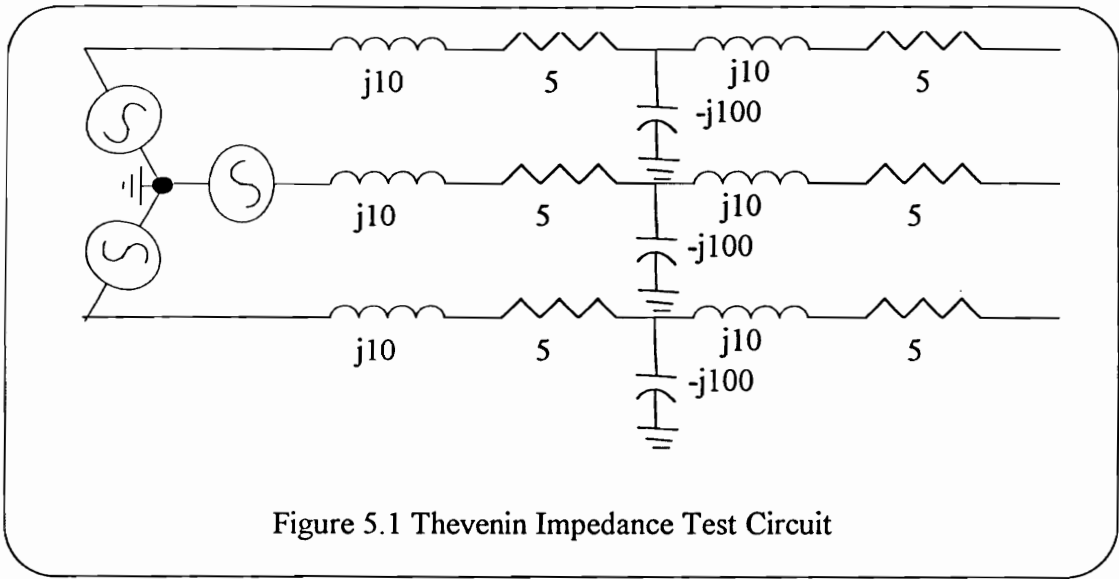
## **Chapter 5**

### **CALCULATION EXAMPLE AND RESULTS**

A distribution network was established in the DEWorkstation to validate the Switch Compensation Method. This network contains mutually coupled lines and is therefore inherently an unbalanced system. The feasibility of solving distribution networks which contain automatic control devices is also evaluated.

#### **5.1 Determination of Sensitivity Matrix**

Several test circuits were employed to test the ability of the algorithm to correctly initialize and update the elements of the sensitivity matrix. The circuit shown in Figure 5.1 illustrates the ability of the algorithm to determine the Thevenin Impedance of a circuit. The circuit shown in Figure 5.1 does not contain any mutually coupled elements. The off-diagonal terms of the sensitivity matrix are zero for this case.



The values shown in Figure 5.1 are all impedances in ohms. The algorithm initialized the diagonal impedances in the sensitivity matrix to the correct values of  $11.15 + j20.77$  ohms. However, off-diagonal terms were observed when the algorithm updated the sensitivity matrix. These terms grew in magnitude for each update of the matrix.

The algorithm checks the magnitude of both the  $\Delta V$  and the  $\Delta I$  before updating the corresponding sensitivity matrix element. These checks serve two purposes. First the checks are done to avoid under and overflow in the calculation. Second the checks are done to determine if the magnitude of the change is smaller than the convergence tolerance of the Ladder Load-Flow.

Initially, the threshold of this magnitude change was set equal to the convergence tolerance of the Ladder Load-Flow method. For the decoupled test case shown in Figure 5.1 off-diagonal terms of approximately 1.5 were observed after four updates of the matrix. The algorithm converged in three

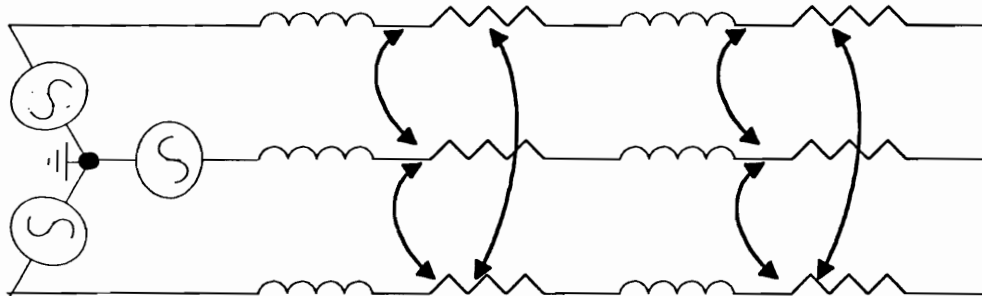
iterations. These terms started at a much smaller value but grew at each update of the matrix.

These errors resulted from small differences in the converged solution at each iteration. A change in phase "A" current should not result in a change in phase "B" voltage when the system is decoupled. However, each time the radial circuits are solved using the Ladder-Method a small difference is observed in the solution.

These small oscillations in the solution values are inherit in any iterative process which is halted and started again. In the absolute sense, the solution changes after each iteration. Consequently, a small change in phase "B" voltage was observed for a corresponding change in phase "A" current for one iteration. This incorrectly, added an off-diagonal term to the sensitivity matrix.

This effect grows as the algorithm nears convergence. The change in current is expected to decrease as the algorithm nears convergence. This results in division by a small current value. Thus a small error in the voltage is amplified by dividing by a progressively smaller current value for each iteration.

This was corrected by raising the threshold at which updates were performed to 10 times the convergence tolerance. This also decreased the solution time for the test network in the next section by about 7 seconds.



$$\mathbf{Z} = \begin{bmatrix} 0.0205 & j0.02480 & 0.0036 & j0.0099 & 0.0036 & j0.0081 \\ 0.0036 & j0.0099 & 0.00208 & j0.0243 & 0.0037 & j0.0092 \\ 0.0036 & j0.0081 & 0.0037 & j0.0092 & 0.0206 & j0.0246 \end{bmatrix}$$

Figure 5.2 Mutually Coupled Impedance Test Circuit

The ability of the algorithm to determine the sensitivity matrix in the presence of mutually coupled components was also tested. The transmission line sections shown in Figure 5.2 are mutually coupled. The resulting impedance matrix for this case is also shown in Figure 5.2. The circuit consists of two three phase sections which are connected in series. The shunt admittance are smaller than 0.00001 in magnitude.

The resulting sensitivity matrix under these conditions is approximately the sum of the two impedances matrixes for each line section. Since the impedance matrixes are equal for each line section the result is twice value of one sensitivity matrix. The results obtained from the program are shown in Figure 5.3 .

0.041050 j0.0497	0.007296 j0.01986	0.007148 j0.01629
0.007296 j0.01986	0.041550 j0.04861	0.007397 j0.01832
0.007148 j0.01629	0.007397 j0.01832	0.041240 j0.04928
Figure 5.3 Resultant Sensitivity Matrix		

## 5.2 Test Network

The distribution network shown in Figure 5.4 was used as a test network for the Switch Compensation Method. The co-tree components in the network are the switches. The network consists of six loops, 35 nodes and four feeders. The cables, transformers, and substation voltage represent typical values for a distribution network [22].

The nominal voltage at the substation is 7.967 kV line-to-neutral. The secondary grid is supplied through the network transformers which are Y-Y connected. The rating of the transformers is 200kVA. The winding losses are 1% and the leakage reactance is 5%. The nominal secondary grid voltage is 125 volts line-to-neutral. Each secondary grid (loop ) corresponds to a city block ( 6 blocks to a mile).

The cables are single-conductor cables with rubber and synthetic insulation without lead sheath protection. The conductor cross section is 4/0 circular mils with 19 strands. The current rating of this 4/0 cable is 480 Amperes. The cables are positioned in a vertical fashion under ground. The

first conductor phase C is positioned 30 inches below the ground plane. The remaining conductors are spaced 3 inches apart. The phase positions in the order they appear under ground are C, B, A, N. This configuration results in an unequal mutual coupling effect between phases.

The mutual coupling effect is modeled using an equivalent 3-by-3 impedance matrix [4]. A typical impedance matrix for a secondary grid cable is shown below in Figure 5.5. The charging current of the cable is modeled using a shunt admittance matrix. A typical shunt admittance matrix is also shown in Figure 5.5.

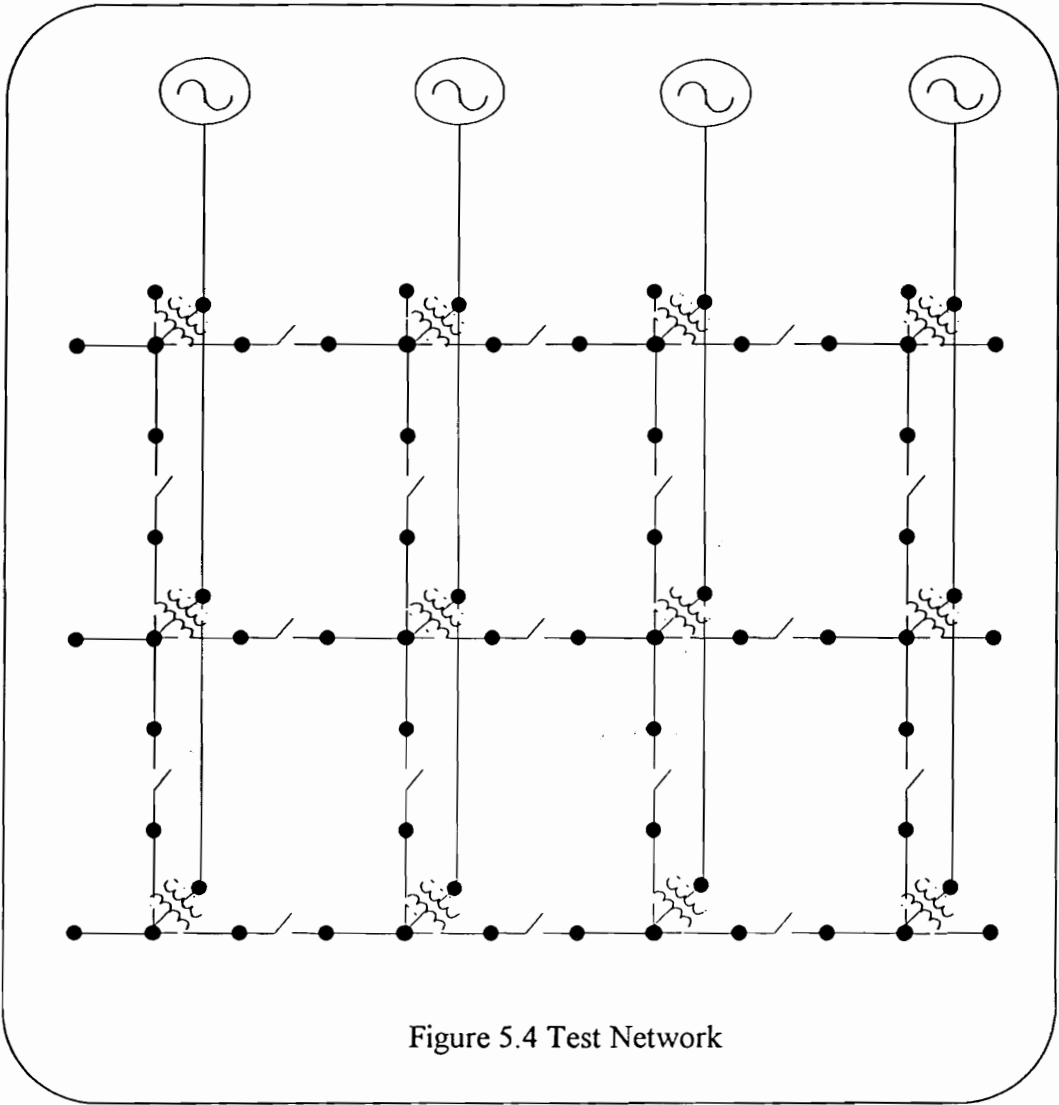


Figure 5.4 Test Network

Phase	Series Impedance						Shunt Admittance		
	A R	A jX	B R	B jX	C R	C jX	A jB	B jB	C jB
A	0.0365,	0.0464	0.0127,	0.0262	0.0122,	0.0226	8.3e-7	-2.8e-7	-9.2e-8
B	0.0127,	0.0262	0.0349,	0.0554	0.0115,	0.0331	-2.8e-7	8.4e-7	-3.1e-7
C	0.0122,	0.0226	0.0115,	0.0331	0.0340,	0.0604	-9.2e-8	-3.1e-7	7.1e-7

**Figure 5.5 Typical Cable Impedance and Admittance values**

The winding leakage reactance and resistance are represented for each transformer. Transformers are modeled as three separate units. The concepts of amp-turn balance and Kirchhoff's law are employed to model the transformers under unbalanced loading conditions. This method is similar to the method of symmetrical components [23]. This method can be used to correctly model the 30 degree phase shift of Wye-Delta transformer configurations. See Appendix A for a detailed explanation of this modeling technique.

Three different load models are available : constant power, constant current and voltage dependent current. The constant power load model determines a load current at the bus such that the specified real and reactive power are maintained. Given a specified power and a voltage at the bus, the load current is determined as indicated below:

$$I = (P - jQ)/V^*$$



The magnitude and phase of the current is maintained constant in the constant current model. The magnitude of the current is determined in the same manner as the constant power model except for the load calculation the voltage is assumed to be a constant at the bus. Thus, it is necessary to determine this value only once at the start of the load flow iterations. As the phase angle of the voltage changes at each iteration the phase of the current is also adjusted to maintain the specified phase relationship.

In the voltage dependent current model modifies the constant current by allowing the load current magnitude to vary with voltage magnitude. This dependency is assumed to be a linear function. The percent difference in voltage magnitude from the nominal value is determined . This difference is then multiplied by a voltage dependency factor. The resulting product is subtracted from the nominal current magnitude. This voltage dependency factor is specified by the user and may be either positive or negative. A positive value simulates a load which behaves like a constant impedance. While a negative value behaves similar to a constant power load.

All the loads are attached to the secondary network. The loads are connected at the end of each of the transmission lines. Figure 5.6 shows the loading of the network. The loads are initially balanced. The solution will then reflect the mutual coupling effect.

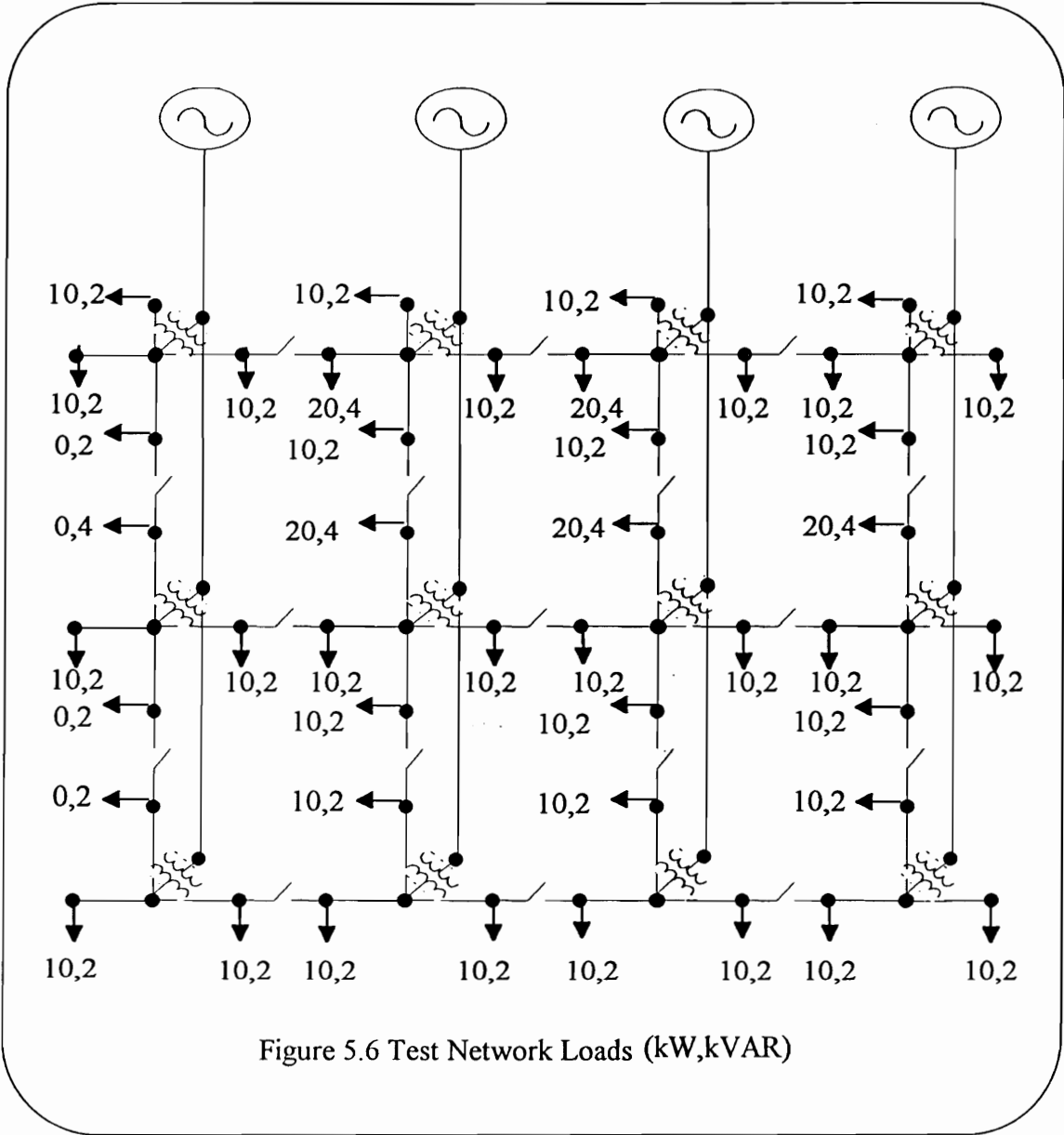


Figure 5.6 Test Network Loads (kW,kVAR)

### 5.3 Implementation

The Switch Compensation Method is implemented as indicated in Section 4.5. The method begins by converging each of the radial circuits using

the Ladder-Method. When the switches are inserted into the network, the executive attaches them with a feeder path pointer to a radial circuit. The connection between radial circuits is achieved by co-tree pointers.

The radial circuit which is used to build the network is shown in Figure 5.7. The horizontally oriented switches represent connections between radial circuits. The vertically oriented switches are attached to the same circuit. The algorithm treats both the horizontal and vertical switches as co-tree components.

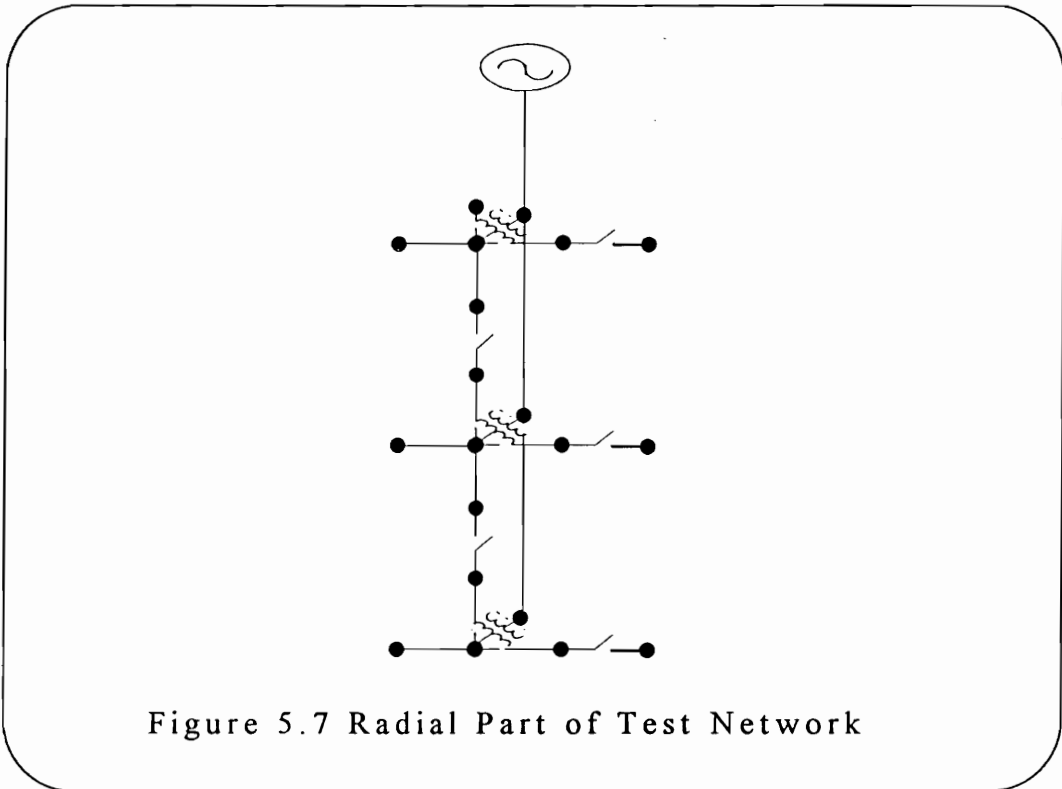


Figure 5.7 Radial Part of Test Network

The next step is to initialize the sensitivity matrix. This is accomplished by injecting a current of one amp into both components which border on a connection point. The radial circuits are again solved using the basic Ladder-Method. The elements in the sensitivity matrix may then be determined by taking the resulting difference in voltage and current from each converged solution.

All the voltages and currents involved in this calculation are complex numbers. The current that is injected is in phase with the voltage resulting from the first radial solution. In theory, the injected current could have any magnitude and angle. From experimentation selection of a current which is in phase with the voltage reduces the number of iterations required to achieve a solution.

The determination of all the sensitivity matrix elements requires that currents are injected for one phase at a time. Since components in the circuit are mutually coupled a current change in one phase is expected to result in a change in voltage in a different phase. A given phase current injection will determine one column of the sensitivity matrix.

Once the sensitivity matrix is initialized, the next step is to approximate the change in current at the connection point required to force the voltage difference across the connection point to zero. The calculation of this change in current proceeds as illustrated in Section 4.1. The resulting

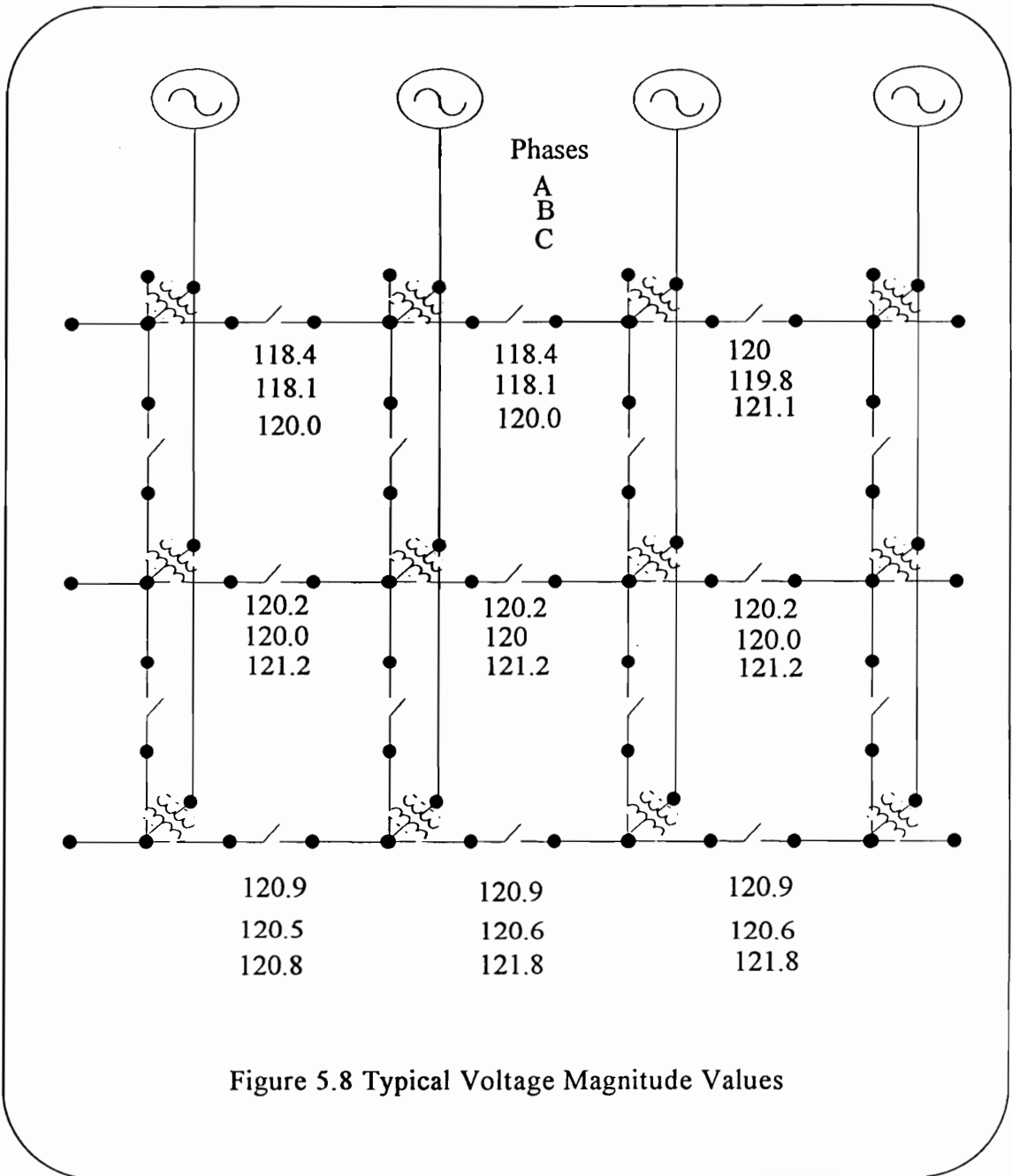
change in current is added to the one amp current injection on one of the components at the border of a connection point. This current is then negated and placed on the other component which borders on the connection point. This keeps the injected currents at the connection point equal and opposite.

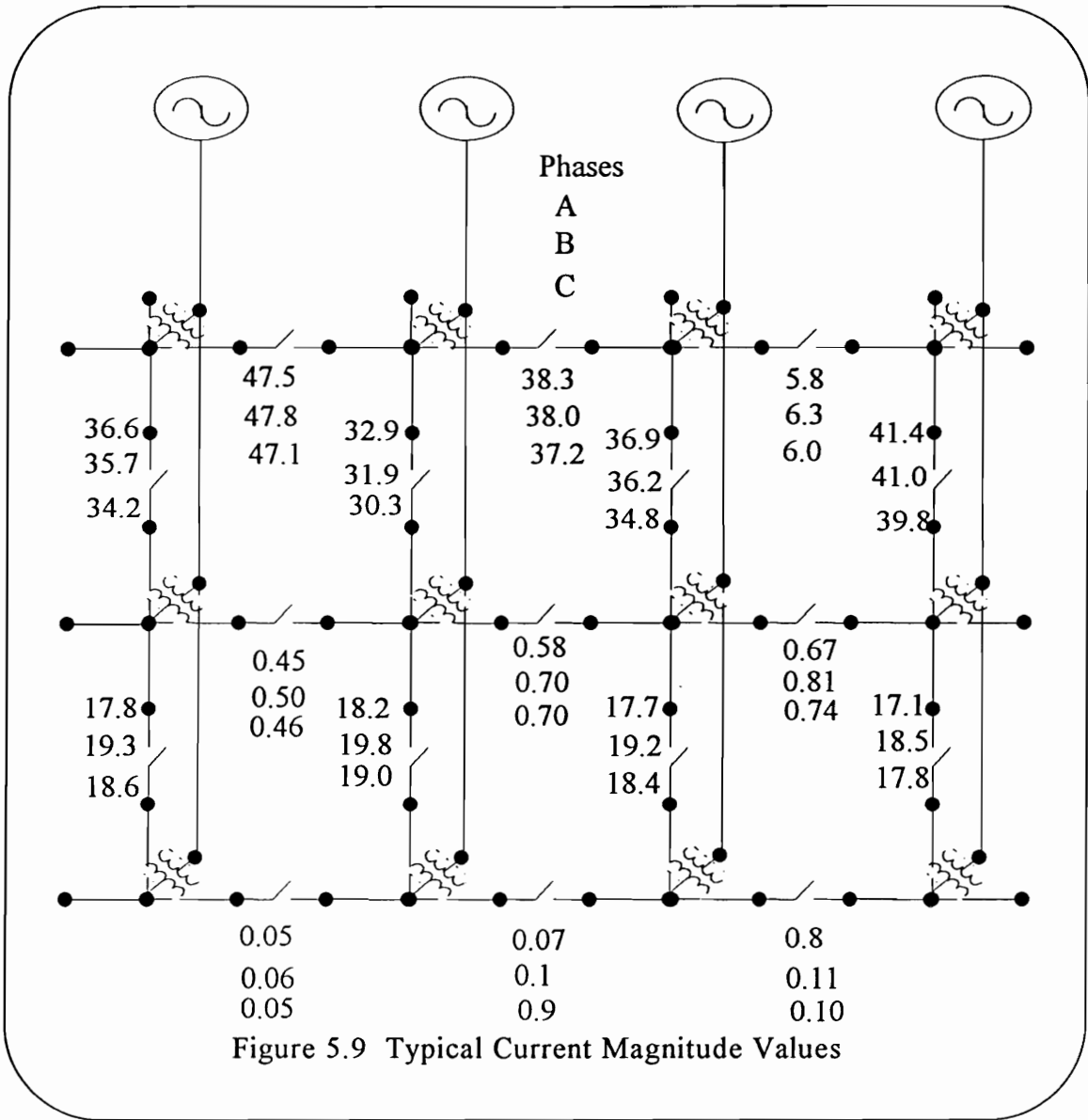
The change in current is determined one phase at a time. This allows the sensitivity matrix to be updated as described in Section 4.2. The voltage is also corrected as described in Section 4.4 after each calculation of the change in current.

Each of the radial circuits which are connected are then solved with the Ladder-Method. This process is repeated for each connection point until the voltage across each connection point is zero.

### 5.3 Results

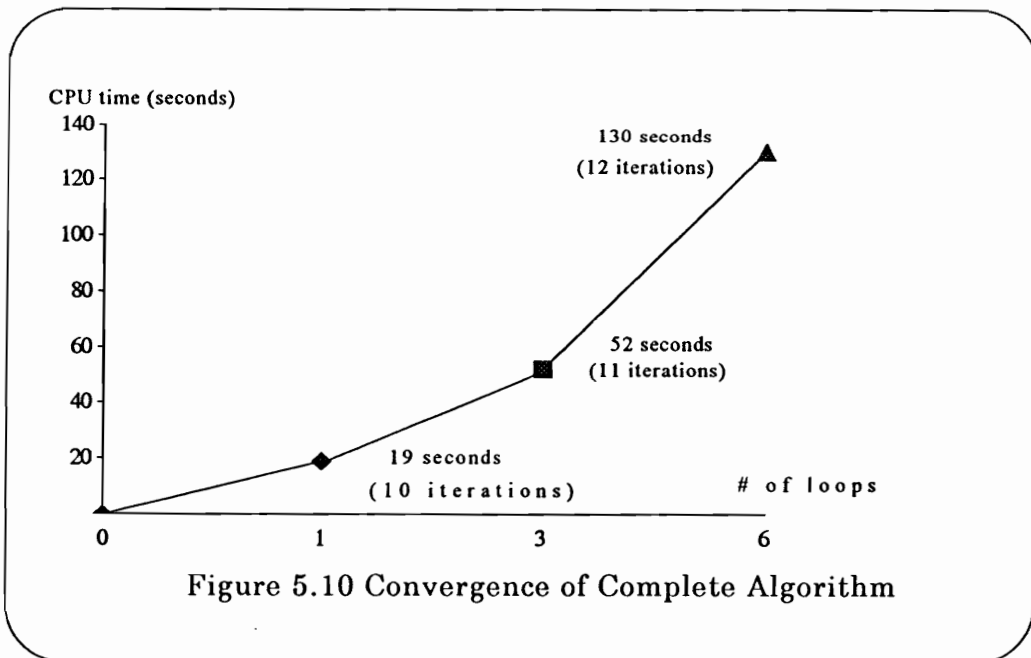
Figure 5.8 shows the resulting voltage magnitudes at the switch points. The results are shown in volts. The resulting current magnitudes are shown in Figure 5.9. The nominal voltage is 125 volts in the secondary network. The resulting currents through the switches vary considerably in magnitude in the network. This tests the algorithm's ability to converge on both large and small current injections at the switch points.





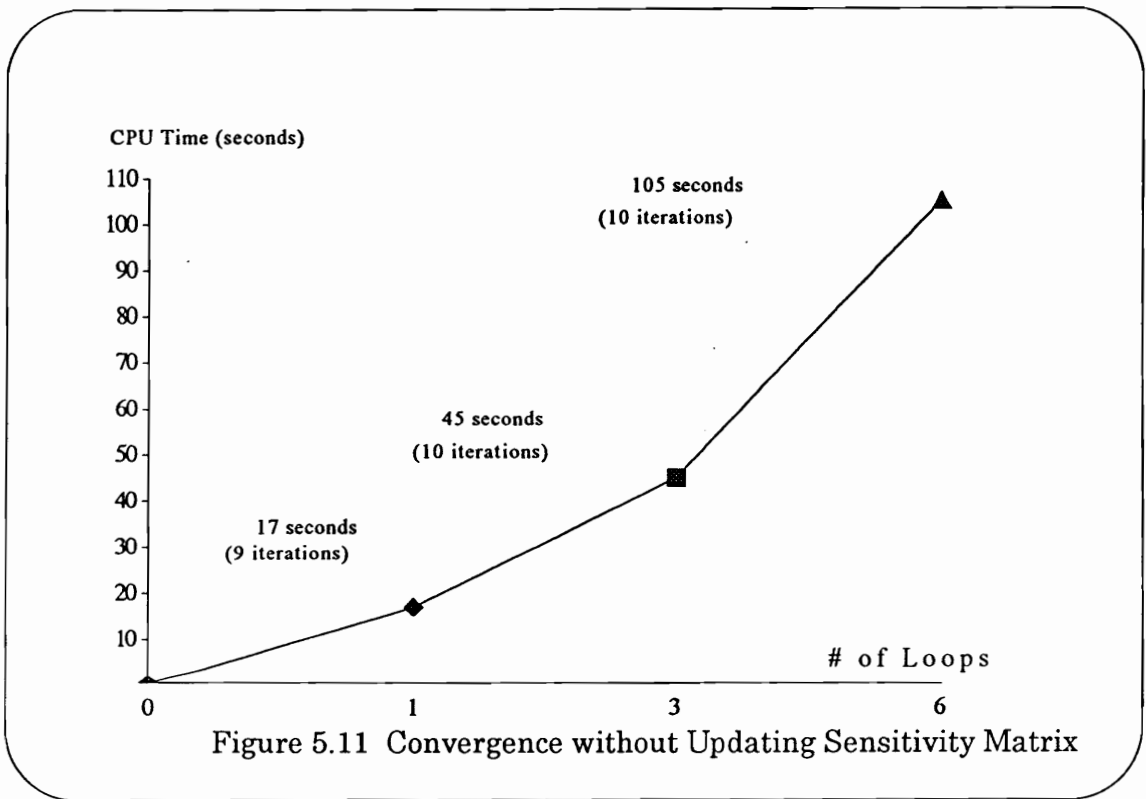
The convergence characteristics of the proposed algorithm were also investigated. The number of iterations required to achieve convergence and CPU time is largely a function of the number of loops in the network. Each iteration of this program corresponds to one change in all the co-tree currents in the network. The CPU time is the solution time seen by the user. This time

includes calculation of impedances and loads and performing data checks. The vast majority of the solution time is consumed in two routines. Approximately 10 percent of the total solution time is consumed converging the initial radial circuits and initializing the sensitivity matrix elements. The majority of the remaining time is spent in performing current corrections at the co-tree components and performing system iterations on the network. A flat start was used in all the cases. Convergence was achieved when the change in voltage magnitude and angle was less than 0.000001. Figure 5.10 illustrates the convergence characteristics of algorithm as it was originally proposed. The loads were modeled as constant power loads for the results shown in Figure 5.10.





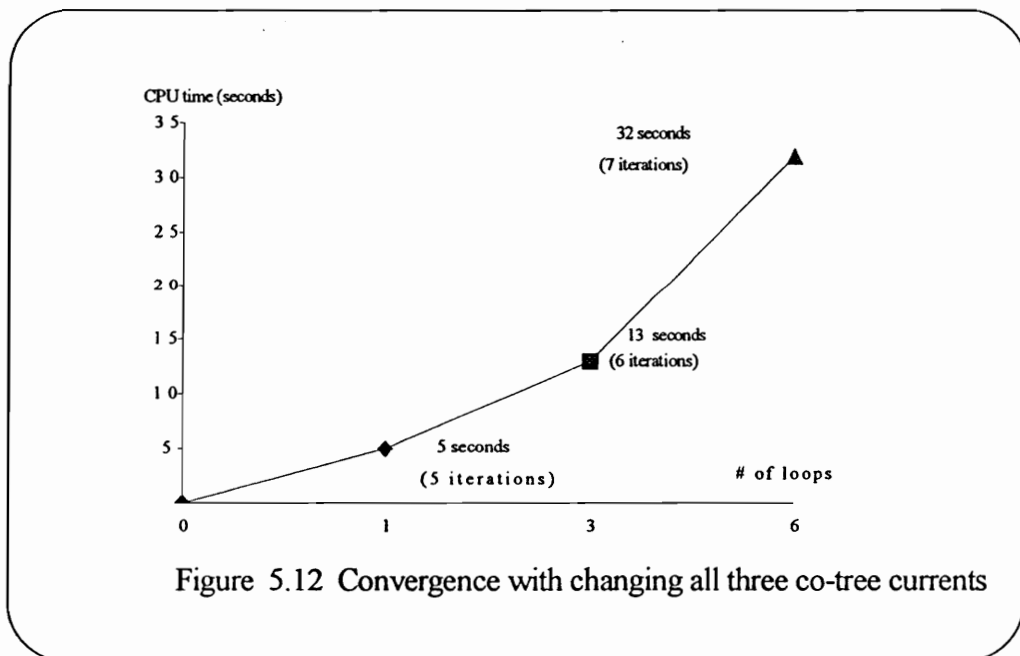
Several experiments with the algorithm indicated that convergence could be achieved for the test network without updating the sensitivity matrix elements. The convergence characteristics without updating the sensitivity matrix elements after each iteration are shown in Figure 5.11



A similar experiment was performed with and without voltage correction. Three different networks were solved first using voltage correction and then without voltage correction. The result of this experiment indicated that the absence or presence of voltage correction had little effect on CPU time and virtually no effect on number of iterations required to achieve a solution.

A difference of 1 second in CPU time was observed on the 6 loop network. No difference was observed on the 1 and 2 looped networks.

The intent of changing only one of the three phase co-tree currents for an iteration was to allow the sensitivity matrix to be updated in conjunction with currents. Since updating the sensitivity matrix elements in this manner actually increased the solution time, the possibility of changing all three co-tree currents in an iteration was also investigated. Figure 5.12 indicates the results of changing all three co-tree currents. The time to initialize the sensitivity matrix is approximately one-third of the total CPU time. The solution time is reduced by approximately one third.



The results of these three tests are summarized in Figure 5.13. The difference in solution time between the first two cases of  $25 = 130 - 105$  seconds

represents the amount of time which is spent in calculating the sensitivity matrix elements for each iteration. The difference in solution time for the last two cases  $73 = 105 - 32$  seconds is the increase CPU time between allowing one phase current to change per iteration and allowing all three co-tree currents to be changed per iteration. This explains why the difference is roughly three times.

# Loops	Complete Algorithm	w/o Sensitivity Update	Change three currents
1	19 seconds	17 seconds	5 seconds
3	51 seconds	45 seconds	13 seconds
6	125 seconds	105 seconds	32 seconds

Figure 5.13 Test Case Comparisons

The effect of adding control devices to the network was also investigated. Some of the transformers in the circuit were changed to tap changing under load (TCUL) transformers and some controlled capacitors were also added to the circuit as shown in Figure 5.14. The transformers controllers were set to control the voltage in the secondary network to 125 volts plus or minus 2 volts. The capacitors controllers were set to control the power factor at the substations to 0.99 plus or minus 0.1.

The four controlled capacitors were added individually to the circuit. After each addition the program was executed without updating the

sensitivity matrix. The addition of these four controlled capacitors did not alter the performance of the algorithm significantly.

The 12 transformers were changed to TCUL type transformers individually. After each change the program was executed without updating the sensitivity matrix. When one of the transformers furthest from the source was changed to a TCUL transformer the program was unable to reach a solution. The program was changed to update the sensitivity matrix. After this change the program reached a solution in 13 iterations and 135 seconds.

The resulting voltages magnitudes are shown in Figure 5.15. The resulting current magnitudes and power factor on each of the feeders is shown in Figure 5.16. Some of the resulting power factors not within the specified control bandwidth. The controlled capacitors are at their upper limit for each of these cases.

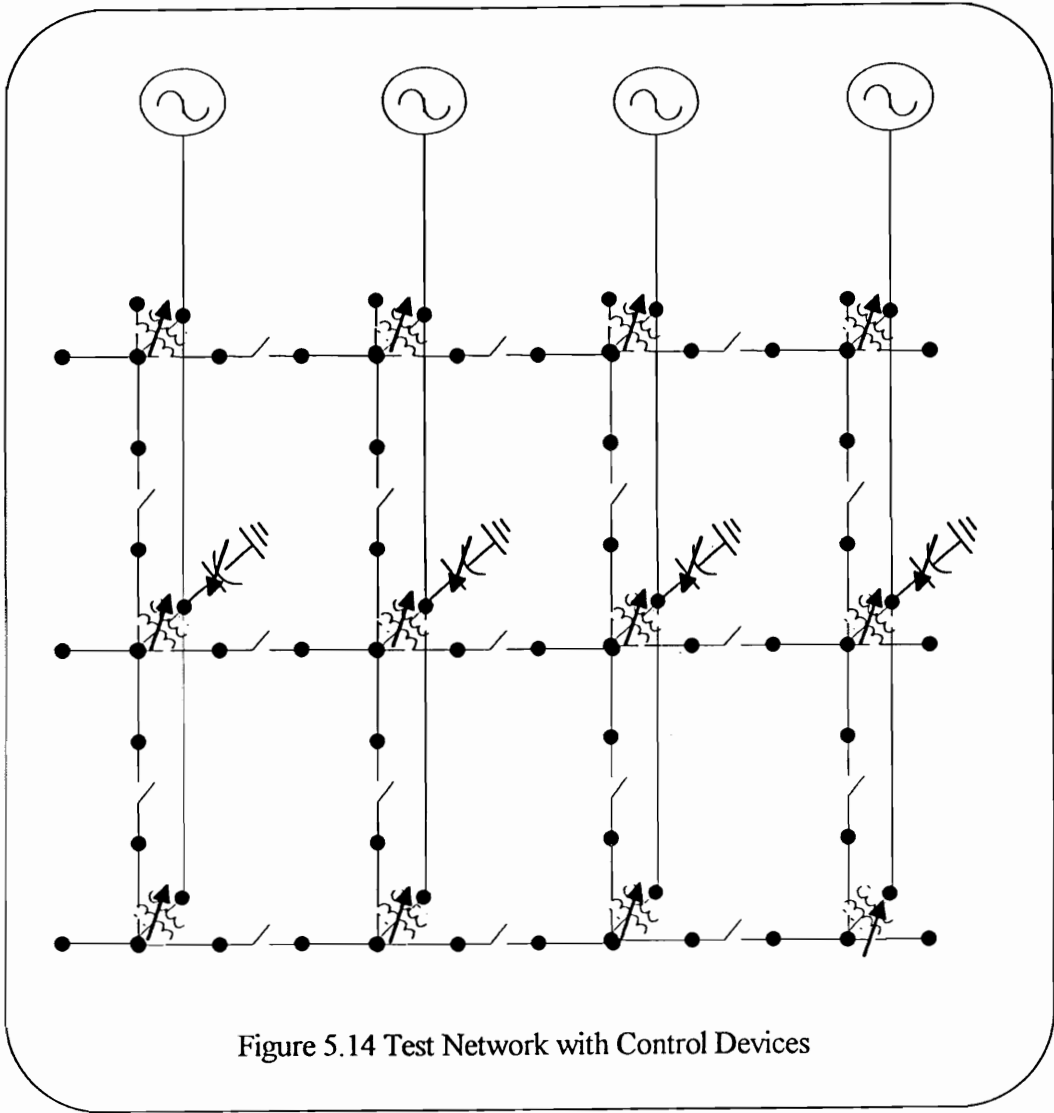


Figure 5.14 Test Network with Control Devices

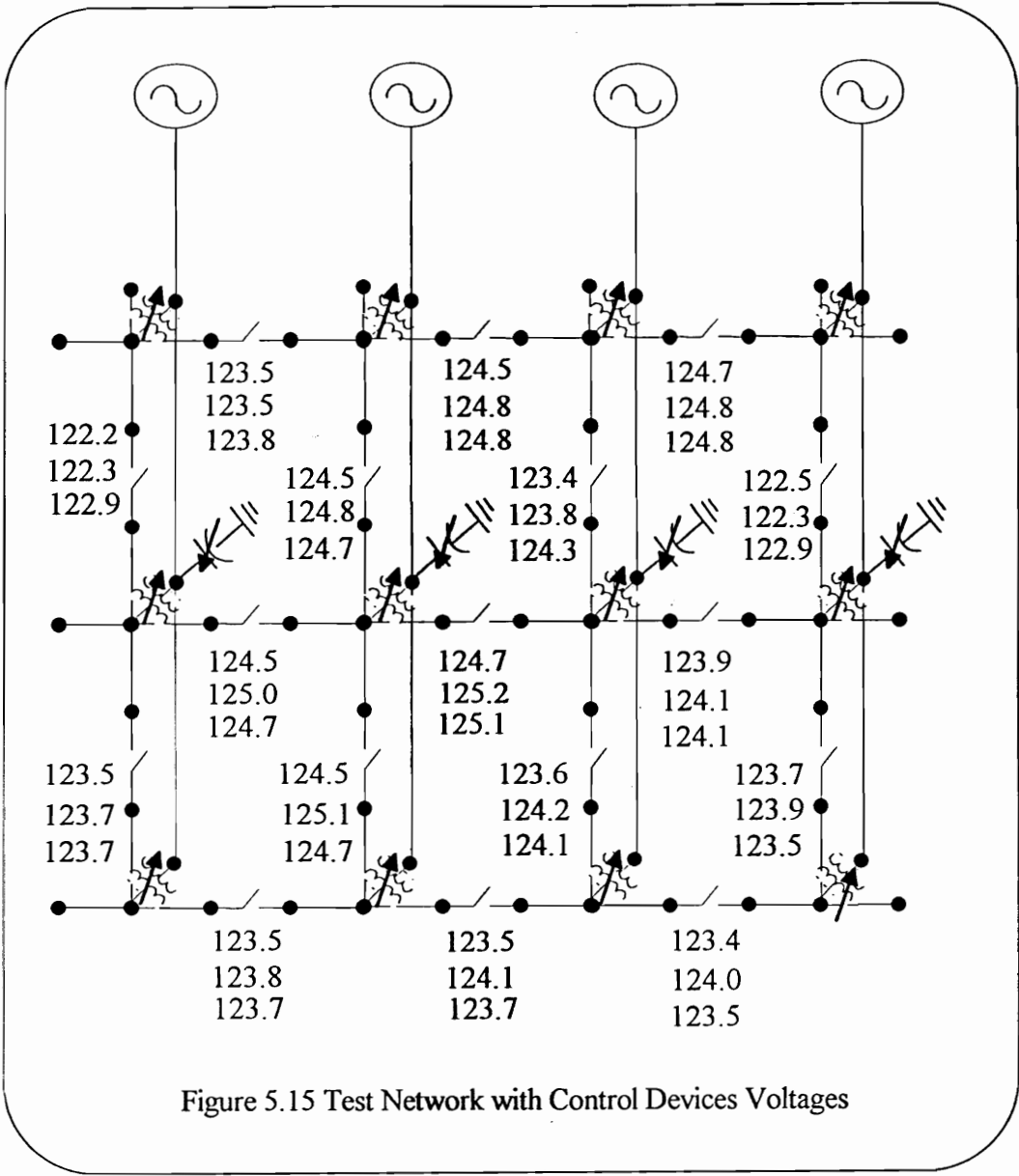


Figure 5.15 Test Network with Control Devices Voltages

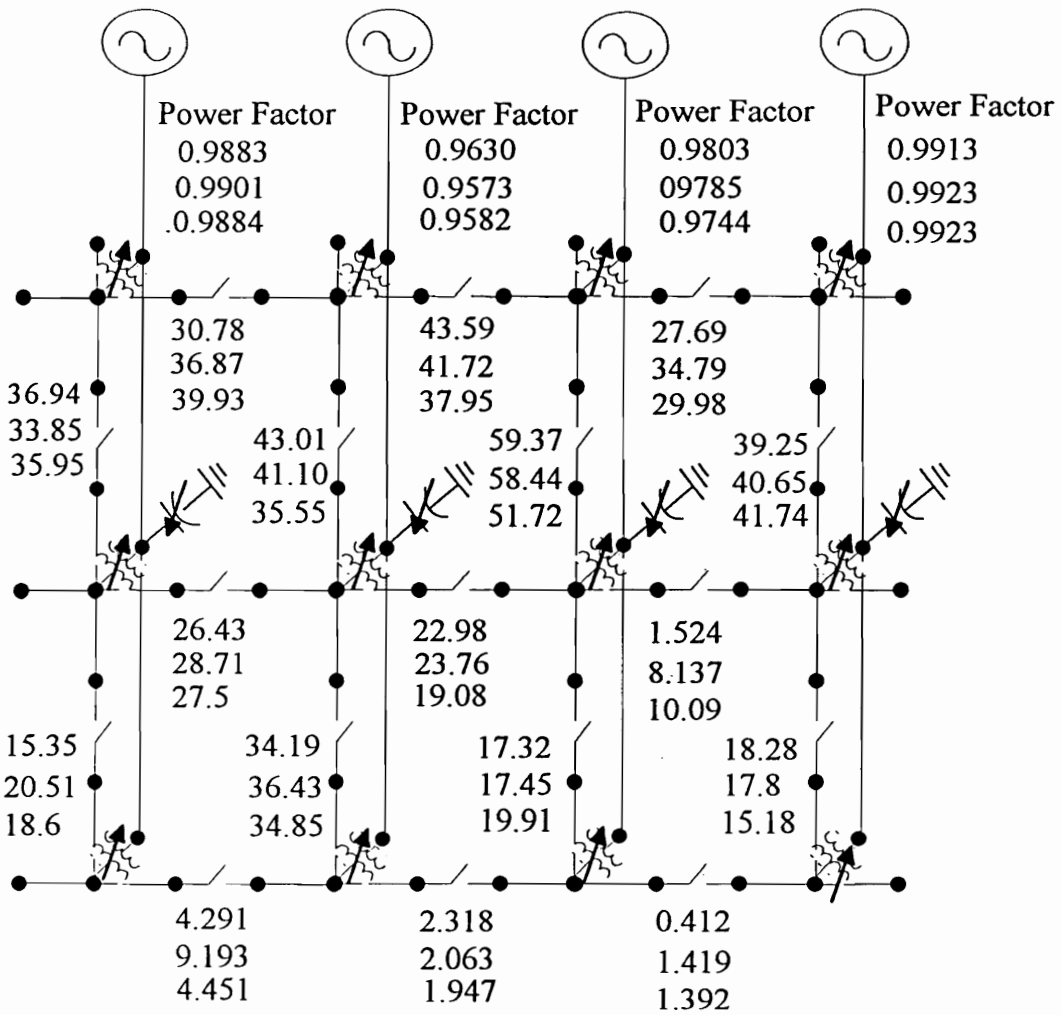


Figure 5.16 Test Network with Control Devices Currents/Power Factor

The loads in the network were increased to test the ability of the algorithm to converge on a heavily loaded circuit. Each of the loads in the network was increased to 60kW and 12kVARs. The resulting current in every 4/0 line section in the secondary network exceeded the rated value of 480 Amperes.

The transformer power rating was increased to allow all the lines in the secondary network to be overloaded. The power rating of each transformer was raised to 12,000kVA. The leakage reactance was maintained at 5% of the base value. Winding losses were neglected for this simulation.

The algorithm was allowed to update the sensitivity matrix after each iteration. Updating the sensitivity matrix is not required for this test network. However, the objective was to determine if the algorithm would converge. The solution time was 208 seconds and required 16 iterations. The resulting voltages at the co-tree components are shown in Figure 5.17.



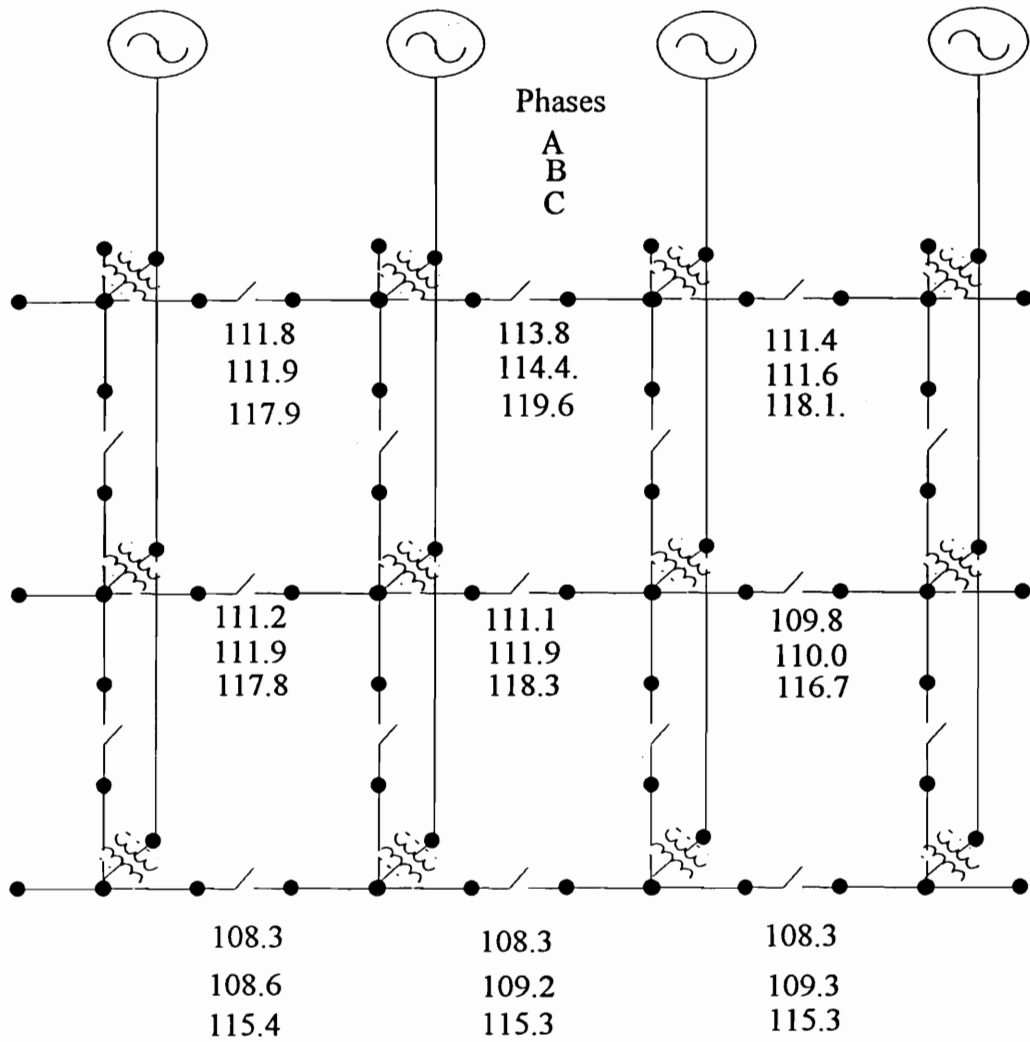


Figure 5.17 Typical Voltage Magnitude Values for Heavily Loaded Circuit

## CHAPTER 6

### CONCLUSIONS AND RECOMMENDATIONS

The Switch Compensation Method can solve distribution networks which are mutually coupled and unbalanced. This method is capable of addressing the modeling challenges required by distribution networks while running on a personal computer platform.

This method is an extension of the Ladder Load-Flow method with the emphasis on the simulation of Tap Changing Under Load Transformers and shunt capacitors with automatic tap controls. The simulation of automatic control devices in the network requires the sensitivity matrix be updated after the control action. The Switch Compensation Method uses the information and data generated by the iterative Ladder Load-Flow method to update the sensitivity matrix.

#### 6.1 Comparisons of Solution Times

The absolute comparison of solution times is difficult due to a number of factors. The solution time depends on the machine on which the system was solved and the operating system. The algorithms are different in terms of functionality and modeling capability. The test systems on which the algorithms are tested are different. The selection of the convergence tolerance will also affect the solution times.

The algorithm proposed by G.X. Luo models P,V Buses but does not model mutual coupling or shunt admittance [9]. The paper reports that program was run on a PC/AT (with a 287 coprocessor). The TS1 test system consisted of 30 Buses, two of which were P,V Buses and three loops. The total solution time reported is 11 seconds. This time drops to 0.4 seconds when the program is run on a VAX Micro-2. The convergence tolerance was 0.005 p.u.

These results are comparable to the solution times obtained in Figure 5.14 for the Switch Compensation Method. The program was run on a 486 microprocessor (50 MHz) with a math coprocessor. The operating system was OS2.1. However, the convergence tolerance of the Switch Compensation Method is less than  $2 \times 10^{-10}$  p.u.

The algorithm proposed by Carol S. Cheng models mutual coupling, shunt admittance and P,V Buses. The program was run on a SUN Sparc 10 with a single CPU. The test system features 1418 line sections, 1419 nodes, 20 capacitors and 3 voltage regulators. The solution time for 5 loops and no P,V nodes is 1.52 seconds. No mention is made of the convergence tolerance which was used to obtain these results.

The SUN Sparc 10 and the IBM 486 machine are clearly generations apart in terms of performance and cost [28]. Cross-platform testing is always complicated by the lack of truly equivalent benchmark tests.

The convergence pattern is also an important characteristic of the algorithm. A nearly monotonic convergence pattern was observed when all three of the co-tree currents were changed before each iteration. This is the same result reported by Cheng. Certainly, more experimentation is required before this result is supported.

## 6.2 Calculation of Sensitivity Matrix

One of the major differences between the Switch Compensation Method proposed here and previous algorithms is the manner in which the sensitivity matrix is determined and used. The work of others in this area has been based on the use of Ohm's Law to determine the equivalent Thevenin impedance at the co-tree components. The Switch Compensation Method uses Thevenin's Theorem to determine the elements of the sensitivity matrix.

There are several advantages to using Thevenin's Theorem with the Ladder Load-Flow Method. The major advantage is that the Ladder Load-Flow Method inherently generates the necessary data to calculate the elements of the sensitivity matrix as part of the iterative solution process. Thus the algorithm can accurately determine the sensitivity matrix for more sophisticated models of power system components than have previously been employed with this method. This includes mutual coupled lines, shunt admittance of lines and delta-wye transformer configurations.

Another advantage of this approach is that the proposed modifications can be implemented as a separate module, without revisions to working code. This makes the code more maintainable. A modification to a component model in the Ladder Load-Flow algorithm will be inherently reflected in the calculation of the sensitivity matrix elements.

The Thevenin based approach also allows the algorithm to update the sensitivity matrix as part of the iterative process. The consequences of this feature will be explored in detail in the next section.

This approach proved sufficient to converge the algorithm to within .00001 percent of the correct solution of all cases tested. The CPU time required to generate the sensitivity matrix in this manner is typically one third of the total solution time. This appears to be competitive with other methods which typically require more than half of the total solution time to build and factor the sensitivity matrix.

### **6.3 Updating the Sensitivity Matrix**

The use of the Ladder Method to calculate the sensitivity matrix elements allows the sensitivity matrix to be updated from the information provided by the previous iteration. This scheme is similar to a predictor / corrector algorithm. This approach is required when tap changing transformers and controlled capacitors effectively change the impedance of the network during the solution process.

The determination of the sensitivity matrix as a  $\Delta V/\Delta I$  is inherit in the network solution process using the Ladder Method. This process begins by converging each of radial circuits to a solution. A current is then injected into each circuit at the co-tree components. The radial circuits are again converged to a solution and the process is repeated until the voltage difference between the connecting circuits is within tolerance. The  $\Delta V$  is the change in voltage on a co-tree component from one radial solution to the next. The current change  $\Delta I$  is the change in the value of the injected current through the co-tree component from one radial solution to the next.

In a network which does not contain mutually components, this determination of the sensitivity matrix after each radial solution requires little additional time and memory. The voltages and currents at the points where the radial circuits connect must be stored in a memory location after each radial solution. This represents the only additional memory requirement. In the absence of mutual coupled components each of the three phase co-tree currents can be determined independently. The only additional calculation is forming the differences between the voltages and currents and dividing the differences by one another  $\Delta V/\Delta I$ .

Updating the sensitivity matrix after each radial solution allows the method to solve networks with automatic control elements. Recent literature reports that the sensitivity matrix may be formed only once for circuits with constant power loads, voltage regulators and capacitors [24]. The results of the simulations contained in this work confirm these assumptions.

However, the inclusion of components in the network which significantly change the impedance from iteration to next defeats the approach of forming the sensitivity matrix only once. The inclusion of tap changing transformers and controlled capacitors in the test network resulted in 37 iterations without a solution. Previously, the algorithm required 7 iterations to reach a solution. This result is expected as sensitivity matrix elements change as a function of the transformer tap setting and amount of capacitance.

The calculation of the sensitivity matrix using the Ladder-Method allows the sensitivity matrix elements to accurately reflect changes to the network as the algorithm is converging. However, as the results indicate the solution time increases in a nonlinear fashion as the number of co-tree elements increases.

The significant increase in solution time is a consequence of modeling mutual coupled components. When components are mutually coupled all the elements of the sensitivity matrix must be calculated. This requires the algorithm to change only one phase of the three phase co-tree currents and determine the corresponding change in voltage in all three phases. The one column of the sensitivity matrix is then constructed for each phase current. This increases the number of iterations by roughly three times that required if all three-phase co-tree currents are modified between iterations. The

additional memory required is a linear function of the number of co-tree elements.

#### **6.4 Absence of Matrix Operations**

The high execution speed of this algorithm can be attributed to two factors. First the extensive use of the backward and forward traces using pointers to accomplish most the processing. Second only small matrixes need to be formed and processed at each of the co-tree elements. These features not only reduce the solution time but also keep the required computer memory resources at a minimum. This allows the method to solve large systems on a personal computer platform.

The work of others in this regard has involved constructing a system matrix which is dimensioned by the number of co-tree components (loops) in the network. As the size of the system increases the proportion of time spent in building and factoring this matrix becomes a greater portion of the total solution time [9]. The memory required for this operation also increases sufficiently. Because the proposed method does not use global matrix operations it does not exhibit the same properties.

Since the method uses the trace based approach to determine the sensitivity matrix the processing time is expected to increase linearly with the number of co-tree elements in the circuit. The memory requirement is also a linear function of the number of co-tree elements.



## 6.5 Voltage Correction

The inclusion of voltage correction at the co-tree buses in the network solution had little effect on the solution time. This is in contrast to the first system in which this technique was applied. The difference results from the manner in which voltage correction was applied in each algorithm.

The algorithm described in Section 4.4 changes all the co-tree currents in the system before solving the radial circuits. The Switch Compensation Method changes only one co-tree current then solves the radial circuits. Voltage correction is helpful in the first case when one Bus is connected to two or more co-tree components.

When one Bus serves as the connection between two or more radial circuits the Bus voltage will be effected by all the co-tree currents. Consequently, changing the voltage at the Bus as a result of one co-tree current injection provides a better estimate of the voltage at the Bus. This voltage is then used in determination of any other co-tree currents at the Bus. This reduces the large swings in co-tree current magnitude and angle for the first several iterations.

The Switch Compensation Method uses the Ladder Load-Flow method to solve each of the connecting radial circuits after each change in co-tree current. Therefore, the only effect of voltage correction is to provide a better estimate of the voltage for the Ladder Load-Flow. As the size of the system

increases the inclusion of voltage correction does reduce the radial solution time slightly (1 second).

## **6.6 Recommendations**

Sensitivity matrix elements should only be updated during the solution process if required. The significant increase in solution time to update the sensitivity matrix can only be justified if it is absolutely necessary to converge the algorithm.

The presence of controlled capacitors did not prevent the algorithm from converging. The presence of Tap Changing Under Load (TCUL) Transformers in the network prevented the algorithm from reaching a solution. Consequently the program should update the sensitivity matrix elements only if TCUL transformers are present in the network. Otherwise, the sensitivity matrix may be formed only once for each load flow solution.

The inclusion of voltage correction as described in Section 2 does not result in sufficient time savings in the solution process for distribution networks. However, this correction can result in sufficient time savings for other types of problems. The small amount of code, data and processing time required to implement this correction justify its inclusion in the method.

## 6.7 Future Considerations

A quick and simple method of approximating the sensitivity matrix for the first iteration would enhance the performance of the algorithm with TCUL transformers. One third of the total solution time is spent initializing the sensitivity matrix for the first iteration. The co-tree currents determined from this sensitivity matrix are only approximation for the first several iterations. Therefore, the time spent in accurately initializing each of the elements in the sensitivity matrix does not seem to be justified.

The selection of co-tree components is another topic which deserves further exploration. The selection of co-tree components is not critical in systems which have been designed for high reliability. This is certainly the case for Distribution Networks. The loss of one or two feeders will generally not cause an interruption of service in a Distribution Network.

The improper selection of co-tree components could prevent the radial load-flow from reaching a solution. This can result when the loading on a radial circuit is larger than the maximum allowed by the Maximum Power Transfer Theorem. A selection scheme which would identify the "optimal" co-tree components in a system would be useful.

Finally, further experimentation with the algorithm is required. The algorithm should be tested on larger networks and systems. The results of these tests need to be validated.

## Appendix A

### UNBALANCED THREE-PHASE TRANSFORMER MODELING

The Ladder Load-Flow method models components with a reverse current trace and a forward voltage trace. The purpose of this section is to derive a three-phase transformer model for the reverse current trace and forward voltage of the Ladder Method. This could be accomplished using the theory of symmetrical components. However, this would involve transforming the currents or voltages from the phase coordinates into the zero, positive and negative values. Since the Ladder Method uses phase quantities these values would have to be transformed back to the phase coordinates. The goal is to determine a transformer model in the phase coordinates system which is both simple and requires few calculations. Therefore, symmetrical components will be employed sparingly.

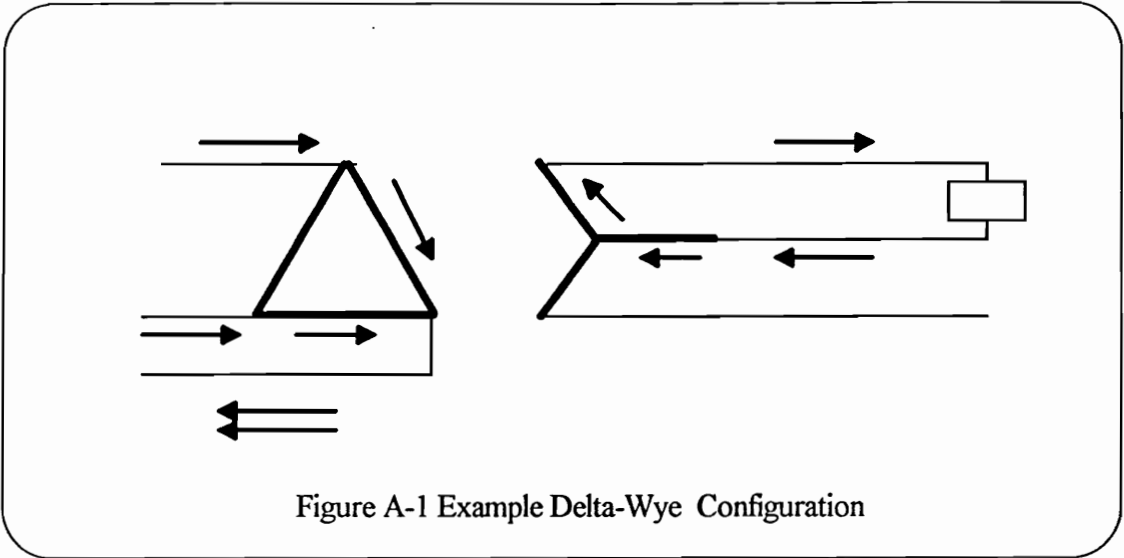
Three phase transformers are modeled as three single phase units. The modeling technique presented here is not applicable to three phase transformers in which the windings are wound on a common core i.e., shell type and core-type transformers. Each of the three units is also assumed identical leakage reactances.

## A.1 Current Reverse Trace

Transformer models must satisfy both the laws of circuit theory and laws of magnetic theory. Kirchhoff's current law states that the sum of the current into a node must equal the sum of the current out of the node. Ampere's law states that the magnetomotive force around a closed path is the current enclosed by the line integral of the field intensity  $H$  around the path. If the magnetic core of the transformer is ideal ( $H_{core} \bullet l = 0$ ), Ampere's law applied to a path enclosing the primary winding  $N_1$  and secondary winding  $N_2$  results in the following relationship.

$$\oint H \bullet dl = N_1 I_1 - N_2 I_2$$

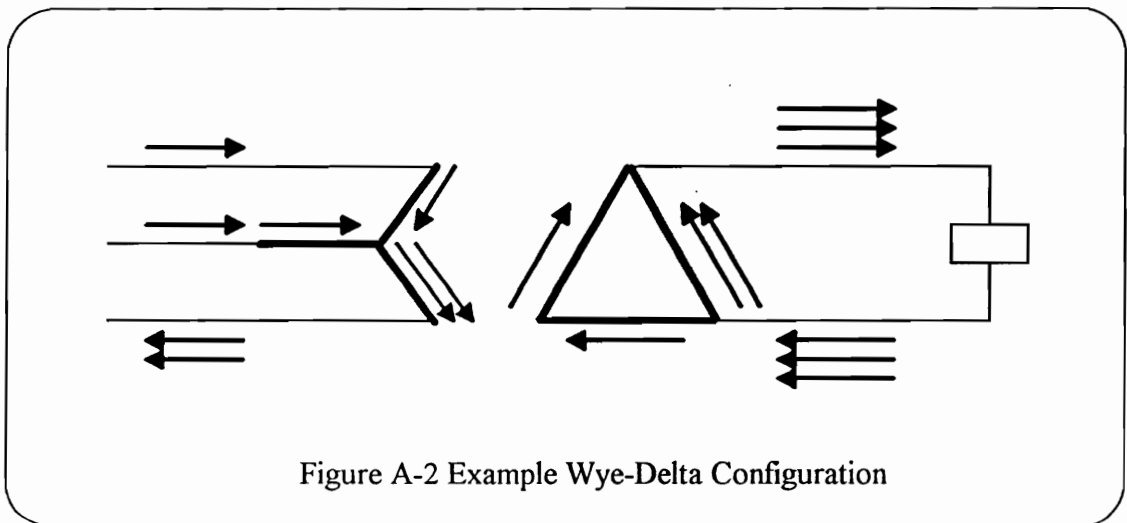
The operation of three-phase transformers under unbalanced loading conditions may be determined by applying these two laws to the resulting configuration. This will be illustrated with the delta-wye configuration shown in Figure A-1. For the sake of simplicity  $N_1 = N_2$  in Figure A-1. Windings which are magnetically coupled are drawn in parallel with one another. Current flow is indicated with arrows. The relative magnitude of the currents is indicated by the number of current arrows drawn.



The determination of the currents begins by determining the current flow that must exist on the secondary of wye side of the transformer. The application of Kirchhoff's current law indicates that the currents must flow in the secondary windings as indicated in Figure A-1 to supply the single phase load. Next the application of Ampere's law determines the current flow in the primary windings. A current flow into a secondary winding must flow out of the primary winding. Kirchhoff's current law is applied again to determine the current flow in the primary.

It is also important to realize that delta connected windings must satisfy Kirchhoff's Voltage Law (KVL) which states that the algebraic sum of the voltages encountered in traversing each loop of a circuit must be zero. KVL is satisfied for the delta winding shown in Figure A-1 only if the leakage reactances of the windings are equal to one another. If the leakage reactances

are unequal the currents must divide to satisfy KVL. However, if the leakage reactances are equal the line current can be divided into three equal values. Two-thirds of the line current will flow through one winding of the delta configuration. One-third of line current will flow through two of windings of the delta configuration. This is the case with the wye-delta configuration shown in Figure A-2.



This approach can be extended to unbalanced three phase currents. For the delta secondary configuration Kirchhoff's current law requires that the sum of the line currents must equal zero ( $I_A + I_B + I_C = 0$ ). One of the three line currents can therefore be expressed in terms of the remaining two line currents. Figure A-3 illustrates how this configuration can be decomposed into two circuits. The principle of superposition can be then be used to obtain the resulting primary line flows.

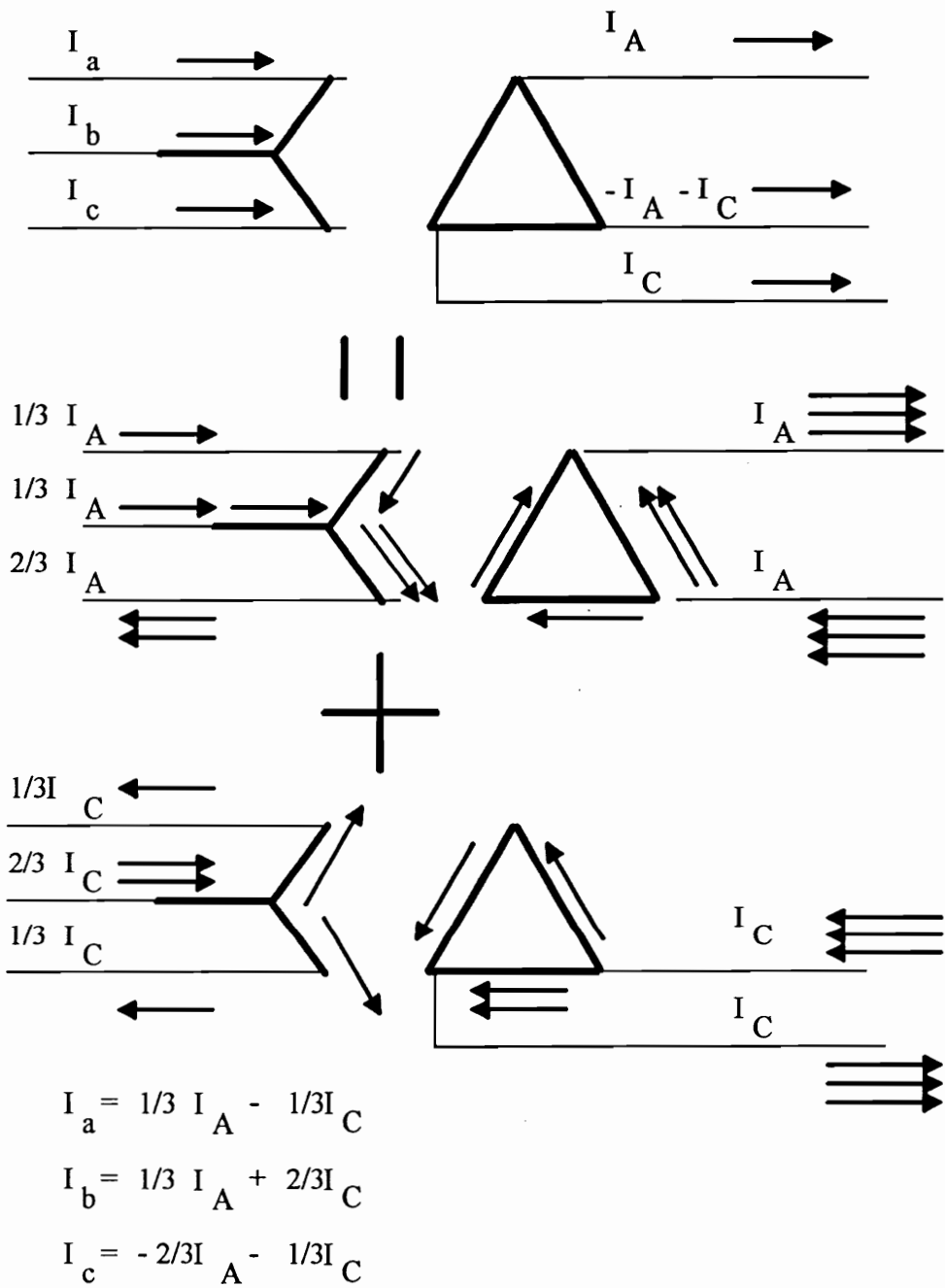


Figure A-3 Superposition Used To Solve Wye-Delta Configuration



A wye connected secondary makes the current transformation from secondary to primary a simple matter. In order for current to flow in the neutral of the wye configuration a zero-sequence component must exist. This will not be case when the primary is a delta configuration. Consequently, the delta-wye and delta-grounded wye configurations result in the same transformation.

The method will also indicate if a transformer configuration cannot carry a single phase load. This is the case for a wye-grounded wye configuration. A single phase load connected from line-to-neutral on the secondary requires current flow through only one of the phase windings of the grounded wye connection. The application of Ampere's law would require current to flow only one winding of the wye connected primary. However, Kirchhoff's law states that current entering the node of this wye configuration must leave node through one of the other windings. This would violate Ampere's law because no current flows through this winding on the secondary i.e.,  $N_1 I_1 \neq N_2 I_s$ .

Laboratory experimentation with the wye-grounded wye configuration indicates that this configuration can carry a limited amount of line-to-neutral load. This can observed result can be understood by considering the magnetizing inductance of the transformers [23]. The three-phase interconnection is shown in Figure A-4.

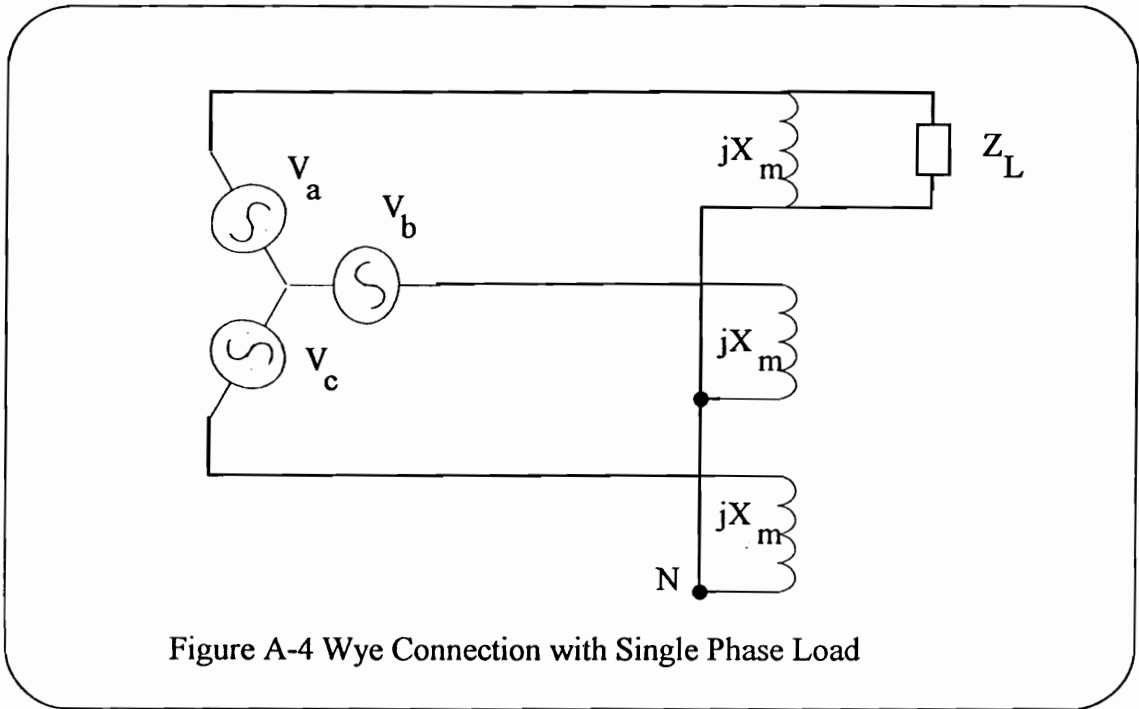


Figure A-4 Wye Connection with Single Phase Load

The leakage flux and losses are neglected in the model shown in Figure A-4. The secondary load can then be represented on the primary side as shown in the Figure. If the transformers are supplied from a balanced set of voltages the neutral point (N) can be determined as the geometric neutral of the voltage triangle.

If the magnetizing reactances are assumed to be linear, Thevenin's theorem may be applied at the load terminals to determine the equivalent circuit shown in Figure A-5. This indicates that the magnetizing reactance limits the current flow to the load.

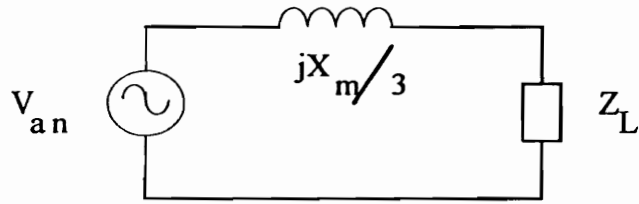


Figure A-5 Thevenin Equivalent Circuit

## A.2 Voltage Forward Trace

The voltage relationship of the primary and secondary windings of a transformer are governed by Faraday's law. We shall begin our analysis by assuming that the flux varies sinusoidally in the core and that the transformer core is ideal, which means that the permeability of the core is infinite. Faraday's law can then be used to show that the ratio of the primary voltage to the secondary voltage is the same as the number of turns on the primary winding to the number of turns on the secondary winding. Kirchhoff's voltage requires the sum of the voltage around any closed loop to be zero.

The configuration of the both the primary and secondary windings must be considered in determining the voltage relationships. Figure A-6 illustrates the voltage relationships that result for a delta-wye configuration.

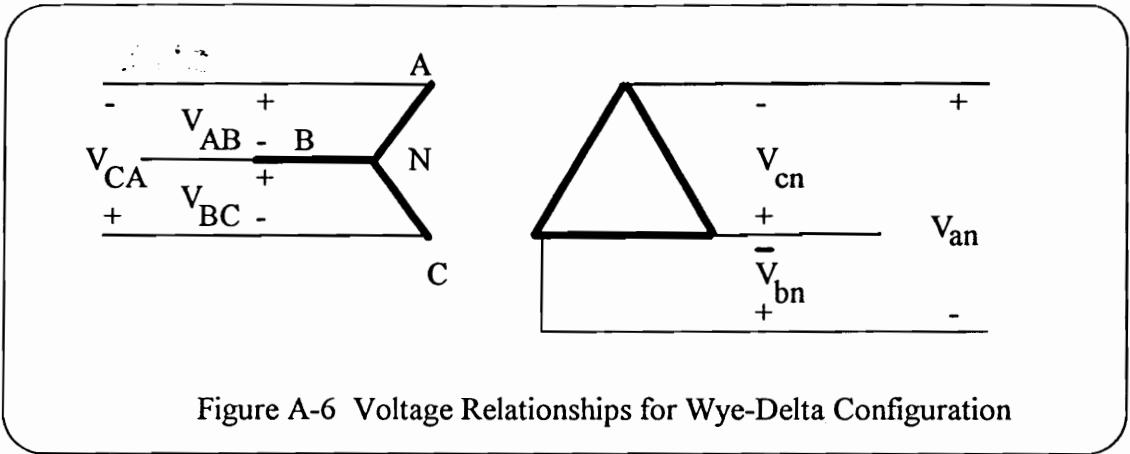


Figure A-6 Voltage Relationships for Wye-Delta Configuration

The voltage loop on the primary side specifies that the sum of the primary line-to-line voltages must be zero ( $V_{AB} + V_{BC} + V_{CA} = 0$ ). The voltage loop on the secondary side specifies that the sum of secondary line-to-line voltages must also be zero ( $V_{an} + V_{bn} + V_{cn} = 0$ ). The line-to-line voltages on the secondary are the line-to-neutral voltages on the primary. This leads to the following relationships:

$$V_{AB} + V_{BC} = -V_{CA} = V_{an} - V_{cn} = V_{an} + V_{an} + V_{bn} = 2V_{an} + V_{bn} \quad (A-1)$$

$$V_{BC} + V_{CA} = V_{bn} - V_{an} \quad (A-2)$$

Combining equations A-1 and A-2 results in an expression for the secondary voltage in terms of the primary voltage  $V_{an} = 1/3(V_{AB} - V_{CA})$ . The remaining secondary voltages can be determined by back-substitution of this expression.

The unbalanced operation of three phase transformers can also be described using symmetrical components. When a neutral connection is not present the zero sequence voltage is zero. Transformer configurations which

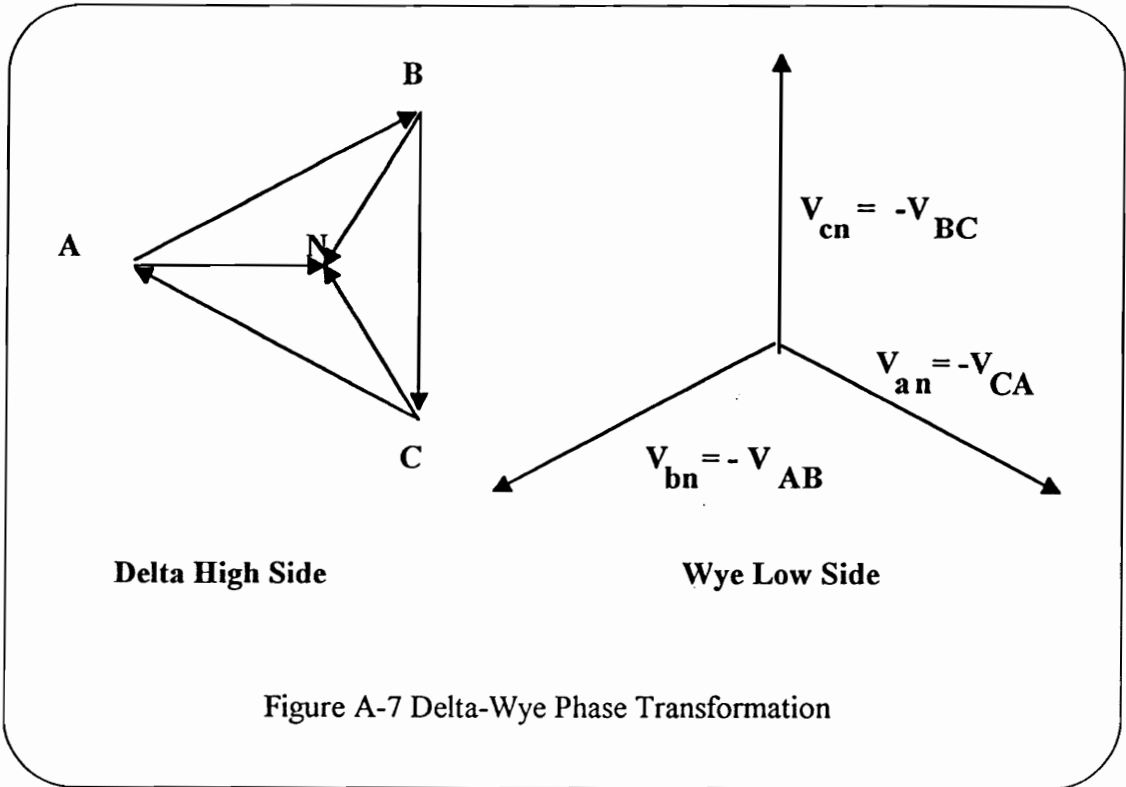
are grounded on the primary and ungrounded on the secondary will filter out the zero sequence voltage. For example a wye ground to delta configuration could have zero sequence currents and voltages in the primary winding but not in the secondary lines. The zero sequence voltage is filtered by the algorithm when line configuration changes from 4-wire to 3-wire.

### A.3 Phase Shift of Wye-Delta

The American standard for designating terminals  $H_1$  and  $X_1$  on wye-delta transformers requires that the positive-sequence voltage drop from  $H_1$  to neutral lead the positive-sequence voltage drop from  $X_1$  to neutral by 30 degrees, regardless of whether the wye or the delta winding is the high voltage side [25]. There are several possible phase relationships which can be used to meet this standard. The phase relationship which was selected for the DEWorkstation is that phase A on the high side must lead phase A on the low side by 30 degrees. Similarly phase B on the high side must lead phase B on the low side etc..

Faraday's law states that the primary and secondary winding voltages will be in phase. Figure A-7 illustrates how the phases can be labeled in a delta-wye configuration to achieve the desired phase relationship. This phase transformation is simple to implement in the forward voltage trace. The voltages of both the primary and secondary are declared as arrays. Assume that the primary voltages are stored in the following fashion.  $V[0]=V_{AB}$ ,

$V[1]=V_{BC}$ , and  $V[2]=V_{CA}$ . The phase transformation shown in Figure A-7 is implemented in the forward trace as follows:  $v[0]=-V[2]$ ,  $v[1]=-V[0]$  and  $v[2]=-V[1]$ . Assuming  $v[0]=v_{an}$ ,  $v[1]=v_{bn}$  and  $v[2]=v_{cn}$ .



A similar analysis of the Wye High Side to a Delta Low Side will indicate the following results.  $V_{AB}=V_{an}$ ,  $V_{BC}=V_{bn}$  and  $V_{CA}=V_{cn}$ . Where the Line-to-neutral values are on the wye side.

## A.4 Computer Code

This Section provides a listing of the transformer Ladder Load-Flow routines provided with the DEWorkstation. The transformers are assumed to be ideal by these routines. The voltage drop across the leakage reactance and winding resistance is calculated prior to calling the "secondaryVoltsXformer" routine.

The routines are written in the "C" programming language. The routines make use of several library functions. The "next" function determines the next phase which is present and returns the phase index in "p". The value of the "next" function is -1 when all phases have been processed. The "message" function displays text on the screen. The "cadd", "csub", "cdiv", and "cmult" are complex arithmetic functions.

```

/*-----
=> Module Name: secondaryVoltsXfromer
=>
=> Module Purpose: Computes secondary voltages of different xfromer
=>                   connections for given primary voltages.
=>
=>
=> Inputs:
=> pCmp      Pointer to transformer
=> acxPriV   Array of complex primary voltages on transformer
=> ad        Array of winding to winding turns ratio
=>
=> acxSecV   Array of complex secondary voltages on transformer
=>
=> Explanation: Voltages are: Line-to-Neutral for GROUNDED system
=>                   Line-to-Line for UNGROUNDED system
=>
=> Notes: For delta-wye connections the phase shift is in accordance
=>         with the American standard assuming (abc) sequence.
/*-----

```

```

long secondaryVoltsXfromer ( struct Cmp *pCmp, struct Cmplx acxPriV[3],
                           double ad[3] ,struct Cmplx acxSecV[3] )
{
    struct Cmplx
        acxIntermV[3], // Winding voltage on secondary of transformer
        acxSeqV[3],   // sequence voltages
        cxSQRT3,      // Complex value of square root of 3
        cxPhaseShift, // Phase shift of 30 degrees
        cxZero;       // Complex zero
    long p;           //phase index

    cxZero.re = 0.0;
    cxZero.im = 0.0;
    if( pCmp->bCmpTyp != XFORMER_CMP)
    {
        message(ERR_MESS,pCmp,iNoApp,"forwardTraceXfromer routine called for this",
                "COMPONENT which is NOT a transformer" );
        return FAIL;
    }
    for( p=0;p<3; p++ )
    {
        acxIntermV[p].re = acxPriV[p].re*ad[p];
        acxIntermV[p].im= acxPriV[p].im*ad[p];
    }

    switch(pCmp->pCtr->iXfrmCon )
    {
        case WYE_WYEGRD:
            cxSQRT3.re  = SQRT3;
            cxSQRT3.im  = 0.0;
            cxPhaseShift.re = 0.866025404;

```



```

cxPhaseShift.im = -0.5;
for( p=0;p<3;++p )
{
    acxSecV[p]=cdiv( acxIntermV[p], cxSQRT3 );
    acxSecV[p]=cmult( acxSecV[p],cxPhaseShift );
}
break;
case YGRD_Y:
    vtovseq( acxIntermV,acxSeqV );
    acxIntermV[0] = cadd( acxSeqV[1],acxSeqV[2] );
    acxIntermV[1] = cadd( cmult(a2,acxSeqV[1] ),
                          cmult(a,acxSeqV[2]) );
    acxIntermV[2] = cadd( cmult(a,acxSeqV[1] ),
                          cmult(a2,acxSeqV[2]) );
    acxSecV[0] = csub( acxIntermV[0],acxIntermV[1] );
    acxSecV[1] = csub( acxIntermV[1],acxIntermV[2] );
    acxSecV[2].re = -acxSecV[0].re-acxSecV[1].re;
    acxSecV[2].im = -acxSecV[0].im-acxSecV[1].im;
break;
case YGRD_DEL:
    vtovseq( acxIntermV,acxSeqV );
    acxSecV[0] = cadd( acxSeqV[1],acxSeqV[2] );
    acxSecV[1] = cadd( cmult(a2,acxSeqV[1] ),
                      cmult(a,acxSeqV[2]) );
    acxSecV[2] = cadd( cmult(a,acxSeqV[1] ),
                      cmult(a2,acxSeqV[2]) );
break;
case DEL_Y:
    acxSecV[0] = csub( acxIntermV[0],acxIntermV[2] );
    acxSecV[1] = csub( acxIntermV[1],acxIntermV[0] );
    acxSecV[2].re = -acxSecV[0].re - acxSecV[1].re;
    acxSecV[2].im = -acxSecV[0].im - acxSecV[1].im;
break;
case DEL_YGRD:
    acxSecV[0] = csub( cxZero,acxIntermV[2] );
    acxSecV[1] = csub( cxZero,acxIntermV[0] );
    acxSecV[2] = csub( cxZero,acxIntermV[1] );
break;
case Y_DEL:
    acxSecV[0].re = (2.0/3.0)*acxIntermV[0].re +
                    (1/3.0)*acxIntermV[1].re;
    acxSecV[0].im = (2.0/3.0)*acxIntermV[0].im +
                    (1/3.0)*acxIntermV[1].im;
    acxSecV[1].re = (1.0/3.0)*acxIntermV[1].re -
                    (1.0/3.0)*acxIntermV[0].re;
    acxSecV[1].im = (1.0/3.0)*acxIntermV[1].im -
                    (1.0/3.0)*acxIntermV[0].im;
    acxSecV[2].re = -acxSecV[0].re -acxSecV[1].re;
    acxSecV[2].im = -acxSecV[0].im - acxSecV[1].im;
break;
default:

```

```
        while( next ( pCmp, &p ))
        {
            acxSecV[p]=acxIntermV[p];
        }
        break;
    } // end switch

return SUCCESS;

} // end of secondaryVoltsXfromer()
```

```

/*
=====> Module Name: primaryAmpsXformer
=====>
=====> Module Purpose: Computes primary currents of different transformer
=====>                    connections for given secondary currents.
=====>
=====> Inputs:
=====>   pCmp      Pointer to transformer
=====>   acxPriA   Array of complex primary currents on transformer
=====>   ad       Array of winding to winding turns ratio
=====>   acxSecA   Array of complex secondary currents on transformer
=====>
=====> Definition:
=====>
=====> Explanation:
=====>
=====> Notes: For delta-wye connections the winding to winding turns
=====>        ratio must be identical for all phases.
=====>
*/
long primaryAmpsXformer( struct Cmp *pCmp, struct Cmplx acxSecA[3],
                        double ad[3], struct Cmplx acxPriA[3] )
{
    struct Cmplx
        cxLineA[3], // Line currents
        cxZero,    //Complex Zero
        cxWindA[3]; // Winding currents
    long p;        // phase index

    cxZero.re=0.0;
    cxZero.im=0.0;
    if( pCmp->bCmpTyp != XFORMER_CMP)
    {
        message(ERR_MESS,pCmp,iNoApp,"forwardTraceXformer routine called for this",
                "COMPONENT which is NOT a transformer" );
        return FAIL;
    }

    switch(pCmp->pCtr->iXfrmCon )
    {
        case YGRD_DEL:
        case Y_DEL:
            acxPriA[0].re = -(ad[0]/3.0)* ( - acxSecA[0].re + acxSecA[1].re );
            acxPriA[0].im = -(ad[0]/3.0) * ( - acxSecA[0].im + acxSecA[1].im );
            acxPriA[1].re = -(ad[1]/3.0) * ( - acxSecA[0].re - 2.0 * acxSecA[1].re );
            acxPriA[1].im = -(ad[1]/3.0) * ( - acxSecA[0].im - 2.0 * acxSecA[1].im );
            acxPriA[2].re = - acxPriA[0].re - acxPriA[1].re;
            acxPriA[2].im = - acxPriA[0].im - acxPriA[1].im;
            break;
        case DEL_YGRD:
        case DEL_Y:
    }
}

```

```

cxWindA[0].re = -acxSecA[1].re *ad[1];
cxWindA[0].im = -acxSecA[1].im *ad[1];
cxWindA[1].re = -acxSecA[2].re *ad[2];
cxWindA[1].im = -acxSecA[2].im *ad[2];
cxWindA[2].re = -acxSecA[0].re *ad[0];
cxWindA[2].im = -acxSecA[0].im *ad[0];
acxPriA[0] = csub( cxWindA[0], cxWindA[2] );
acxPriA[1] = csub( cxWindA[1], cxWindA[0] );
acxPriA[2].re = -acxPriA[0].re -acxPriA[1].re;
acxPriA[2].im= -acxPriA[0].im -acxPriA[1].im;
break;
default:
while ( next ( pCmp, &p ) )
{
    acxPriA[p].re=acxSecA[p].re*ad[p];
    acxPriA[p].im=acxSecA[p].im*ad[p];
} //end of while next
break;
} // end switch
return SUCCESS;

} // end function primaryAmpsXfromer0

```

## **Appendix B**

### **NETWORK LOAD-FLOW DRIVER AND MODULES**

The Switch Compensation Method requires the reverse trace of the Ladder Load-Flow algorithm to process the co-tree currents. If constant load currents can be attached to any component in the circuit then no modifications are necessary to the Ladder Load-Flow Method. Another approach would be to require the co-tree components to be a particular type of component (i.e. switch or line).

The additional processing required by the Method can be accomplished with separate modules. The data interface is simply the voltage and co-tree currents at each component which connects to another radial circuit. The tasks at each of these components is as follows:

1. Main Driver
2. Determination of the Sensitivity Matrix
3. Addition of two Sensitivity Matrixes
4. Solve a system of three equations
5. Determination of the desired change in co-tree current
6. Determination of the change in voltage (Optional)
7. Initialize Co-tree currents to One Amp
8. Check for Network Convergence

Routines which will add two matrixes and solve a system of linear equations are commonly available as library routines. Therefore the design of these two modules will not be presented here. The design of the remaining modules will be presented in Program Design Language (PDL). This is not working computer code.

The salient variables used and generated by each module will be defined. The data and operation flow of each module will be outlined. It is assumed that the voltage and co-tree currents on each connecting component are available from the Ladder Load-Flow program. The connecting components connect between two radial circuits. The implementation is only one of several possible solutions.

## B.1 Main Driver

Purpose	This module controls the program flow and provides the global variables for all the modules	
Global Variables for each Connecting Component	SensMatrix[3][3]	Sensitivity Matrix with complex numbers for each element
	LastVoltage[3]	Voltage from last solution with complex numbers for each voltage
	LastCoTreeA[3]	Co-Tree Current from last solution with complex numbers for each current
Global Variables	phase	The phase index of the component
	pCkt	Radial Circuit in Network which connects to pConCkt through pCmp
	NetConverge	A Flag to indicate when a solution to the network is achieved
	pCmp	Connecting Component in Circuit pCkt which connects to pConCmp
	pConCkt	Radial Circuit in Network which connects to pCkt through pConCmp
	pConCmp	Connecting Component in Circuit pConCkt which connects to pCmp

NetworkDriver

```
{
  for( All radial circuits in the Network=pCkt)
  {
    for ( All Connecting Components = pCmp)
    {
      for ( All phases=phase)
      {
        Call Ladder Load Flow to Solve Radial Circuit
        Store Voltages in LastVoltage
        Call Module to InitializeCurrent(pCmp, phase)
        Call Ladder Load Flow to Solve Radial Circuit
        Call Module to DetermineSensMatrix(pCmp,phase)
      }end of for All phases
    }end of for All Connecting Components
  }end of for All radial circuits in the Network
while (network has not converged to a solution)
{
  for( All radial circuits in the Network=pCkt)
  {
    for ( All Connecting Components = pCmp)
    {
      for ( All phases=phase)
      {
```



```

Store Voltages in LastVoltage for pCmp
Store Co-tree Currents in LastCoTreeA for pCmp
Call Module DetermineCurrentChange(pCmp,phase)
Call Ladder Load Flow to Solve Radial Circuit pCkt
Call Module to DetermineSensMatrix(pCmp,phase)
Determine Connecting circuit = pConCkt
Determine Connecting component = pConCmp
Store Voltages in LastVoltage for pConCkt
Store Co-tree Currents in LastCoTreeA for pConCkt
Call Ladder Load Flow to Solve Radial Circuit pConCkt
Call Module to DetermineSensMatrix(pConCmp,phase)
}end of for All phases
}end of for All Connecting Components
}end of for All radial circuits in the Network
Call Module CheckNetConverge
}end of while network has not converged to a solution
}end of NetworkDriver

```

## B.2 Determination of the Sensitivity Matrix

Purpose:	This module initializes the elements of the Sensitivity Matrix and updates the elements of the matrix after each system iteration.	
Internal Variables	Voltdiff	Voltage difference between system iterations
	Ampdiff	Co-tree current difference between system iterations.
Passed Parameters	pCmp	A component for which the sensitivity matrix is to be determined.
	phase	The phase index of the component

DetermineSensMatrix ( pCmp, phase)

```

{
  for (all existing elements in the selected column "I" of the matrix )
  {
    Voltdiff = LastVoltage[I] - Voltage[I] at present
    Ampdiff = LastCoTreeA[phase] - Co-Tree Amps at present
    if ( |Voltdiff| < threshold*Convergence Tolerance )
    {
      return to the calling module
    }
    if ( |Ampdiff| < threshold*Convergence Tolerance )
    {
      return to the calling module
    }
  }
}

```

```

SensMatrix[I][phase]=Voltdiff/Ampdiff
if ( | SensMatrix[I][phase] | < Small number)
{
    SensMatrix[I][phase] = 0.0
    Return to calling module
}end if | SensMatrix[I][phase] | < Small number
if ( the real part of SensMatrix[I][phase] is negative)
{
    Change the sign of both the real and imaginary values of SensMatrix[I][phase]
}end if the real part of SensMatrix[I][phase] is negative
}end for all existing elements in the selected column "I" of the matrix

}end of DetermineSensMatrix

```

### B.3 Initialize Co-tree Currents

Purpose	This module initializes the Currents of the co-tree currents of the connecting components	
Passed Parameters	pCmp	A component for which co-tree currents are to be initialized
	phase	The phase index of the component

InitializeCoTreeCurrents (pCmp,phase)

{

    Determine phase angle of the voltage

    Set the phase of Co-tree current magnitude to one amp at the same voltage phase angle

}end of InitializeCoTreeCurrents

## B.4 Determine Current Change

Purpose	This module determines the desired change co-tree current to force the voltage difference between the two connecting components to zero.	
Internal Variables	SVoltDiff	Voltage difference between components pCmp and pConCmp which is a complex value
	pConCmp	Connecting Component
	ChangeA[3]	Desired current change which is a complex value
	MatrixSum[3][3]	Sum of the Sensitivity Matrixes each element is a complex value
Passed Parameters	pCmp	A component for which the change in co-tree current is to be determined
	phase	The phase index of the component

DetermineCurrentChange(pCmp,phase)

```

{
    SVoltDiff = Voltage on pCmp - Voltage on pConCmp
    if( |SVoltDiff| < Convergence Tolerance )
    {
        Return to Calling Module
    }end if |SVoltDiff| < Convergence Tolerance
    Determine connecting component pConCmp
    Call module to add Sensitivity Matrixes of pCmp and pConCmp return MatrixSum
    Call module to solve system of equations defined by MatrixSum[3][3] and SVoltDiff
    return ChangeA[3]
    Add ChangeA[phase] to pCmp co-tree current[phase]
    Change sign of the co-tree current on pCmp and place this current on pConCmp[phase]
    Call module VoltChange(pCmp, ChangeA[3])
}end of DetermineCurrentChange

```

## B.5 Voltage Change

Purpose	This module determines change in voltage on each phase which results from a change in all three phase currents	
Internal Variables	ChangeV	Voltage Change which is a complex value
	pConCmp	Connecting Component
Passed Parameters	pCmp	A component for which the change in co-tree current is to be determined
	ChangeA[3]	Change in current at component pCmp which is a complex value

VoltChange (pCmp, ChangeA[3])

```

{
  for ( all phases)
  {
    Determine ChangeV for phase of pCmp by multiplying row SensMatrix by ChangeA
    Subtract ChangeV from voltage at pCmp[phase]
    Determine connecting component pConCmp
    Add ChangeV to voltage at pConCmp[phase]
  }end of for(all phases)
}end of VoltChange

```

## B.6 Network Convergence

Purpose	This module determines if the network has converged to a solution	
Internal Variables	phase	The phase index of the component
	pCkt	Radial Circuit in Network which connects to pConCkt through pCmp
	pCmp	Connecting Component in Circuit pCkt which connects to pConCmp
	pConCmp	Connecting Component in Circuit pConCkt which connects to pCmp
	SVoltDiff	Voltage difference between pCmp and pConCmp which is a complex value
	SAmpDiff	The co-tree current difference between pCmp and pConCmp which is a complex value

CheckNetConverge

```

{
  if ( Maximum number of iterations exceeded )
  {
    Display a message
    Terminated the program
  }end if ( Maximum number of iterations exceeded )
  for( All radial circuits in the Network=pCkt)
  {
    for ( All Connecting Components = pCmp)
    {

```

```

for ( All phases=phase)
{
    Determine connecting component pConCmp

    Determine SVoltDiff = Phase Voltage pCmp - Phase Voltage pConCmp
    Determine SAmpDiff = Phase Co-tree Current on pCmp - Phase Co-tree
    Current on pConCmp
    if( |SVoltDiff | > Convergence Tolerance)
    {
        NetConverge = NO
    }end if( |SVoltDiff | > Convergence Tolerance)
    if( |SAmpDiff | > Convergence Tolerance)
    {
        NetConverge = NO
    }end if( |SAmpDiff | > Convergence Tolerance)
    }end for ( All phases=phase)

} end for ( All Connecting Components = pCmp)

}end for( All radial circuits in the Network=pCkt)

}end of CheckNetConverge

```



## REFERENCES

1. D.I. Sun, "Distribution System Loss Analysis and Optimal Planning," Ph.D. Dissertation, The University of Texas at Arlington, May 1980.
2. William D. Stevenson, Jr., **Elements of Power System Analysis**, 4 th Edition , McGraw Hill Book Company, New York, 1982, p 5.
3. Jay Britton, "An Open, Object-Based Model As The Basis Of An Architecture For Distribution Control Centers", IEEE Power Engineering Society 1992 Winter Meeting, New York, New York, Paper No. 92 WM 184-2 PWRS.
4. J. Arrillaga, C.P. Arnold and B.J. Harker, **Computer Modeling of Electrical Power Systems**, John Wiley and Sons, New York, 1983.
5. Howard A. Smolleck and Raymond R. Shoults, "A Straightforward Method for Incorporating Mutually-Coupled Circuits in the Bus Admittance Matrix Using the Concept of Artificial Branches", IEEE Transactions on Power Apparatus and Systems, Vol. 5, No. 2 (May 1990) pp. 486-491.
6. M.A. Wortman, D.L. Allen and L.L. Grigsby, "Techniques for the Steady State Representation of Unbalanced Power Systems Part I. A Systematic Building Block Approach to Network Modeling", IEEE Transactions on Power Apparatus and Systems, Vol. 104, No 10, (October 1985) pp. 2805-2814.
7. R.A. Stevens, D.T. Rizy, and S.L. Purucker, "Performance of Conventional Power Flow Routines for Real-Time Distribution Automation Applications" , Proceedings 18th Southeastern Symposium on System Theory, IEEE Computer Soc., 1986, pp. 196-200.
8. R.P. Broadwater, A. Chandrasekaran, C.T. Huddleston, and A.H.Khan, "Power Flow Analysis of Unbalanced Multiphase Radial Distribution Systems", Electric Power System Research, 14, 1988 pp. 23-33.
9. G.X. Luo and A. Semlyen, "Efficient Load Flow for Large Weakly Meshed Networks", IEEE Transactions on Power Systems, Vol. 5, No. 4, (November 1990) pp. 1309-1316.
10. Renato Cespedes G., "New Method for the Analysis of Distribution Networks" IEEE Transactions on Power Delivery, Vol. 5, No 1, (January 1990) pp. 391-396.
11. Tasi-Hsiang Chen, Mo-Shing Chen, Kab-Ju Hwang, Paul Kotas and Elie A. Chebli, "Distribution System Power Flow Analysis- A Rigid Approach", IEEE Transactions on Power Delivery, Vol. 6, No. 3, (July 1991) pp. 1146-1152.
12. William H. Kersting and David L. Mendive, "An Application of Ladder Theory to the Solution of Three-Phase Radial Load-Flow Problems", IEEE Power Engineering Society Winter Meeting and Telsa Symposium, New York, New York, (January 1975) Paper No. A 76 044-8.

13. Robert P. Broadwater, Jeffery C. Thompson, and Thomas E. McDermott, "Pointers and Linked Lists in Electric Power Distribution Circuit Analysis", Proceedings of 1991 IEEE PICA Conference, Baltimore, Maryland, (May 7 1991) pp. 16-21.
14. 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 Transactions on Power Systems, Vol. 3, No. 2, (May 1988) pp. 753-762.
15. Daniel J. Tylavsky and Laxmi Gopalakrishnan, "Identifying Modeling Errors In Mine Electrical Power Flow Input", IEEE Industry Applications Society Annual Meeting, 1989, Vol. 2, pp. 1509-1515.
16. G.T. Heydt and W.M. Grady, "A Z-Matrix Method for Fast Three Phase Load Flow Calculations", Proceedings of the Power Industry Computer Applications Conference, (June 1973) pp. 168-173.
17. R.P. Broadwater, Jeffery Thompson, Mike Ellis, Harry Ng, Nand Singh and Darrell Loyd "Application Programmer Interface For The EPRI Distribution Engineering Workstation", Paper No. WM94-308 (To be published in IEEE Transactions).
18. R. Broadwater, P. Dolloff, M. Ellis, N. Singh, T. Corbin, and H. Ng "EPRI Distribution Engineering Workstation Design Concepts", EPRI CON Conference Proceedings, Scottsdale, Arizona Dec. 9-11, 1992.
19. "Test Plan for Distribution Engineering Workstation Phase I Applications", May 1994, EPRI.
20. Lee W. Johnson and R. Dean Riess, **Numerical Analysis**, 2nd Edition Addison-Wesley, Reading Massachusetts, 1982, pp. 44..
21. IEEE Distribution Planning Working Group, "Radial Distribution Test Feeders", IEEE Transactions on Power Systems, Aug. 91, pp. 975-985.
22. John Zaborszky and Joseph W. Rittenhouse, **Electric Power Transmission**, The Rennsselaer Bookstore, Troy, New York, 1969, pp. 18-21.
23. M. Harry Hesse, Notes For Power Engineering Analysis Course 34.681, Rensselaer Polytechnic Institute, Fall 1983.
24. Carol S. Cheng and Dariush Shirmohammadi, " A Three-Phase Power Flow Method For Real-Time Distribution System Analysis", Paper No. 94 SM 603-1 PWRS ( To be presented at 1994 Summer Power Meeting).
25. William D. Stevenson, Jr. **Elements of Power Systems Analysis**, 4th Edition, McGraw Hill Book Company, New York, 1982, p 282.
26. R. Berg, Jr., E.S. Hawkins, W.W. Pleines, "Mechanized Calculations of Unbalanced Load-Flow on Radial Distribution Circuits", IEEE Transactions on Power Apparatus and Systems, Vol. 86, (April 1967), pp. 415-421.

27. Allan Greenwood, **Electrical Transients in Power Systems**, John Wiley and Sons Inc. New York 1971 p 6.
28. "Performance Tests: RISC Workstations", PC Magazine, May 31, 1994, p 150.

## VITA

Mr. Ellis is a Licensed Professional Engineer in Virginia. He has been an Instructor in Electrical Engineering at Virginia Polytechnic and State University. He has also been an Assistant Professor in the Electronics Engineering Technology Department at Weber State University.

He graduated Magna Cum Laude from Brigham Young University in 1983. He received his Masters Degree in Electrical Power Engineering from Rensselaer Polytechnic Institute in 1984.

*Mike V Ellis*