A Software Framework for Advanced Power System Analysis: Case Studies in Networks, Distributed Generation, and Distributed Computation

by

Fangxing Li

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

Computer Engineering

Robert P. Broadwater, Chairman
Walling Cyre, Co-advisor
Terry Herdman
Yilu Liu
Binoy Ravindran

April, 2001
Blacksburg, Virginia USA

Keywords: Framework, Software Architecture, Composite Cable, Distributed Resource, Distributed Computing, Power Distribution System

Copyright © 2001, Fangxing Li
A Software Framework for Advanced Power System Analysis: Case Studies in Networks, Distributed Generation, and Distributed Computation

by

Fangxing Li

Dr. Robert P. Broadwater, Chairman
Bradley Department of Electrical Engineering

(ABSTRACT)

This work presents a software framework for power system analysis, PowerFrame. It is composed of four layers. This four-layer architecture is designed for extensibility and reusability so that more complex power system problems can be tackled within the architecture.

In the context of PowerFrame, this work explores complex power system problems. Included in these problems are parallel-placed cables with multiple conductors, and distributed resources operating in unbalanced power distribution systems. Mathematical models are derived. Errors between more exact models and conventional approaches are presented.

PowerFrame is also designed to handle distributed computation for intensive power system calculations on multiple, networked computers. Distributed power flow algorithms are presented. Tests on Ethernet LANs show the feasibility of distributed computation under current computer network bandwidth.
Acknowledgments

Special thanks to Dr. Robert Broadwater for giving generously of his valuable time, intellect and friendship.

I would like to thank Electric Distribution Design (EDD) Inc. for providing facilities to finish this research work, and Electric Power Research Institute (EPRI) Distribution Engineering Workstation (DEW) for benchmark analysis of the power flow calculations.

Thanks are also due to Dr. Walling Cyre, Dr. Terry Herdman, Dr. Yilu Liu, and Dr. Binoy Ravindran for serving on my advisory committee.

I am indebted to my wife, Qicheng Zhou, for her love, understanding and support during my pursuit of this degree.
Contents

Figures ....................................................................................................................... vii
Tables............................................................................................................................ x

Chapter 1  Introduction ........................................................................ 1
1.1 Problems to be Investigated...........................................................................1
  1.1.1 Object-oriented Software Engineering in Power Systems ....................... 1
  1.1.2 PowerFrame: Composite Cables ................................................................. 3
  1.1.3 PowerFrame: Distributed Resources ............................................................ 6
  1.1.4 PowerFrame: Distributed Computing ......................................................... 7
1.2 Literature Review .......................................................................................... 9
  1.2.1 Object-oriented Analysis and Design .......................................................... 9
  1.2.2 Object-oriented Software Engineering in Power Systems ....................... 11
  1.2.3 Cable Impedance ......................................................................................... 13
  1.2.4 Distributed Resources ............................................................................... 14
  1.2.5 Distributed Computing .............................................................................. 14
1.3 Summary...................................................................................................... 15

Chapter 2  Architecture of PowerFrame......................................................... 17
2.1 Frameworks ................................................................................................ 17
  2.1.1 Definition of Frameworks .......................................................................... 17
  2.1.2 Horizontal and Vertical Frameworks ............................................................ 18
  2.1.3 A Layered Architecture for Frameworks ..................................................... 19
2.2 Concept of PowerFrame .............................................................................. 22
  2.2.1 About PowerFrame ..................................................................................... 22
  2.2.2 System Requirements.................................................................................. 22
  2.2.3 The Four-layer Architecture of PowerFrame .............................................. 24
2.3 Architectural Design of PowerFrame ......................................................... 25
  2.3.1 Element Layer ............................................................................................ 26
  2.3.2 Iterator Layer .............................................................................................. 31
  2.3.3 Algorithm Layer ........................................................................................ 38
  2.3.4 DistributedAlgorithm Layer ..................................................................... 40
  2.3.5 PowerFrame with All Four Layers .............................................................. 43
2.4 Summary...................................................................................................... 43

Chapter 3  Object-Oriented Model for Composite Cables with
Accurate Impedance Calculation ........................................................................... 45
3.1 Composite Cables ...................................................................................... 45
3.2 Exact Method and Approximate Method ................................................. 48
  3.2.1 Exact Calculation ....................................................................................... 48
  3.2.2 Approximate Calculation .......................................................................... 53
  3.2.3 Comparing Approximate and Exact Methods ......................................... 55
### Chapter 3

3.2.4 Parallel Conductors of Different Types ................................................................. 59

3.3 Object-oriented Design for Modeling Cables within PowerFrame .........................61
   3.3.1 A Sub-package to Model Cables ................................................................................. 59
   3.3.2 Composite Pattern to Model Cables with Exact Impedance Calculation ................. 62

3.4 Summary ..............................................................................................................68

### Chapter 4

Extending PowerFrame to Model Distributed Resources 70

4.1 Distributed Resources ..........................................................................................70

4.2 Mathematical Models of Distributed Resources ......................................................71
   4.2.1 Constant Voltage Source ......................................................................................... 71
   4.2.2 Voltage-Dependent Current Source ............................................................................ 77
   4.2.3 Impacts on Imbalance from Two Models ............................................................... 88

4.3 Extending PowerFrame to Model Distributed Resources ....................................89
   4.3.1 DistRes: An Extended Module of PowerFrame ......................................................... 89
   4.3.2 Interactions with PowerFrame ............................................................................... 91
   4.3.3 Why Interfaces are Sufficient for Extensions ......................................................... 94

4.4 Summary ..............................................................................................................95

### Chapter 5

Distributed Algorithms for Radial and Looped Power Flows 96

5.1 A General Approach to Distributed Algorithms for Tree-based System ............97
   5.1.1 Controller and Workers ............................................................................................. 97
   5.1.2 General Approach .................................................................................................. 98
   5.1.3 Assumption of Tree Partition .................................................................................... 100

5.2 Distributed Algorithm for Radial Power Flow ..................................................101
   5.2.1 Review of Sequential Algorithm based on Tree Approach .................................... 101
   5.2.2 Design of Distributed Algorithm for Radial Power Flow ...................................... 104
   5.2.3 Convergence, Complexity and Efficiency of the Distributed Algorithm for Radial Power Flow 108
   5.2.4 Test Results ............................................................................................................. 111

5.3 Distributed Algorithm for Looped Power Flow ................................................115
   5.3.1 Sequential Algorithm based on Sensitivity Matrix ............................................... 115
   5.3.2 Design of Distributed Algorithm for Looped Power Flow .................................... 116
   5.3.3 Convergence, Complexity and Efficiency of Distributed Algorithm for Looped Power Flow 117
   5.3.4 Test Results ............................................................................................................. 119

5.4 Summary ............................................................................................................122

### Chapter 6

Conclusions ................................................................................................................ 123

6.1 Concluding Remarks ............................................................................................ 123

6.2 Contributions ....................................................................................................... 124

6.3 Future work ......................................................................................................... 126

Reference 128
Vita 132
Figures

Figure 1.1 A Framework Containing Reusable, Object-oriented Design ...... 2
Figure 1.2 Impedance Matrix Reduction................................................. 4
Figure 1.3 A Composite Cable Containing Two Sets of Three-phase Conductors and One Independent Neutral Return in a Duct System of Nine Ducts........................................................... 5
Figure 1.4 Side View of the Composite Cable in Figure 1.3 ................. 5
Figure 1.5 Distributed Resources in a Power Distribution System......... 7
Figure 1.6 Workstations Connected in a LAN ................................. 8
Figure 2.1 A Diagram of a Layered Framework and its Associated Applications ................................................................. 21
Figure 2.2 Architecture of PowerFrame ............................................. 25
Figure 2.3 Physical Model and Circuit Laws for Electric Elements ........ 27
Figure 2.4 Two Representations of a Radial Power Distribution System... 27
Figure 2.5 Interfaces for Element Layer ........................................... 28
Figure 2.6 A Sketchy Diagram of Element Layer .............................. 31
Figure 2.7 Iterator Layer and ITraverse interface .............................. 35
Figure 2.8 Graph Classes Containing the Structural Information of Graphs ........................................................................... 35
Figure 2.9 Iterator Classes................................................................. 37
Figure 2.10 Iterator Layer and ITraverse Interface ............................ 38
Figure 2.11 Algorithm Layer ............................................................... 39
Figure 2.12 DistributedAlgorithm Layer and IMethodInvoker Interface .... 42
Figure 2.13 Four-layer Architecture of PowerFrame .......................... 44
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Two Circuits with Concentric Neutral Cables in Separate Ducts with a Separate Grounded Neutral</td>
<td>46</td>
</tr>
<tr>
<td>3.2</td>
<td>Side View of Conductors Shown in Figure 3.1</td>
<td>47</td>
</tr>
<tr>
<td>3.3</td>
<td>Configuration of Two Phase Conductors with Concentric Neutrals, where c = 6</td>
<td>50</td>
</tr>
<tr>
<td>3.4</td>
<td>Percentage Error of R₀ Approximate Calculation for Both Kron and Neutral Return Current (NRC) Reductions</td>
<td>57</td>
</tr>
<tr>
<td>3.5</td>
<td>Percentage Error of X₀ Approximate Calculation for Both Kron and Neutral Return Current (NRC) Reductions</td>
<td>57</td>
</tr>
<tr>
<td>3.6</td>
<td>Results for R₀ by Exact Method for Both Kron and Neutral Return Current (NRC) Reductions</td>
<td>58</td>
</tr>
<tr>
<td>3.7</td>
<td>Results for X₀ by Exact method for Both Kron and Neutral Return Current (NRC) Reductions</td>
<td>58</td>
</tr>
<tr>
<td>3.8</td>
<td>Voltage Drop Study for Parallel Circuits with Different Cable Types</td>
<td>60</td>
</tr>
<tr>
<td>3.9</td>
<td>The Element Layer with Cable sub-package</td>
<td>62</td>
</tr>
<tr>
<td>3.10</td>
<td>Class Hierarchy of Cable Classes (1)</td>
<td>65</td>
</tr>
<tr>
<td>3.11</td>
<td>Object Diagram for a Composite Cable</td>
<td>66</td>
</tr>
<tr>
<td>3.12</td>
<td>Class Hierarchy of Cable Classes (2)</td>
<td>67</td>
</tr>
<tr>
<td>4.1</td>
<td>DR Modeled as Three-Phase Voltage Source in a Circuit with Balanced Loads and Unbalanced Impedances</td>
<td>72</td>
</tr>
<tr>
<td>4.2</td>
<td>DR Terminal Voltage at Node k and Complex Power Flow in Line ij for General Power System</td>
<td>75</td>
</tr>
<tr>
<td>4.3</td>
<td>Distributed Resource Modeled as Voltage-Dependent Current Source</td>
<td>81</td>
</tr>
<tr>
<td>4.4</td>
<td>Voltages and Power Flows for Line with DR at Ending Node</td>
<td>84</td>
</tr>
<tr>
<td>4.5</td>
<td>Phase Voltage Relationships for Power Line Shown in Figure 4.4</td>
<td>84</td>
</tr>
<tr>
<td>4.6</td>
<td>Accumulation of Voltage Change in DR’s Upstream Lines</td>
<td>87</td>
</tr>
</tbody>
</table>
Figure 4.7 Extended *Element* Layer to Model DRs................................. 90
Figure 4.8 Interaction of *VoltDepCurrDR* with other Classes .............. 93
Figure 5.1 Ethernet LAN with Multiple Workstations............................ 97
Figure 5.2 General Approach of Distributed Computing Model for a  
Tree-based System ........................................................................... 100
Figure 5.3 Job Assignment .................................................................... 101
Figure 5.4 A Tree with 7 Elements....................................................... 104
Figure 5.5 Sequential Diagram for Distributed Algorithm of Radial  
Power Flow ........................................................................................ 106
Figure 5.6 Testing System ...................................................................... 112
Figure 5.7 Efficiency Comparison of Distributed Algorithms of Radial  
Power Flow and Looped Power Flow................................................. 121
Tables

Table 3-1 Sequence Impedance Results with Kron Reduction ..................... 54
Table 3-2 Sequence Impedance Results with Neutral Return Current Reduction ................................................................................. 55
Table 3-3 Conductor Configuration............................................................................................................................. 55
Table 3-4 Sequence impedance calculated from Actual and Approximate methods with Kron and NRC reductions........... 59
Table 3-5 Power Flow Results for a Feeder in Figure 7............................. 60
Table 4-1 DR Real Power Flows as a Percentage of the Total Load for Phases A, B, and C and Real Power Flow Imbalance Versus Percentage Change in DR Source Voltage Magnitude for Circuit of Figure 4.1................................. 74
Table 4-2 DR Real Power Flows as a Percentage of the Total Load for Phases A, B, and C and Real Power Flow Imbalance Versus Change in DR Source Voltage Angle for Circuit of Figure 4.1 ................................................................................. 74
Table 4-3 Loads for Circuit Shown in Figure 4.3........................................ 81
Table 4-4 Customer Level Phase Voltages for Circuit Shown in Figure 4.3 With and Without DR Operating ............................. 82
Table 5-1 Running Time of the Sequential Algorithm for Radial Power Flow with 1 processor ............................................................ 113
Table 5-2 Running Time of the Distributed Algorithm for Radial Power Flow with 2 Processors .......................................................... 113
Table 5-3 Running Time of the Distributed Algorithm for Radial Power Flow with 4 Processors .......................................................... 114
Table 5-4 Running Time of the Distributed Algorithm for Radial Power Flow with 8 Processors .......................................................... 114
Table 5-5 Speed-up and Efficiency of the Distributed Algorithm for Radial Power Flow ........................................................................... 114
Table 5-6 Running Time of the Sequential Algorithm for Looped Power Flow with 1 processor................................................. 119

Table 5-7 Running Time of the Distributed Algorithm for Looped Power Flow with 2 Processors............................................... 120

Table 5-8 Running Time of the Distributed Algorithm for Looped Power Flow with 4 Processors............................................... 120

Table 5-9 Running Time of the Distributed Algorithm for Looped Power Flow with 8 Processors............................................... 120

Table 5-10 Speed-up and Efficiency of the Distributed Algorithm for Looped Power Flow ............................................................... 120
1.1 Problems to be Investigated

1.1.1 Object-oriented Software Engineering in Power Systems

Object-oriented Software Engineering, including analysis, design, implementation, and testing, has been widely accepted and has gained great success [1-14]. This revolution in the software industry is making it easier for people to write, distribute, reuse and extend existing software packages. This revolution also has a significant impact on software engineers working in power systems.

In the past several decades many software packages have been developed for electric power systems. Unfortunately, most of them were procedure-oriented. This made the software unsuitable for reuse and extension when the power system problems and resulting application software became more complicated. Considering this, efforts have been made to push power system software into the new technology of object-oriented development [15-20].

Object-oriented programming has improved software development for power system analysis. However, there are many repeated works in software development, particularly in the design phase [15-20]. An
approach is needed to achieve reusability and extensibility of design. Then application developers can reuse design as well as implementation in this specific domain-- power system analysis.

A solution to obtain extensible and reusable designs for power system analysis is a software framework. A framework aims to reuse design as well as implementation [10-14]. It could have multiple implementations, but only one design. Hence, an application extended from the framework will reuse the unified design as well as a specific implementation of this framework. Figure 1.1 illustrates this idea.

![Figure 1.1 A Framework Containing Reusable, Object-oriented Design](image-url)
Our work here focuses on architecture design issues of such a reusable framework, and in developing a C++ prototype for the framework. This framework, PowerFrame, is an object-oriented framework for power system analysis. Although power systems contain power transmission and distribution systems, this work addresses on power distribution system. The term power systems is referred to as power distribution systems by default.

Within the umbrella of PowerFrame, case studies in cable impedance, distributed resources, and parallel distributed computing will demonstrate the following aspects of the framework:

- Object-oriented modeling of composite cables with a simple, common interface
- Extensible design for modeling new elements such as distributed resources
- Design of distributed algorithms for power flows.

1.1.2 PowerFrame: Composite Cables

As a successful framework, PowerFrame should be capable of addressing problems that have been too complicated to be modeled in a straightforward and reusable way in the past. The cable impedance calculation is such a problem. It is also fundamental to power system analysis.

In the past, an approximate method has commonly been used to model cables. A 4x4 impedance-matrix model has been developed for a cable, which consists of three phase conductors and a neutral return [21]. There
are two common approaches to reduce the 4x4 impedance-matrix to a 3x3 impedance-matrix [21]. The reduction approaches are Kron reduction and Neutral Return Current (NRC) reduction. This reduction is shown in Figure 1.2.

\[
\begin{bmatrix}
Z_{aa} & Z_{ab} & Z_{ac} & Z_{an} \\
Z_{ba} & Z_{bb} & Z_{bc} & Z_{bn} \\
Z_{ca} & Z_{cb} & Z_{cc} & Z_{cn} \\
Z_{na} & Z_{nb} & Z_{nc} & Z_{nn}
\end{bmatrix}
\xrightarrow{\text{Kron/NRC reduction}}
\begin{bmatrix}
Z'_{aa} & Z'_{ab} & Z'_{ac} \\
Z'_{ba} & Z'_{bb} & Z'_{bc} \\
Z'_{ca} & Z'_{cb} & Z'_{cc}
\end{bmatrix}
\]

**Figure 1.2 Impedance Matrix Reduction**

The expansion of urban power distribution systems causes networks to be pushed closer to limits. Utilities need to reevaluate cable network capabilities using accurate impedance models, particularly for composite cables.

A composite cable consists of multiple sets of three-phase or single-phase cables, which are placed in parallel with like-phase-terminals tied together. A typical configuration for a composite cable is shown in Figure 1.3. In this figure, there is a duct array containing nine (3x3) square ducts. Each of the upper-right and bottom-left ducts contains a set of three-phase, concentric-neutral cables. The central duct contains an independent neutral. The terminals of the three-phase cables are tied phase-by-phase at each end of the duct run. All neutral conductors, including the six concentric neutrals and one independent neutral, are also tied together as a common neutral at each end of the duct run.

Figure 1.4 depicts the side view of this composite cable. Conductors 1, 2, and 3 correspond to A, B, and C phase conductors in the upper-right duct,
respectively. Conductors 4, 5, and 6 correspond to A, B, and C phase conductors in the bottom-left duct, respectively. Conductors 7-12 are the concentric neutral conductors in both of the upper-right and bottom-left ducts. Conductor 13 is the independent neutral in the central duct.

To model this composite cable, the traditional approach approximates the impedance of all circuits by ignoring the mutual coupling between
conductors placed in two different ducts. A more accurate, object-oriented approach will be proposed to model this composite conductor. This approach will consider all mutual couplings between any two conductors.

1.1.3 **PowerFrame: Distributed Resources**

*PowerFrame* should be extensible such that new device models may be incorporated. It should be possible to attach a new software module to the framework without any impact or modification or recompilation of the existing software. In this work, Distributed Resources will be used as a testbed for extensibility of *PowerFrame*.

Distributed Resource (DR) generation comes in many forms including gas turbine driven synchronous generators, wind powered induction generators, fuel cells with inverter circuitry, and others. They are usually connected into a power distribution system near heavy load sites, as shown in Figure 1.5.

The use of DR generation is projected to grow. This growth is due to cost reductions available with DRs. The cost reductions may be the result of released system capacity or just reductions in generation costs at peak conditions.
Distributed resources are modeled as constant PQ sources in previous works [22-23]. However, this model may not be sufficient for unbalanced system analysis. There are other steady-state models for DRs. It is necessary to evaluate the effects of different models for DRs. In addition, previous DR studies are performed with the assumption of a balanced power system [22-23]. It is desired to investigate DRs operating under unbalanced power distribution systems, because imbalance is the nature of power distribution circuits.

1.1.4 *PowerFrame*: Distributed Computing

Since parallel and distributed computing can solve power system calculations much faster than sequential computing, it is helpful if *PowerFrame* supports distributed computing algorithms. Hence,
*PowerFrame* needs to provide an abstraction of distributed algorithms for power system analysis.

Parallel and distributed computing are usually used to solve large-scale or time-critical problems [24-25]. It has also been applied to solve power system analysis calculations [26-31].

Parallel computing is usually referred to as computing carried out on a dedicated supercomputer with multiple processors and shared memory. An example of a parallel supercomputer is Sequent Symmetry S81 [27-29]. Distributed computing is usually thought of as a less “coupled” computing carried out among multiple, separate machines without shared memory. For example, computing with workstations connected in an Ethernet Local Area Network (LAN) is distributed computing. Figure 1.6 depicts the connection of an Ethernet LAN. Since there is no shared memory and central control unit, multi-instruction and multi-data is the architecture of distributed computing [24].

![Figure 1.6 Workstations Connected in a LAN](image)

Figure 1.6 Workstations Connected in a LAN
A power distribution system model may conceivably contain millions of components. Many computations are time consuming and/or time critical, such as network reconfigurations, optimizations or real-time power flow. These computations, usually running in one machine, have to sacrifice accuracy to gain speed. However, with parallel and distributed computing, large-scale problems can be solved faster and with more accuracy. Here a distributed algorithm for power flow will be designed and prototyped, which utilizes multiple processors connected in an Ethernet Local Area Network (LAN).

1.2 Literature Review

1.2.1 Object-oriented Analysis and Design

Object-oriented software development has four major phases, which are object-oriented analysis (OOA), object-oriented design (OOD), object-oriented implementation (OOI), and object-oriented testing (OOT). In the OOA phase a set of objects are determined for a problem domain. In the OOD phase the detailed behaviors and properties of the objects are determined that are necessary for the objects to interact to solve the problems at hand. In the OOI phase the design is translated into code. In the OOT phase the faults and errors in OO systems are identified. However, in most real world developments, both OOA and OOD continue to some extent in the OOI and OOT phase. Here only literature on OOA and OOD are reviewed.
The major object-oriented analysis and design methods are the Booch method [1], Rumbaugh method [2], Jacobson method [3], Coad and Youdon method [4], and the Wirfs-Brock method [5]. Booch, Rumbaugh and Jacobson have collaborated to combine the best features of their individual works into a unified method, which utilizes the Unified Modeling Language (UML) [6-7].

With UML for OOA [6-7], a system is represented using five different views that describe the system from distinctly different perspectives. These views are User, Structure, Behavioral, Implementation, and Environmental Model Views.

With UML for OOD [6-7], two major activities, system design and object design, are carried out. System design addresses the software architecture. Object design focuses on a description of objects and their interactions with one another.

Like many engineering design processes, OOD repeatedly utilizes design patterns to achieve reusability. Each design pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution many times over, without ever doing it the same way twice [8-9]. Reference 9 presents 27 object-oriented design patterns categorized into three groups: creational patterns, structural patterns, and behavioral patterns.

A software framework is a set of cooperating classes that make up a reusable design for a specific class of software [10-14]. Like design patterns, frameworks emphasize reusability of design. However, frameworks are a reusable foundation working with a certain domain, while design patterns
are smaller, more general units. A framework may use one or more design patterns, but the reverse is never true [9].

1.2.2 Object-oriented Software Engineering in Power Systems

A number of papers [15-20] have been published concerning software developments in power systems. References [15-17] have placed more emphasis on implementation than on the analysis and design of software. They are reviewed in part A of this section. More recent papers [18-20] have placed emphasis on the analysis and design of the software. They are reviewed in part B.

A) Object-oriented Programming in Power Systems

In Reference 15 the authors discuss an object-oriented model for power distribution systems, referred to as a Distribution Circuit Object Model. The application of the object-oriented concepts of aggregation, association and inheritance are presented. However, the examples are limited to the modeling of components that make up the power system. Furthermore, the focus of the paper is on the attributes of component modeling and not on the behaviors.

In Reference 16 the authors discuss mathematical models of large-scale networks containing FACTS-controlled branches. Object-oriented, UNIX based software for implementing the mathematical models is discussed.
Similar to Reference 15, this paper considers power system component models as objects. However, it makes an advance over Reference 15 in that it also considers the power flow calculation itself as an object.

Reference 17 presents an actual implementation based upon object-oriented techniques. Transmission planning is addressed in their work. Similar to Reference 16, both power system components and power system analysis are modeled as objects. Aggregation and polymorphism are used in illustrating the advantage of the object-oriented approach.

References 15-17 have applied object-oriented techniques to very specific problems.

B) Object-oriented Design in Power Systems

More recently, papers [18-20] have been published which discuss object-oriented software analysis and design for power systems.

In Reference 18 the authors introduce design patterns and demonstrate the potential of applying design patterns in power system software. In their paper they consider 2 design patterns, the composite pattern and the proxy pattern.

Reference 19 presents a set of objects, including methods and properties, which provide an example for the way in which a standard may be developed for power system distribution software. COM and CORBA are mentioned, and it is indicated that the standard could be implemented using
either of these technologies. Other software tools are also considered including, XML, SGML, MAST, and others.

The basic idea in Reference 19 is to provide power industry computer users “plug compatible” software components. The authors consider both fat and minimal interfaces for these components. A fat interface is one which has many behaviors, whereas a minimal interface has few behaviors. The authors favor the minimal interface for power system standards and do not address employing multiple interfaces.

Reference 20 discusses an object-oriented design for power system applications. Power system apparatus, sparse matrices, and power flow are considered in this paper. However, the authors do not discuss whether and why the designs are reusable, extensible and portable in the domain of power system analysis. Hence, the concern of this paper is toward a specific application, not for all applications in a domain. Also, this paper addresses a power transmission system. A sparse matrix approach is used to represent the power transmission system.

1.2.3 Cable Impedance

Reference 21 discusses the equivalent geometric spacing for a three-phase concentric cable. Mutual coupling between conductors is calculated to obtain a 4x4 matrix. Kron and Neutral Return Current (NRC) reductions are used to compress the 4x4 impedance matrix into a 3x3 matrix.
1.2.4 Distributed Resources

Reference 22 deals with issues related to existing DR interconnection practices. An investigation of eleven utilities and industry interconnection standards was performed to identify the key requirements for a DR connection. The results of this investigation led to the development of a unified approach for determining interconnection requirements. This work models DRs as constant current sources.

Reference 23 considers many aspects of DRs in distribution systems, including protection, harmonics, transients, voltage and frequency control. Power flow studies were based on the positive sequence model of the distribution circuits analyzed. The DRs themselves were represented as constant PQ sources.

1.2.5 Distributed Computing

Parallel and distributed computing has been applied in many scientific computations such as weather forecasting and nuclear simulations [24-25]. It has also been applied into power system analysis [26-31]. Research on optimal power flow (OPF) has been carried out in high-performance parallel computation studies with shared memory [27-29]. All of these works address optimal power flows in transmission systems, based on a matrix approach. Moreover, these works are based on parallel supercomputers such as the Sequent Symmetry S81. They are dedicated and expensive, though they have low communication delay.
Distributed computing in LAN workstations costs much less than supercomputers, while gaining performance improvements over a single computer. In recent years, distributed implementations for Optimal Power Flow in transmission systems have been discussed [30-31]. Their work does not require a centralized database to hold some critical data for processor communications. Instead of using shared memory, a scheme of message passing was used.

1.3 Summary

The following are issues that previous works did not investigate and this work will address.

- No framework-based approach has been proposed for power system software development. Since a framework defines the software architecture for all applications in a certain domain, design reusability can be obtained. Hence, it will be beneficial to design a software framework for power distribution system analysis. This work will present a high-level, architectural design for the framework.

- There is no consideration of coupling for composite cable impedance calculations. Also, there is no software that internally considers this complicated cable model. An exact model needs to be derived and incorporated, object-orientedly, into the proposed framework.
• Recent research work does not address three-phase model and imbalance issues for Distributed Resources. However, due to the characteristics of imbalance in modern power distribution systems, those issues should be addressed to gain more accuracy. Also, these unbalanced DR models will serve as a test bed for extensibility of the proposed framework. A three-phase, unbalanced DR model as an extension of the framework will be investigated.

• Previous investigations in parallel and distributed computing are based on a matrix approach for Optimal Power Flow in power transmission systems. There is no discussion on power distribution systems. Are power distribution systems suited for distributed computing, as their names imply? Distributed algorithms for power flows will be investigated in this work.

The proposed investigations are summarized as follows:

• A high-level design of a software framework for power distribution system analysis with a layered architecture;

• An exact impedance model of composite cables which is a complex problem within the framework;

• A three-phase, unbalanced model of Distributed Resource generations designed as an extension of the framework;

• Distributed Algorithms of Radial and Loop Power Flow calculations that make the framework suited for a distributed computation environment.
Chapter 2
Architecture of PowerFrame

The work in this chapter addresses the system level design of a framework for power distribution system analysis, referred to as PowerFrame. A four-layer architecture is proposed to present the internal structure of PowerFrame.

2.1 Frameworks

2.1.1 Definition of Frameworks

A framework is a set of cooperating classes that make up a reusable design for a family of related problem domains [8,10-14]. It is also referred to as an application framework or a software framework. As its definition implies, a framework is sometimes called a class library.

A framework dictates a fundamental architectural design for applications that are incorporated into the framework. A framework predefines many mundane design parameters and actions so that an application designer can concentrate on the specifics, which make the application unique from other applications in the problem domain, of a given application.
A well-designed framework can make good reuse of design as well as code. Such a framework brings software development to a higher and more abstract level, and by doing so improves productivity and the software development success.

### 2.1.2 Horizontal and Vertical Frameworks

Frameworks can be categorized as vertical or horizontal [41-42]. A horizontal framework provides system level services such as file access or device drivers. It is usually domain-independent and provides support for many other vertical frameworks that lie on top of the horizontal framework.

A vertical framework is more domain-specific. It aims to provide abstractions of attributes and behaviors of a specific problem domain. A vertical framework is usually supported by one or more horizontal frameworks.

An example of a horizontal framework is the Microsoft Foundation Class (MFC) Library [32-33], which provides a platform for Windows-based application developers. An example of a vertical framework is the proposed PowerFrame, which aims to provide a platform for application developers in power distribution systems.
2.1.3 A Layered Architecture for Frameworks

A) Layers

A framework can be constructed with many different types of architectures. Here a layered architecture is proposed.

This layered architecture has a strict top-down dependency. Any layer should depend on only its lower layers. A lower layer in the framework provides more fundamental objects than a higher layer. A lower layer may be more reusable than a higher layer.

The dependency is implemented through interfaces. In other words, each layer provides an interface and depends on those interfaces provided by lower layers. Ideally, an interface is the sufficient and necessary behaviors or responsibilities of a layer in the specific domain.

Figure 2.1 describes a three-layer framework, where Layer2 depends on Layer1 through interface ILayer1. Layer3 depends on Layer2 and Layer1 through interfaces ILayer1 and ILayer2, respectively.

Here the term “layer” may mean package, module, component or unit, such as used in many other software engineering references. Here we use the term “layer” to describe the strict one-way dependency between higher and lower “packages”, or “layers”.

This top-down dependency can decrease the coupling among software layers, since this is a one way dependency. Lower modules can always be portable without any concern about higher layers.
B) Sub-layers

A problem domain can usually be divided into several sub-domains. Hence, it is beneficial to split a layer into several parallel and independent sub-layers. As indicated in Figure 2.1, Layer3 is divided into three sub-layers. Each handles problems in a specific sub-domain.

Then, as illustrated in Figure 2.1, applications App1 and App2 depend on and utilize sub-layer Layer3.1 (through interface ILayer3). There is no requirement for any knowledge of or involvement with other sub-layers, such as Layer3.2 and Layer3.3.

This approach can optimize the cost since a specific application does not require all sub-layers in the framework. Such dividable layers are often higher layers since they work closer to applications than lower layers.
Figure 2.1 A Diagram of a Layered Framework and its Associated Applications
2.2 Concept of *PowerFrame*

2.2.1 About *PowerFrame*

The concept of *PowerFrame* is a software framework that abstracts the most common attributes and behaviors in power distribution systems analysis. *PowerFrame* is intended to do for application programmers in power systems what Microsoft Foundation Class (MFC) Library does for Windows programmers. *PowerFrame* assists power system software developers, who are working independently, in developing code that works together in a seamless fashion.

2.2.2 System Requirements

Since *PowerFrame* aims to represent a reusable and extensible framework for power systems, *PowerFrame* does not solve any non-power-system problems, such as file services, graphic user interface (GUI), or database. Those are part of the outside world, with which *PowerFrame* needs to interact.

On the other hand, the framework should capture the expertise of its problem domain so that it is independent of both the original problem and the future problem. For each client to adapt the framework, it can solve existing problems and future problems in a graceful way.
The envisioned characteristics of *PowerFrame* are summarized as follows:

- An extensible approach to model power system elements
- A topology engine that is suitable for handling different topological traverse methods
- An approach to model algorithms for power system analysis with a separation between data and algorithms that operate on data
- An approach for solving a problem with a distributed computing mode, particularly for a large-scale system with hundreds-of-thousands of elements.

Power distribution systems are large systems with more than one million elements for a single, large utility. *PowerFrame* should be able to model common power system elements and their behaviors efficiently from the point of view of circuit laws. *PowerFrame* should also be able to handle topological management since most power system calculations involve topology. In addition, *PowerFrame* should provide an abstraction for many power system algorithms.

With the rapid development in computer networks, communication delay has become relative small in Ethernet LANs. Thus distributed algorithms may greatly shorten computing time over sequential algorithms. *PowerFrame* is designed to provide a high-level abstraction of distributed computing for power system analysis. This can benefit some time-consuming or time-critical calculations such as network reconfigurations and real-time power flows, particularly for a bulk system with up to 1 million elements.
2.2.3 The Four-layer Architecture of *PowerFrame*

*PowerFrame* is designed with the idea of a layered architecture described in Section 2.1.3. It is a four layer framework as depicted in Figure 2.2. The four layers, from bottom to top, are *Element* layer, *Iterator* layer, *Algorithm* layer and *DistributedAlgorithm* layer.

A brief description of these four layers is presented as follows.

- *Element* layer characterizes and models the electric devices or elements in power distribution systems.
- *Iterator* layer models graph representations and traversing methods of power distribution systems.
- *Algorithm* layer presents abstractions of many domain-specific algorithms, such as power flow, short circuit, and reliability calculations.
- *DistributedAlgorithm* layer models distributed versions of algorithms such that *PowerFrame* can work in a distributed computation environment.

In Figure 2.2, the two round rectangles represent two frameworks working on two different machines. They communicate through communication networks, such as LANs or Internet. With the assistance of the *DistributedAlgorithm* layer, the network is transparent to application developers. The underlying communication techniques for the distributed algorithms could be Remote Procedure Call (RPC), Java Remote Method Invocation (RMI), or even primitive TCP/IP protocol.
The DistributedAlgorithm layer is an optional layer if application developers need to solve power system problems only on a single, stand-alone computer. The lower three layers provide sufficient solutions for power system analysis in a stand-alone machine. Hence, application developers only need to reuse the lower three layers in this case.

### 2.3 Architectural Design of PowerFrame

In this work the approach for object-oriented analysis and design (OOAD) is carried out in the following two steps.

1. Common responsibilities, or behaviors, of all domain objects in a layer are summarized. These behaviors consist of the interfaces for this layer.
2. Classes in this layer are identified. Those classes are categorized into inheritance architecture, based on their physical models and their roles in power distribution systems.

2.3.1 **Element Layer**

A) Identifying Interfaces

There are many kinds of electric devices, or elements, in power distribution systems such as cables, transformers, switches and loads. They demonstrate many similarities as well as some specialties. The interface of this layer identifies the common behaviors or interface for all elements.

The model here for elements is a two-port model as shown in Figure 2.3. In other words, each element contains a single upstream port or port 1, and a single downstream port or port 2. The upstream port is closer to the power source, while the downstream port is closer to the ending customer load. Current, voltage and power flow at each port are considered as internal attributes of each element. With this approach, there is no need to use a node-edge-based topological representation, which considers nodes and edges separately. An element-based tree can represent the topology of power distribution systems (This is discussed in detail in Section 2.3.2).

Figure 2.4 illustrates node-edge-based and element-based representations for a simple system. In this figure, part (a) shows a node-edge-based representation, while part (b) shows an element-based tree representation in which the two internal ports are not shown. Again, this will be addressed in Section 2.3.2 where the system-level topological representations are discussed.
Figure 2.3 Physical Model and Circuit Laws for Electric Elements

Figure 2.4 Two Representations of a Radial Power Distribution System
To assist the system-level topological management, the *Element* layer provides an interface, *IConnElem*, to identify each element’s local connectivity. This is shown in Figure 2.5. The second layer, *Iterator*, uses this interface to identify and traverse the whole system. In other words, the *Element* layer is responsible to identify local, element-level topological connections (through the *IConnElem* interface), while the *Iterator* layer is responsible for global, system-level topological connection and traverse. More details of system-level traverse will be discussed in Section 2.3.2.

![Figure 2.5 Interfaces for Element Layer](image)

The *IConnElem* interface is described with Object Management Group (OMG) Interface Description Language (IDL) [43]. (Note that the implementation of *PowerFrame* does not necessarily depend on IDL. It is used here only for readability and understandability.)
The interface \textit{IConnElem} is given in IDL as follows, where \textit{AbstractElem} is defined as the parent class of all element classes.

\begin{verbatim}
interface IConnElem{
    AbstractElem parent();       // parent element
    AbstractElem leftMostChild(); // left most child elem.
    AbstractElem rightSibling();  // right sibling element
}
\end{verbatim}

Here is an example for the outcome of the above methods. For the simple system illustrated in Figure 2.4, methods \texttt{parent()}, \texttt{leftMostChild()} and \texttt{rightSibling()} of element S5 return elements C2, C6 and S3, respectively. For element C2, those methods return T1, S5, and NULL, respectively.

The above analysis is to find the responsibility of each element relative to the aspect of local topological connectivity. On the other hand, each element should demonstrate the knowledge or responsibility of electric circuit laws. That is, each element must follow the circuit laws for calculating current, voltage and power. This is also shown in Figure 2.3. Therefore, circuit laws can be viewed as common responsibilities for all elements. In other words, there should be an interface between the \textit{Element} layer and other software modules, which are involved with circuit law calculations.

Based on the above analysis, in the \textit{Element} layer, all elements need to follow another interface to model circuit laws for two-port elements. Figure 2.5 also illustrates this interface, referred to as \textit{ICktLaws}, for the \textit{Element} layer.
Considering the three-phase model of elements in power distribution systems, the \textit{ICktLaws} interface is given as follows:

\begin{verbatim}
interface ICktLaws{
    Current[3] I();    //current through elements
    Voltage[3] V1();   //voltage at internal port 1
    Voltage[3] VDrop(); //voltage drop across element
    Voltage[3] V2();   //voltage at internal port 2
    Power[3] S1();    //power flow at internal port 1
    Power[3] S2();    //power flow at internal port 2
}
//Assume Current, Voltage, and Power are predefined
//data types handling complex numbers.
\end{verbatim}

**B) Identifying Class Inheritance Hierarchy**

In this step all elements are grouped into a class inheritance hierarchy. \textit{AbstractElem} is an abstract, ultimate super class. Then, elements are grouped into major categories such as cables, transformers, and generators, represented by abstract classes \textit{AbstractXfrm}, \textit{AbstractCable}, and \textit{AbstractGen}. They are further grouped into concrete subclasses. For example, the \textit{AbstractCable} class has subclasses \textit{SinglePhaseCable} and \textit{ThreePhaseCable}. As self-explained by their names, these two subclasses represent single-phase and three phase cables. Figure 2.6 illustrates the model of the class inheritance.

The full design and description for every subclass of \textit{AbstractElem} are beyond the main scope of the architectural design of \textit{PowerFrame}. The key idea in this layer is to find the exact interfaces for all elements, which involves power systems analysis.
2.3.2 **Iterator Layer**

The *Iterator* layer presents an abstraction of topological management to traverse power distribution systems according to certain traversal methods. Here the tree-based topological representation of power distribution systems is discussed in part A. Then, the design of the *Iterator* layer based on the tree approach is described in part B.

**A) Topological Representation with Tree Approach**

Power distribution systems are radial systems (for 80% cases), or weakly-looped systems (for 20% cases). The traditional representation models a system with an incident matrix. However, the *Iterator* layer employs a tree-based topological representation, which views a system as a tree if it is...
radial, or a set of radial trees connected by co-tree elements if it is weakly looped.

The traditional matrix representation is based on traditional power flow calculations, which are derived from a matrix and vector model. This representation employs the edge-node-based model as shown in Figure 2.4(a) in page 27, and models a system with an incident matrix. In this approach, the behaviors at edges and nodes are considered separately.

This work models a system with a tree-based representation. This approach employs the two-port model as described in Section 2.3.1. For a radial system, it can be viewed as a set of connecting elements in a tree. It is certain that nodes in the matrix approach become internal ports of elements in this new approach. Figure 2.4(b) presents a tree representation of the system in Figure 2.4(a). Here each big, ellipse “node” is a two-port element as described in Section 2.3.1. Figure 2.4(b) does not show the internal two ports for simplicity. As a matter of fact, each downstream port of an element is hooked up with every upstream port of all succeeding elements.

For a weakly-looped system, it may contain a few closed, connecting switches which create loops. These switches are referred to here as co-tree switches. The system is weakly looped due to the small number of co-tree switches. By disconnecting co-tree switches, the system is partitioned into a number of radial trees. Hence, a weakly-looped system can be represented by a collection of radial trees and the associated co-tree switches. Therefore, the tree representation can be applied to weakly-looped systems.
This tree representation has advantages over the matrix approach:

- The tree representation has less coupling than the matrix representation. The former has an element model only, while the latter has both node and edge models. Hence, more couplings are logically involved in the matrix representation.

- The tree representation fits the distributed computing model better than the matrix representation. Distributed computing needs to partition a system into several systems for concurrent computing. With the tree representation, it is easy and natural to partition a looped system into several radial trees, because the system is represented with a collection of radial trees and the associated co-tree switches. However, for a matrix representation, additional work must be done to split a system (matrix) into several subsystems (sub-matrices). Chapter 5 discusses details about creating distributed algorithms for the power flow calculation, a power distribution system analysis.

B) Design of the *Iterator* Layer

As in the OOAD approach used in Section 2.3.1, we first explore the interface of this layer. Many power system algorithms depend on this layer to traverse the system according to some traversal methods. Hence, this layer should have an interface, which defines general methods to traverse the tree. Figure 2.7 illustrates the interface, *ITraverse*, provided by the *Iterator* layer. The *ITraverse* interface is defined in IDL as follows.
interface ITraverse{
    AbstractElem firstElem();
    AbstractElem lastElem ();
    AbstractElem nextElem ();
    AbstractElem previousElem ();
    AbstractElem currentElem ();
    AbstractElem[] cotreeElem ();
}

Next, we need to determine how to group and organize the classes in this layer. As shown in Figure 2.8, the AbstractGraph class is the abstract super class of all graph classes. The subclasses Tree and Network represent a radial system and a weekly-looped system, respectively.

As shown in Figure 2.8, the key attributes of Tree and Network are described as follows:

- For the Tree Class:

  root: the root element in the system

  last: the last element in the system

- For the Network class:

  roots: an array of root elements in every radial tree

  lasts: an array of the associate last element in every radial tree

  cotreeElems: an array of co-tree elements.
A set of classes representing and traversing the system topologically

**Figure 2.7 Iterator Layer and ITraverse interface**

<table>
<thead>
<tr>
<th>Tree</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>- root: AbstractElem</td>
<td>- roots: AbstractElem[]</td>
</tr>
<tr>
<td>- last: AbstractElem</td>
<td>- lasts: AbstractElem[]</td>
</tr>
<tr>
<td></td>
<td>- ctreeElems: AbstractElem[]</td>
</tr>
</tbody>
</table>

**Figure 2.8 Graph Classes Containing the Structural Information of Graphs**

In a small software system, it may be common to implement the ITraverse interface within AbstractGraph and its subclasses. However, this may mess up in a large software system with many different algorithms, which may require different traversal methods.
For instance, consider three traversal methods for elements simply connected like a linked list. In the first one, a “next” operation returns the next element. However, based on the semantics of second traversal method, a “next” operation returns the next available sectionalizing element following the current element. In the third traversal method, a “next” operation returns the next available transformer. Then, to handle this within only one class, we have to define such methods:

```
NextBasedOnOrdinaryTraverse();
NextAvailableSwitchingElement();
NextAvailableTransformer();
```

The water will become even muddier if more traversal methods must be implemented in the future.

With the Iterator design pattern [9], a separation is made between the structural representation of a graph and the traversal methods operating on the graph. Figure 2.8 already illustrates the structural representation of graphs. Next the traversal methods are grouped into a set of classes, referred to as the iterator classes.

An abstract iterator class, `AbstractIterator`, is created as the super class for all concrete iterator classes. Different concrete iterator classes are responsible for different traversing approaches. When a new traversal method is needed, a new subclass of `AbstractIterator` is created to implement this new traversal method. There is no need to modify or access the existing iterator classes.
Figure 2.9 describes the iterator classes for power distribution systems. Their names are self-explanatory. This figure shows two key attributes of \textit{AbstractIterator}:

- graph: the reference of the graph which represents the system
- currElem: the reference of a current element during a traverse.

\begin{center}
\begin{tikzpicture}
  \node {AbstractIterator} [rectangle, rounded corners, draw, text width=2cm, align=center] {
    - graph : AbstractGraph *
    - currElem : AbstractElem *
  }
  \node [rectangle, rounded corners, draw, below=of AbstractIterator] {TreelIterator} [rectangle, rounded corners, draw, text width=2cm, align=center] {
    - ForwardTreelIterator
    - BackwardTreelIterator
  }
  \node [rectangle, rounded corners, draw, below=of AbstractIterator] {NetworkIterator} [rectangle, rounded corners, draw, text width=2cm, align=center] {
    - ForwardWLTreelIter
    - BackwardWLTreelIter
  }
\end{tikzpicture}
\end{center}

\textbf{Figure 2.9 Iterator Classes}

The relation between the \textit{AbstractIterator} class and the \textit{AbstractGraph} class is presented in Figure 2.10. It is a many-to-one relation. Every iterator object must know which graph it operates on, while a graph has no knowledge of iterators. In addition, there may be more than one iterator working on a graph. All of the subclasses of \textit{AbstractIterator} and \textit{AbstractGraph} are not shown here.
The *ITraverse* interface is implemented by iterator classes. This layer depends on the *IConnElem* interface provided by *Element* layer. Please refer Figure 2.13 in page 43 for the big picture of *PowerFrame*.

![Figure 2.10 Iterator Layer and ITraverse Interface](image)

### 2.3.3 Algorithm Layer

There are many algorithms in power distribution systems. Each of them solves a different engineering problem, such as power flow, short circuit, and network optimization. The *Algorithm* layer models these algorithms.

What the user of this layer cares about is to execute one or more algorithms. Hence, the interface provided here is simply a method, which executes an algorithm over a graph with the assistance of one or more iterators. This interface, *IExecAlg*, is defined with IDL as follows.
interface IExecAlg{
    long execCalc([in] AbstractIterator[] i,
    [in, out] AbstractGraph g);
    //using iterators to exec a calculation on a graph
}

![Class Diagram](image)

**Figure 2.11 Algorithm Layer**

All algorithms are encapsulated into classes. An abstract class, `AbstractAlgorithm`, is modeled as the super class as shown in Figure 2.11. `AbsPowerFlow` and `AbsShortCircuit` are two subclasses of `AbstractAlgorithm`. `AbsPowerFlow` has two subclasses `RadialPowerFlow` and `LoopedPowerFlow`, which model radial and looped power flow calculations, respectively. `AbsShortCircuit` has two subclasses
RadialShortCircuit and LoopedShortCircuit, which model radial and looped short circuit calculations, respectively.

Since power flow and short circuit calculations are considered as two specific sub-domain problems, two sub-layers are created to package classes modeling different algorithms. Sub-layer PowerFlow contains AbsPowerFlow and its subclasses, and sub-layer ShortCircuit contains AbsShortCircuit and its subclasses. If application developers work on only one sub-domain, they need to purchase or access only one sub-layer. Hence, cost is optimized due to sub-layers.

The classes in the Algorithm layer directly depend on two interfaces to accomplish their tasks. First, it needs ITraverse provided by the Iterator layer to traverse the elements one by one. Then, it needs the ICktLaws interface provided by the Element layer to accomplish the circuit calculations. The Algorithm layer also indirectly depends on the IConnElem interface since the Iterator layer depends on this interface. Please refer to Figure 2.13, which depicts the whole architecture of PowerFrame.

2.3.4 DistributedAlgorithm Layer

The above three layers are sufficient for developing power systems analysis applications in a stand-alone machine. The fourth layer, DistributedAlgorithm, is mounted on top of them to provide an abstraction of distributed computations.

The latest development in computer networks makes distributed computation faster than ever, as the communication bandwidth has been
boosted dramatically. This makes it possible that distributed computing will have great success in power system analysis. However, it is still tedious work to implement distributed computing algorithms. Therefore, if \textit{PowerFrame} provides an abstraction of distributed algorithms, many application developers can avoid repeated work in network programming details and benefit from this high-level abstraction.

Unlike Chapter 5 which presents mathematical descriptions of distributed algorithms, this section presents an architectural design of the \textit{DistributedAlgorithm} layer, which abstracts distributed algorithms from the perspective of a framework designer. This abstraction follows the distributed computing model carried out among multiple computers, which are connected in a network to complete power system analysis calculations.

As a framework for application developers, \textit{PowerFrame} should handle distributed computation gracefully. The computer network should be transparent from the point of view of application developers, or framework users. In other words, they should not struggle with programming details of communication protocols. Hence, an interface to invoke remote methods is provided by this layer. This interface, \textit{IMethodInvoker}, is given in IDL as

\begin{verbatim}
interface IMethodInvoker{
    void invokeCalc ( [in] long calcID );
    void assignTask ( [in] AbstractGraph g);
    void assignTask ( [in] AbstractElem[] e,
                        [in] long elemSize);
    void receiveData( [out] AbstractElem[] elem,
                      [out] long elemSize);
}
\end{verbatim}

As shown in Figure 2.12, this layer defines a parent class \textit{AbsDistributedAlgorithm} and several subclasses, \textit{DistributedPowerFlow} and
DistributedShortCircuit. Since power flow and short circuit are considered as problems in different sub-domain, these two subclasses are packaged into sub-layers of DPF and DSC, respectively. Within the sub-layers, subclasses DistributedPowerFlow and DistributedShortCircuit are further subclassed in the inheriting architecture.

This layer depends on the IExecAlg interface provided by the Algorithm layer, since this layer needs to call some sequential algorithm in the Algorithm layer. Please refer to Figure 2.13 that illustrates the four layers of PowerFrame.

![Figure 2.12 DistributedAlgorithm Layer and IMethodInvoker Interface](image-url)


2.3.5 *PowerFrame with All Four Layers*

The system architecture of *PowerFrame* with four layers is described in Figure 2.13.

For each layer, there could be multiple eligible implementations if they follow exactly the interfaces in the figure. Then, those implementations can be exchangeable without any modification or knowledge of other layers. In addition, a new module may be attached to this layer if it follows the required interfaces. Other layers can use this new module without explicit and pre-compilation knowledge of this module. This makes the framework reusable and extensible.

2.4 **Summary**

A framework aims to reuse design as well as implementation in a specific domain. This chapter presents a layered architecture for frameworks, which has a strict top-down dependency. This layered architecture is applied to design *PowerFrame*, a software framework abstracting the generic problems in power distribution system analysis. An inheritance hierarchy and interfaces provide extensibility and reusability for application development based on *PowerFrame*. 
Figure 2.13 Four-layer Architecture of PowerFrame
Cables are fundamental elements in power systems. *PowerFrame* is designed to be able to model accurate cable impedance. This chapter discusses the approximate model and the exact model of cable impedance first. Then, the accurate model is incorporated into *PowerFrame* with a detailed design. The *Composite* design pattern is employed for the design.

### 3.1 Composite Cables

Power distribution networks in many cities in the United States are becoming relatively old. Due to increased load growth and also the changing geographical location of load centers, a number of utilities in large cities are looking into re-design of these networks. As part of evaluating the re-design, power flow analysis is required. The results of the analysis depend upon the accuracy of the impedances used in the model.

In distribution networks as many as ten circuits may run in parallel, where a circuit consists of three-conductors making up phases A, B, and C. Current practice is to approximate the impedance of $n$ circuits in parallel (where all
conductors are the same) by calculating the impedance of a single circuit and dividing the result by \( n \).

This chapter considers two circuits in parallel. Consider the manhole configuration shown in Figure 3.1, where two sets of three-phase conductors with concentric neutrals are placed in two adjacent round ducts. Each set of three-phase conductor sets is referred to as a circuit. In addition, there is a separate neutral placed in a duct below the circuits. The separate neutral and concentric neutrals are run in parallel. This is the configuration that is analyzed in this chapter.

![Figure 3.1 Two Circuits with Concentric Neutral Cables in Separate Ducts with a Separate Grounded Neutral](image)

The impedance for these circuits will be calculated using a common practice. Results from this calculation will be referred to as the Approximate method. More exact calculations of the impedance of the two circuits will then be performed. This method will be referred to as the Exact
method. Error comparisons between the Approximate and Exact methods are made. Both calculations use the modified Carson’s equations.

In deriving equivalent phase and sequence impedance matrices for both the Exact and Approximate methods, two different matrix reduction approaches are used, which are the Kron and the Neutral Return Current reduction methods [21]. The Kron method should be used where the engineer expects significant earth return currents exist, such as where physical deterioration of the neutral has occurred. The Neutral Return Current method should be used where the engineer wishes to assume that all of the return current flows through the neutral conductors.

Figure 3.2 provides a schematic view of the conductors of Figure 3.1. Figure 3.2 illustrates 13 current paths consisting of six phase conductors numbered 1-6, six concentric neutrals numbered 7-12, and the separate neutral numbered 13. As indicated in Figure 3.2, at each manhole like phase conductors are tied together, such as conductor 1 and 4 are tied together. Also, at each manhole all neutrals, conductors 7-13, are tied together and grounded. Counting the earth return, there are actually 14 current paths.

![Figure 3.2 Side View of Conductors Shown in Figure 3.1](image-url)
In Section 3.2.1, the Exact method is considered. This results in a 13x13 impedance matrix for the system illustrated in Figure 3.1. The Kron and Neutral Return Current reduction methods are then used to reduce this matrix to equivalent 3x3 matrices. After the calculation using the Exact method, the common approximation is then used to obtain the impedance corresponding to Figure 3.1. Section 3.2.2 presents the Approximate method and Section 3.2.3 presents errors between two methods.

### 3.2 Exact Method and Approximate Method

#### 3.2.1 Exact Calculation

The Exact method is performed using the following steps:

1. Use the Modified Carson’s Equations to calculate the elements of the 13x13 impedance matrix, where the \((i,j)\) impedance element multiplies current \(j\) to obtain the voltage drop in line \(i\) due to current \(j\). [21,44]

2. Reduce all neutral return currents to an equivalent neutral return current. Also reduce all like phase currents to an equivalent phase current. These reductions result in a 4x4 impedance matrix.

3. Apply both Kron and Neutral Return Current reduction methods to the 4x4 impedance matrix obtained in Step 2 to obtain 3x3 equivalent phase impedance matrices.

4. Transform the 3x3 phase impedance matrices obtained in Step 3 to sequence impedance matrices.
**A) Primitive Impedance Matrix**

First consider the calculation of the self-impedance and mutual-impedance between any two conductors. Applying the Modified Carson’s Equations we have [21]

\[
Z_{ii} = (r_i + 0.0953) + j0.12134 \left( \frac{1}{GMR_i} + \frac{7.93402}{\text{ln} \frac{1}{GMR_i}} \right) \text{ (\Omega/mile)} \tag{3-1}
\]

\[
Z_{ij} = 0.0953 + j0.12134 \left( \frac{1}{D_{ij}} + 7.93402 \right) \text{ (\Omega/mile)} \tag{3-2}
\]

where

- \( r_i \) = conductor resistance
- \( GMR_i \) = the GMR for conductor \( i \)
- \( D_{ij} \) = spacing between conductors \( i \) and \( j \).

There are 156 values of \( D_{ij} \) to be determined for the 13x13 matrix. Referring to Figure 3.3, \( D_{ij} \) is calculated in one of four ways as follows:

- For a phase conductor to its corresponding concentric neutral, \( D_{ij} = R \)
- For a phase conductor to an adjacent phase conductor, \( D_{ij} = L \)
- For a phase conductor to an adjacent concentric neutral, \( D_{ij} = \sqrt{L^2 - R^2} \)
- For a concentric neutral to an adjacent concentric neutral, \( D_{ij} = L \)

where

- \( R \) = radius of a circle going through the center of the concentric neutral strands
- \( L \) = center-to-center distance between phase conductors
- \( c \) = number of concentric neutral strands.
Equations (3-1) and (3-2) are used to calculate the elements of the 13x13 symmetric primitive impedance matrix. Inverting this matrix gives an admittance matrix, also symmetric, that relates the phase currents to the phase voltage drops as follows:

\[
\begin{bmatrix}
I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & I_7 & I_8 & I_9 & I_{10} & I_{11} & I_{12} & I_{13} \\
\end{bmatrix}
= \begin{bmatrix}
\Delta V_1 & \Delta V_2 & \Delta V_3 & \Delta V_4 & \Delta V_5 & \Delta V_6 & \Delta V_7 & \Delta V_8 & \Delta V_9 & \Delta V_{10} & \Delta V_{11} & \Delta V_{12} & \Delta V_{13} \\
\end{bmatrix}
\]

(3-3)
B) Admittance Matrix Reduction: From 13x13 to 4x4

Equation (3-3) will now be reduced to a 4x4 matrix equation. This will be performed by adding appropriate rows and by making use of voltage drops that are equal in paralleled conductors.

From Figure 3.2 we have

\[
\begin{align*}
I_a &= I_1 + I_4 \\
I_b &= I_2 + I_5 \\
I_c &= I_3 + I_6 \\
I_n &= \sum_{k=7}^{13} I_k
\end{align*}
\] (3-7)

also,

\[
\begin{align*}
\Delta V_a &= \Delta V_1 = \Delta V_4 \\
\Delta V_b &= \Delta V_2 = \Delta V_5 \\
\Delta V_c &= \Delta V_3 = \Delta V_6 \\
\Delta V_n &= \Delta V_7 = \Delta V_8 = \cdots = \Delta V_{13}
\end{align*}
\] (3-11)

Making use of (3-4) – (3-11) in (3-3), we have

\[
\begin{bmatrix}
I_a \\
I_b \\
I_c \\
I_n
\end{bmatrix} =
\begin{bmatrix}
y_{aa} & y_{ab} & y_{ac} & y_{an} \\
y_{ba} & y_{bb} & y_{bc} & y_{bn} \\
y_{ca} & y_{cb} & y_{cc} & y_{cn} \\
y_{na} & y_{nb} & y_{nc} & y_{nn}
\end{bmatrix} \begin{bmatrix}
\Delta V_a \\
\Delta V_b \\
\Delta V_c \\
\Delta V_n
\end{bmatrix}
\] (3-12)

where

\[
\begin{align*}
y_{aa} &= \gamma_{11} + \gamma_{14} + \gamma_{41} + \gamma_{44} \\
y_{ab} &= \gamma_{12} + \gamma_{15} + \gamma_{52} + \gamma_{55} \\
y_{ac} &= \gamma_{13} + \gamma_{16} + \gamma_{43} + \gamma_{46} \\
y_{ba} &= \gamma_{21} + \gamma_{24} + \gamma_{42} + \gamma_{45} \\
y_{bb} &= \gamma_{22} + \gamma_{25} + \gamma_{52} + \gamma_{55} \\
y_{bc} &= \gamma_{23} + \gamma_{26} + \gamma_{53} + \gamma_{56} \\
y_{ca} &= \gamma_{31} + \gamma_{34} + \gamma_{43} + \gamma_{46} \\
y_{cb} &= \gamma_{32} + \gamma_{35} + \gamma_{52} + \gamma_{55} \\
y_{cc} &= \gamma_{33} + \gamma_{36} + \gamma_{63} + \gamma_{66}
\end{align*}
\]
Note that $Y_{ab} = Y_{ac} = Y_{ba} = Y_{ca} = Y_{cb}$, because the configuration in Figure 3.1 is symmetric for all three phases.

**C) Impedance Matrix Reduction: From 4x4 to 3x3**

Rewriting (3-12) in impedance form we have

\[
\begin{bmatrix}
\Delta V_a \\
\Delta V_b \\
\Delta V_c \\
\Delta V_n
\end{bmatrix} =
\begin{bmatrix}
z_{aa} & z_{ab} & z_{ac} & z_{an} \\
z_{ba} & z_{bb} & z_{bc} & z_{bn} \\
z_{ca} & z_{cb} & z_{cc} & z_{cn} \\
z_{na} & z_{nb} & z_{nc} & z_{nn}
\end{bmatrix}
\begin{bmatrix}
i_a \\
i_b \\
i_c \\
i_n
\end{bmatrix}
\]

(3-13)

Also, $Z_{ab} = Z_{ac} = Z_{ba} = Z_{ca} = Z_{cb}$.

Having the 4x4 symmetric impedance matrix shown in (3-13), we can now use two different matrix reduction methods to obtain an equivalent 3x3 phase impedance matrix. The two methods are the Kron Reduction and the Neutral Return Current methods. The Kron Reduction method assumes that half of the return current flows through the earth, whereas the Neutral Return Current method assumes that all of the return current flows through the neutral conductors present. Applying these assumptions results in the following matrix element transformation for Kron Reduction...
\[ z'_{ij} = z_{ij} - \frac{z_{in} z_{jn}}{z_{nn}} \quad (3-14) \]

where \( z'_{ij} = (i,j) \) element in equivalent 3x3 impedance matrix.

Similarly, the matrix element transformation for the Neutral Return Current method is given by

\[ z'_{ij} = z_{ij} + z_{nn} - z_{in} - z_{jn} \quad (3-15) \]

It should be noted that the above reductions only apply to wye connected systems. Both reductions result in a 3x3 equivalent phase impedance matrix which is symmetric, given by

\[
Z'_p = \begin{bmatrix}
  z_s & z_m & z_m \\
  z_m & z_s & z_m \\
  z_m & z_m & z_s \\
\end{bmatrix} 
\quad (3-16)
\]

Applying the symmetrical components transformation to (3-16) gives

\[
Z'_s = \begin{bmatrix}
  z_0 & 0 & 0 \\
  0 & z_+ & 0 \\
  0 & 0 & z_- \\
\end{bmatrix} 
\quad (3-17)
\]

The notation used in (3-16) and (3-17) will be used in Section 3.2.4 when results are presented and compared.

### 3.2.2 Approximate Calculation

In calculating the impedance for the configuration shown in Figure 3.1, the Approximate method assumes that there is no coupling between the two sets of conductors in the separate ducts. Hence, this practice only considers one set of conductors and the independent neutral return, i.e., conductors 1, 2, 3,
7, 8, 9, and 13 in Figure 3.1. The steps used in the Approximate method calculation are:

1. Use the Modified Carson’s Equations to calculate the elements of the 7x7 impedance matrix, where the (i,j) impedance element multiplies current j to obtain the voltage drop in line i due to current j. [21,44]

2. Reduce the neutral return currents of conductors 7, 8, 9, and 13 to an equivalent neutral return current. These reductions result in a 4x4 impedance matrix.

3. Apply both Kron and Neutral Return Current reduction methods to the 4x4 impedance matrix obtained in Step 2 to obtain 3x3 equivalent phase impedance matrices.

4. Transform the 3x3 phase impedance matrices obtained in Step 3 to sequence impedance matrices.

5. The matrix of Step 4 is divided by 2 to get the final result.

This appears to be a good engineering approximation because the two circuits are in parallel. However, the next section shows that this approximation may result in significant impedance errors, compared with the Exact method presented in Section 3.2.1.

<table>
<thead>
<tr>
<th>Table 3-1 Sequence Impedance Results with Kron Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>350 MCM AA</td>
</tr>
<tr>
<td>500 MCM AA</td>
</tr>
<tr>
<td>750 MCM AA</td>
</tr>
<tr>
<td>1000 MCM AA</td>
</tr>
</tbody>
</table>
Table 3-2 Sequence Impedance Results with Neutral Return Current Reduction

<table>
<thead>
<tr>
<th></th>
<th>Exact Method (Ω/1000ft)</th>
<th>Approximate Method (Ω/1000ft)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R₀</td>
<td>X₀</td>
<td>Rₜ</td>
</tr>
<tr>
<td>350 MCM AA</td>
<td>0.5943</td>
<td>0.1420</td>
<td>0.1752</td>
</tr>
<tr>
<td>500 MCM AA</td>
<td>0.4469</td>
<td>0.1075</td>
<td>0.1235</td>
</tr>
<tr>
<td>750 MCM AA</td>
<td>0.2956</td>
<td>0.0767</td>
<td>0.0865</td>
</tr>
<tr>
<td>1000 MCM AA</td>
<td>0.2352</td>
<td>0.0658</td>
<td>0.0696</td>
</tr>
</tbody>
</table>

Table 3-3 Conductor Configuration

<table>
<thead>
<tr>
<th></th>
<th>Conductor Type</th>
<th>GMR of Phase Conductor (inch)</th>
<th>Resistance of Phase Conductor (Ω/mile)</th>
<th>Number of Concentric Neutral Strands</th>
<th>Resistance of Concentric Neutral (Ω/mile)</th>
<th>GMR of Concentric Neutral (inch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>350 MCM AA</td>
<td>1</td>
<td>0.262</td>
<td>0.342</td>
<td>14</td>
<td>0.9821</td>
<td>0.590</td>
</tr>
<tr>
<td>500 MCM AA</td>
<td>2</td>
<td>0.312</td>
<td>0.242</td>
<td>12</td>
<td>0.7339</td>
<td>0.665</td>
</tr>
<tr>
<td>750 MCM AA</td>
<td>3</td>
<td>0.385</td>
<td>0.159</td>
<td>12</td>
<td>0.4646</td>
<td>0.770</td>
</tr>
<tr>
<td>1000 MCM AA</td>
<td>4</td>
<td>0.445</td>
<td>0.122</td>
<td>12</td>
<td>0.3696</td>
<td>0.845</td>
</tr>
<tr>
<td>4/0 CU</td>
<td>Independent Neutral</td>
<td>0.200</td>
<td>0.311</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

3.2.3 Comparing Approximate and Exact Methods

Considering the impedance of the conductor system shown in Figure 3.1, results from the Approximate method presented in Section 3.2.2 are to be compared with the results from the Exact method presented in Section 3.2.1. The notation of (3-16) and (3-17) will be used in the comparisons.

Tables 3-1 and 3-2 present the comparisons for four representative concentric neutral conductors with an independent neutral. Results where the Kron reduction approach is used are presented in Table 3-1. Table 3-2 shows results for the Neutral Return Current approach. Table 3-3 presents parameters for the conductors of Tables 3-1 and 3-2. The last conductor in Table 3-3 is the independent neutral return used for each of the calculations.
[45]. The value shown in the Conductor Type column of Table 3-3 is used to identify the conductors in Figures 3.4 - 3.7.

From Tables 3-1 and 3-2 it may be noted that there is essentially no error in the positive sequence quantities. This result could have been anticipated from the theory of balanced three-phase circuits.

The errors in the zero sequence quantities shown in Tables 3-1 and 3-2 are significant. The zero sequence reactance has the largest errors, ranging from approximately 15% for the largest conductor to almost 29% for the smallest conductor.

Figure 3.4 plots the percent error in $R_0$ and Figure 3.5 plots the percent error in $X_0$ for both the Kron and Neutral Return Current methods, where the conductor type is shown in Table 3-3. In both cases larger conductors result in smaller errors. But even for the largest conductor, 1000 MCM AA, the errors are significant. $R_0$ is always approximated low, whereas $X_0$ is always approximated high.

Figure 3.6 plots $R_0$ in ohms for the Exact method for both the Kron and Neutral Return Current reduction methods. Because the curves are so close together the plots appear to lie on top of one another in Figure 3.6. Likewise, Figure 3.7 plots $X_0$ in ohms for the Exact method for both the Kron and Neutral Return Current methods. This plot presents a range of values for $X_0$ which fall between the Kron assumptions and the Neutral Return Current assumptions.
Figure 3.4 Percentage Error of $R_0$ Approximate Calculation for Both Kron and Neutral Return Current (NRC) Reductions

Figure 3.5 Percentage Error of $X_0$ Approximate Calculation for Both Kron and Neutral Return Current (NRC) Reductions
Figure 3.6 Results for $R_0$ by Exact Method for Both Kron and Neutral Return Current (NRC) Reductions

Figure 3.7 Results for $X_0$ by Exact method for Both Kron and Neutral Return Current (NRC) Reductions
3.2.4 Parallel Conductors of Different Types

Different types of conductors may be placed in the same manhole. For instance, a set of three-phase 350 MCM CU conductors is placed in one duct, while a set of three-phase 500 MCM AA conductors is placed in a second duct. Also, a 4/0 Cu conductor as an independent neutral return is placed in a third duct. In this configuration, neglecting the mutual coupling causes considerable error. Calculation results for the impedances are shown in Table 3-4.

<table>
<thead>
<tr>
<th></th>
<th>Exact Method (Ω/1000ft)</th>
<th>Approximate Method (Ω/1000ft)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_0$ $X_0$ $R_+$ $X_+$</td>
<td>$R_0$ $X_0$ $R_+$ $X_+$</td>
</tr>
<tr>
<td>Kron</td>
<td>0.3920 0.1303 0.0718 0.0961</td>
<td>0.4389 0.1874 0.1092 0.1057</td>
</tr>
<tr>
<td>NRC</td>
<td>0.3954 0.1085 0.0718 0.0961</td>
<td>0.4512 0.1547 0.1092 0.1057</td>
</tr>
</tbody>
</table>

In the above table, the Exact Method is to calculate the full impedance matrix and then crunch it to 3x3 with Kron or NRC reduction. The Approximate method is to calculate the impedance for two conductors separately, and then calculate the overall impedance by considering them as parallel impedances.

Now let us consider voltage drops predicted by impedances of Table 3-4 with the Kron Exact and Approximate methods. The system to be considered is shown in Figure 3.8. The feeder is 3,000 feet long and a customer load of
1600kW+900kVAR for each phase is connected at the end point of the feeder. The custom level bus voltage at the source end is 120V.

![Diagram](image)

**Figure 3.8 Voltage Drop Study for Parallel Circuits with Different Cable Types**

The results of the power flow are shown in Table 3-5. The voltage at the end point of the feeder is 109.6V with the actual impedance, while it is 105.6V with the approximate impedance. That is, the voltage drops are 10.4V and 14.4V, respectively. The voltage drop error in this case is 38.6%.

<table>
<thead>
<tr>
<th></th>
<th>S (kVA)</th>
<th>I (A)</th>
<th>V (V)</th>
<th>ΔV (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>1600+j900</td>
<td>725</td>
<td>109.6</td>
<td>10.4</td>
</tr>
<tr>
<td>Approx.</td>
<td>1600+j900</td>
<td>753</td>
<td>105.6</td>
<td>14.4</td>
</tr>
</tbody>
</table>

The above results show that the Exact Method allows the feeder to carry heavier loads than that predicted by the Approximate Method. That is, the capacity of the system is larger with the Exact Method. Another aspect to be considered here is that the Approximate Method may predict a voltage that is out of limits, where the voltage may actually be within limits.
3.3 Object-oriented Design for Modeling Cables within *PowerFrame*

3.3.1 A Sub-package to Model Cables

Chapter 2 presented a conceptual and logical design for *PowerFrame*. It does not touch detailed designs for any layer. In this section a detailed object-oriented design for cables will be presented as part of the *Element Layer* in *PowerFrame*.

Cables are the most important elements in power distribution systems. Also, the number of cables overwhelms any other kind of element. Therefore, a good design for cables is critical for *PowerFrame* to work accurately and efficiently. In other words, this design should consider the Exact method of cable impedance discussed in Section 3.2, and model it in an efficient way.

A sub-package, *Cable*, is created for modeling cables. Figure 3.9 illustrates the role of the sub-package in the *Element Layer* in *PowerFrame*. The parent class, *AbstractCable*, inherited from *AbstractElement*, the upper-most class in this layer. Since *AbstractElement* implements the interfaces *IConnElem* and *ICktLaws*, *AbstractCable* and all subclasses implement these two interfaces.

Our design for the *Cable* sub-package is an object-oriented design in nature, since it is in the boundary of *PowerFrame*. This design employs the Composite design pattern to model multiple, parallel cables with the Exact method discussed in Section 3.2.
3.3.2 Composite Pattern to Model Cables with Exact Impedance Calculation

A) Identifying Common Attributes

From Section 3.2, we know that the configuration of cables can be any one of the following:

- A single-phase cable with only one path
- A single-phase cable with concentric neutral (It has two paths, a phase conductor and a neutral return).
- A three-phase cable with three paths
• A three-phase cable with concentric neutrals (It has 6 paths, 3 phase paths and 3 neutral returns).

• A composite cable consists of multiple sets of three-phase conductors placed in different ducts, with terminals tied together.

Although we have simple and complicated configurations of cables, each configuration has many similarities. From the client’s point of view, each cable provides two interfaces $IConnElem$ and $ICktLaws$, as stated in Chapter 2. Next we discuss the common attributes of cable models.

All cables share the following similarities that can be presented in the $AbstractCable$ class.

• Each cable impedance can be represented by a 3x3 impedance matrix. For a single-phase conductor, there is only one non-zero entry. For the composite cable, after reduction, as stated in Sections 3.2, its impedance matrix is a full 3x3 matrix.

• As a two-port element, each cable has currents, voltages, and power flows at two ports. They are represented with a 3x1 vector as $I$, $V1$, $V2$, $S1$ and $S2$, since the three-phase model is considered here. For single-phase cables, the above vector variables contain zero elements.

• All components keep track of directly connected components. Each has pointers pointing to its preceding (parent) element, left-most succeeding (child) element, and right sibling element, geographically. This is common to all elements, including any cable classes.
B) Designing Class Hierarchy

Since the three-phase model concerns the phase characteristics of conductors, we can group the types of cables in the following way.

- Single-phase cable
  - One-path, single-phase cable
  - Two-path, single-phase cable (with concentric neutral return)

- Three-phase cable
  - Three-paths, three phase cable
  - Six-paths, three phase cable (with concentric neutral returns)

- Composite cable
First, the two categories, single-phase and three-phase cables, are discussed. It is natural to place them into two-level subclasses of AbstractCable. The class hierarchy is described with the inheritance diagram shown in Figure 3.10. The behaviors (implementations of two interfaces) and attributes of

![Figure 3.10 Class Hierarchy of Cable Classes (1)](image-url)
AbstractCable are also shown in the figure. They are self-explanatory by their names.

The CompositeCable class to model composite cables is added into the diagram of Figure 3.10 as a subclass of AbstractCable. However, CompositeCable plays more roles than SinglePhaseCable and ThreePhaseCable, which serve only as child classes of AbstractCable.

A CompositeCable object contains some objects of SinglePhaseCable, ThreePhaseCable, or even an object of CompositeCable. As shown in Figure 3.11, the composite cable contains a three-phase cable, a single-phase cable, and another composite cable. This object relationship is shown in Figure 3.11.

![Figure 3.11 Object Diagram for a Composite Cable](image)

The above example indicates that the CompositeCable class may be associated with multiple subclasses of AbstractCable. This one-to-many association is shown in Figure 3.12. Certainly, the CompositeCable contains
an array of references, shown as $pCables$ in Figure 3.12, to keep track of all associated cable objects.

![Class Hierarchy of Cable Classes (2)](image)

**Figure 3.12 Class Hierarchy of Cable Classes (2)**

The pattern for this design is known as the Composite pattern. With the Composite pattern, a class is used to represent a container of many similar classes. References of contained classes are recorded. This makes it possible to keep an accurate model and reduce the primitive matrix to a common 3x3 impedance matrix.

Without the Composite pattern, we can manually calculate the reduced matrix and hardcode it. This approach works, but it loses track of what cables are contained in a composite cable. Particularly, when two or more different types of cables are contained in a composite cable, it may be
desired to know the information of the contained cables. With the Composite pattern, every contained cable in a composite cable can be traced through the references, $pCables$.

Without the Composite pattern, we also have to keep additional information or intermediate classes to keep track of contained cables in a composite cable. This approach works too, but this increases the complexity of the software and even destroys the hierarchy of cable classes, since additional information or even intermediate classes are involved. With the composite pattern, only references are recorded. So, the composite pattern is an easy and neat way to represent this part-whole relation. Since a unified interface is followed by “part” and “whole”, the client can still treat individual and composite objects uniformly within the Composite pattern.

### 3.4 Summary

*PowerFrame* is designed to be able to model cables, particularly composite cables, accurately and efficiently. The mathematical model shows an Exact method to model impedances for composite cables. The composite pattern shows a software design technique to model composite cables efficiently.

The mathematical analysis of cable impedances shows that the Approximate method would result in significant errors in unbalanced power flow studies. It is also shown that a combination of different types of conductors may cause errors in the positive sequence impedance with the Approximate
method. Hence, errors may be produced even in balanced power flow studies.

The detailed design of *PowerFrame* employs the Composite design pattern to model composite cables. The Composite pattern provides an efficient way to represent part-whole relation within hierarchy architecture.
Chapter 4
Extending PowerFrame to Model Distributed Resources

This chapter discusses mathematical models of Distributed Resources first. Unbalanced circuits are considered here. Then Distributed Resources are used as a test bed for the extensibility of PowerFrame. Interfaces of Element Layer in PowerFrame are demonstrated to be neutral to particular problems.

Section 4.1 introduces Distributed Resources. Section 4.2 discusses two different models of Distributed Resources. Sections 4.3 demonstrates an object-oriented design of extending PowerFrame to model Distributed Resources.

4.1 Distributed Resources

The use of Distributed Resource (DR) electrical generation in distribution circuits is growing. DR generation comes in many forms including gas turbine driven synchronous generators, wind powered induction generators, fuel cells with inverter circuitry, and others. The economic and technical advantages of DRs are being considered in residential, commercial, and industrial market places [22-23].
During normal conditions distribution circuits operate in unbalanced states, with at a given location secondary voltage magnitudes deviating between phases by as much as 8 volts or more on a 120 volt basis. This chapter investigates power flow analysis of DRs in association with distribution circuit imbalances, where imbalances arise from multi-phase circuit construction, unbalanced loads, and non-symmetric circuit impedances and admittances. For accurate analysis of distribution systems, the imbalances must be taken into account. This implies the use of a multiphase model for analysis that simulates each current path and each phase load.

Two circuit models of DRs are presented in the next section. The first model presented in Section 4.2.1 considers the DR operating as a three-phase voltage source. This is not the most accurate model of a DR, but it is considered in order to investigate and illustrate unbalanced effects. The second DR model presented in Section 4.2.2 is derived just for synchronous generators, and it models the DR as a voltage-dependent current source.

4.2 Mathematical Models of Distributed Resources

4.2.1 Constant Voltage Source

In electrical networks and power system analysis, common models for sources, such as synchronous generators, are voltage or power sources. The general literature uses single-phase equivalent models. Previous system studies involving DRs have made use of single-phase equivalent models [22-23]. Due to unbalanced conditions in distribution systems, more accurate analysis may be achieved by using multi-phase models.
In this section a generic model of a DR as a three-phase balanced, constant voltage source is considered. The assumption for this model is that for small deviations the DR is strong enough to control the voltage to any value and supply the resulting power flows. Observations of unbalanced effects in distribution circuits are made with this model.

In Figure 4.1 the substation has a single distribution feeder and the DR is placed at the very end of the feeder. The distribution feeder is modeled with three switches labeled as SW1 – SW3, and four line sections. Each line section is a multi-grounded, three-phase section with 336ACSR phase conductors and a 4/0 ACSR neutral return. A horizontal line spacing with A-B spacing of 0.76 meter (2.5 feet) and A-C spacing of 2.23 meters (7.3 feet), with the neutral 0.91 meter (3 feet) below the B phase, is used for the construction. The line model used here is representative of distribution systems. The length of each line section is 3048 meters (10000 feet), making the overall length of the distribution feeder approximately 12.228 km (7.6 miles). The phase impedance matrix, where impedances are specified in ohms, for each line section is given by

\[
\begin{bmatrix}
0.74 + j1.98 & 0.27 + j0.89 & 0.26 + j0.67 \\
0.27 + j0.89 & 0.76 + j1.90 & 0.27 + j0.72 \\
0.26 + j0.67 & 0.27 + j0.72 & 0.74 + j1.98
\end{bmatrix}
\]  

\( (4-1) \)
Each line section supplies a three-phase load of 100 kW + j100 kVar or the overall feeder is supplying a total three-phase load of approximately 1.7 MVA. The only unbalance in the distribution circuit is due to the line impedance.

First assume that the voltage magnitude and angle of the DR are equal, for each corresponding phase, to those of the substation. In this case balanced voltages and currents exist throughout the feeder, with the substation supplying the loads on line sections between SW1 and SW2 and the DR supplying the loads on line sections between SW2 and SW3.

Next assume that the voltage magnitude of each phase of the DR deviates from the substation nominal by the same amount. In this case the DR is still operating as a balanced source, but its values are slightly different from those of the substation. Also in this case unbalanced voltages, currents, and power flows exist throughout the feeder.

To calculate the resulting voltages and currents, a distribution power flow program was used [36-37]. Tables 4-1 and 4-2 present results of power flow runs and show power flows at the DR as a function of deviations in DR voltage magnitudes and angles, where for magnitude deviations a 1% difference indicates that the DRs voltage magnitude is running 1% above that of the substation.

In Tables 4-1 and 4-2 imbalance is calculated from

\[ Imbalance = \frac{\max\{P_i-\text{average}(P_i)\}}{\text{average}(P_i)} \quad i = a, b \text{ and } c \quad (4-2) \]
Table 4-1 DR Real Power Flows as a Percentage of the Total Load for Phases A, B, and C and Real Power Flow Imbalance Versus Percentage Change in DR Source Voltage Magnitude for Circuit of Figure 4.1

<table>
<thead>
<tr>
<th>ΔV (% of nominal)</th>
<th>P_a (% of total load)</th>
<th>P_b (% of total load)</th>
<th>P_c (% of total load)</th>
<th>Average (% of total load)</th>
<th>Imbalance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>49.8</td>
<td>49.9</td>
<td>49.8</td>
<td>49.8</td>
<td>0.1</td>
</tr>
<tr>
<td>1.0</td>
<td>63.4</td>
<td>60.8</td>
<td>57.9</td>
<td>60.7</td>
<td>4.6</td>
</tr>
<tr>
<td>2.0</td>
<td>77.3</td>
<td>71.9</td>
<td>66.2</td>
<td>71.8</td>
<td>7.8</td>
</tr>
<tr>
<td>3.0</td>
<td>91.5</td>
<td>83.2</td>
<td>74.5</td>
<td>83.1</td>
<td>10.3</td>
</tr>
<tr>
<td>4.0</td>
<td>105.9</td>
<td>94.7</td>
<td>83.1</td>
<td>94.6</td>
<td>12.1</td>
</tr>
<tr>
<td>5.0</td>
<td>120.6</td>
<td>106.5</td>
<td>91.8</td>
<td>106.3</td>
<td>13.6</td>
</tr>
</tbody>
</table>

Table 4-2 DR Real Power Flows as a Percentage of the Total Load for Phases A, B, and C and Real Power Flow Imbalance Versus Change in DR Source Voltage Angle for Circuit of Figure 4.1

<table>
<thead>
<tr>
<th>Δθ (degrees)</th>
<th>P_a (% of total load)</th>
<th>P_b (% of total load)</th>
<th>P_c (% of total load)</th>
<th>Average (% of total load)</th>
<th>Imbalance (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>49.8</td>
<td>49.9</td>
<td>49.8</td>
<td>49.8</td>
<td>0.1</td>
</tr>
<tr>
<td>0.5</td>
<td>70.4</td>
<td>75.1</td>
<td>71.2</td>
<td>72.2</td>
<td>4.0</td>
</tr>
<tr>
<td>1.0</td>
<td>78.5</td>
<td>100.4</td>
<td>92.6</td>
<td>90.5</td>
<td>13.2</td>
</tr>
<tr>
<td>1.5</td>
<td>111.8</td>
<td>125.8</td>
<td>114.0</td>
<td>117.2</td>
<td>7.4</td>
</tr>
<tr>
<td>2.0</td>
<td>132.6</td>
<td>151.3</td>
<td>135.5</td>
<td>139.8</td>
<td>8.2</td>
</tr>
</tbody>
</table>

From Tables 4-1 and 4-2 it may be noted that as the DRs voltage magnitude or angle is increased, the imbalance in the feeder phase power flow increases. This imbalance is due to the unbalanced impedance and admittance of the distribution feeder, where the unbalanced phase impedance matrix is presented in Equation (4-1) above. It may be noted that if the DR’s voltage magnitude is increased by 4% above that of the substation’s voltage magnitude, then the DR supplies all of the feeder load on phase A, 95% of the feeder load on phase B, and 83% of the feeder load on phase C. A point of interest is that the operation of a balanced source
(with balanced loads) is causing significant unbalanced power flows in the feeder due to the unbalanced impedance.

Image: Figure 4.2 DR Terminal Voltage at Node k and Complex Power Flow in Line ij for General Power System

An analytic expression will now be derived that relates a change in power flow in a line section to a change in a DRs terminal voltage. This expression shows the relation between unbalanced impedances and unbalanced power flows, and may be used to verify the power flow results observed.

The objective of our analysis is to find the relation between a change in voltage at the DR terminals and a change in a line flow in the system. For example, consider Figure 4.2 where it is desired to determine the change in the complex power flows in line (i,j) for all three phases when there is a change in the terminal voltage of the DR at node k.

Letting $\bar{S}$ represent a complex power flow, $\bar{V}$ a phasor voltage, $\bar{I}$ a phasor current, and asterisk(*) the conjugate of a complex number, consider the following per phase equations in relation to the $ij$ power line shown in Figure 4.2.
Rewriting in vector-matrix form gives

\[
\mathbf{s}_{ij}^* = \mathbf{V}_i^* \cdot I_{ij}
\]  

(4-4)

Assuming that the voltage at node \(i\) remains constant for a small change in the terminal voltage of the DR gives

\[
\Delta \mathbf{s}_{ij}^* = \mathbf{V}_i^* \cdot \Delta I_{ij} = \left(\mathbf{V}_i^* \cdot [z_{ij}]^{-1} \cdot \Delta \mathbf{v}_i - \mathbf{v}_j\right)
\]  

(4-5)

where \([z_{ij}]\) is the 3x3 impedance matrix of line \(ij\), such as Equation (4-1).

Now let \([Z_{ik}],[Z_{jk}]\) and \([Z_{kk}]\) be system bus impedance matrices, such that

\[
\mathbf{v}_i - \mathbf{v}_j = \left[Z_{ik}\right] \cdot \mathbf{I}_k - \left[Z_{jk}\right] \cdot \mathbf{I}_k = \left[Z_{ik}\right] \cdot \left[Z_{jk}\right]^{-1} \cdot \mathbf{v}_k
\]  

(4-6)

Combining Equations (4-5) and (4-6), we have

\[
\Delta \mathbf{s}_{ij}^* = \left(\mathbf{r}_i^* \cdot [z_{ij}]^{-1} \cdot \left[Z_{ik}\right] \cdot \left[Z_{jk}\right]^{-1} \right) \cdot \Delta \mathbf{v}_k
\]  

(4-7)

Equation (4-7) may be used to verify the power flow results shown in Tables 4-1 and 4-2.

Modeling the DR as a balanced voltage source does not take into account the interaction between the DR and the system. Realistically, in order to have an effect on its terminal voltage, the DR has to interact with the power system. This is considered in the next section.
4.2.2 Voltage-Dependent Current Source

A) Synchronous Generator Model for Unbalanced Circuit Calculations

The model developed here for the synchronous generator will no longer assume that the terminal voltage magnitudes and angles can be controlled to any value. Here the synchronous generator can affect the terminal voltage values only through an interaction with the distribution power system itself. The solution of the distribution system determines the machine’s terminal voltages, and the currents injected by the machine will be a function of the terminal voltages. Considered in this form, the DR is looked upon as a voltage-dependent current source.

Since the terminal voltages may be unbalanced, the injected currents may also be unbalanced. If the imbalance in the injected currents becomes too large, the machine may be shut down by its protection system. Only small imbalances in injected currents are considered here. Note that the solution of the distribution system sets the machine’s terminal voltages, but the machine in turn can affect these terminal voltages by varying the injected currents.

Both single-phase and three-phase synchronous generators may serve as distributed resources. The model developed here is for a three-phase synchronous generator.

Equations (4-8)-(4-10) shown below, based on Faraday’s Law, is the starting point for the model development [46-47].
where

- $v_k$ = voltage of phase $k$
- $\lambda_k$ = flux linkages for phase $k$
- $k = a, b, c.$

Assuming a round rotor synchronous generator, the flux linkages may be calculated from

\[ \lambda_a = L_S \cdot \left( 2i_a - i_b - i_c \right) + L_{SR} \cdot I_R \cdot \cos \left( \omega_s t \right) \]  
\[ \lambda_b = L_S \cdot \left( -i_a + 2i_b - i_c \right) + L_{SR} \cdot I_R \cdot \cos \left( \omega_s t - 120 \right) \]  
\[ \lambda_c = L_S \cdot \left( -i_a - i_b + 2i_c \right) + L_{SR} \cdot I_R \cdot \cos \left( \omega_s t + 120 \right) \]  

where

- $i_k$ = stator current for phase $k$
- $I_R$ = rotor current,
- $L_S$ = self-inductance of stator
- $L_{SR}$ = mutual-inductance between stator and rotor.

Assuming that the phase currents may have different magnitudes and phase angles we may write

\[ i_a = I_A \cos \left( \omega_s t + \phi_A \right) \]  
\[ i_b = I_B \cos \left( \omega_s t + \phi_B \right) \]  
\[ i_c = I_C \cos \left( \omega_s t + \phi_C \right) \]
Substituting Equations (4-14)-(4-16) into Equation (4-11) results in

$$\lambda_a = 2L_S I_A \cos(\omega_s t + \phi_A) - L_S I_B \cos(\omega_s t + \phi_B) - L_S I_C \cos(\omega_s t + \phi_C) + L_{SR} I_R \cos(\omega_s t)$$  \hspace{1cm} (4-17)

Substituting Equation (4-17) into Equation (4-8) gives

$$v_a = 2\omega_s L_S I_A \sin(\omega_s t + \phi_A) - \omega_s L_S I_B \sin(\omega_s t + \phi_B) - \omega_s L_S I_C \sin(\omega_s t + \phi_C) + \omega_s L_{SR} I_R \sin(\omega_s t)$$  \hspace{1cm} (4-18)

Applying a phasor transformation to Equation (4-18) gives

$$V_A e^{j\theta_A} = \frac{2\omega_s L_S I_A}{\sqrt{2}} e^{-j\phi_A} - \frac{\omega_s L_S I_B}{\sqrt{2}} e^{-j\phi_B} - \frac{\omega_s L_S I_C}{\sqrt{2}} e^{-j\phi_C} + \frac{\omega_s L_{SR} I_R}{\sqrt{2}} e^{-j0}$$  \hspace{1cm} (4-19)

where

- $V_A$ = voltage magnitude of phase A
- $\theta_A$ = voltage angle of phase A
- $I_k$ = current magnitude for phase k
- $\phi_k$ = phase current angle of phase k.

Defining

$$X_s = \frac{\omega_s L_S}{\sqrt{2}}$$  \hspace{1cm} (4-20)

$$V_f = \frac{\omega_s L_{SR} I_R}{\sqrt{2}}$$  \hspace{1cm} (4-21)

Substituting Equations (4-20) – (4-21) into Equation (4-19) gives

$$V_A e^{j\theta_A} = 2X_s I_A e^{j\phi_A} - X_s I_B e^{j\phi_B} - X_s I_C e^{j\phi_C} + V_f e^{-j0}$$  \hspace{1cm} (4-22)

Performing a similar derivation for phases B and C as was done for phase A gives
Equations (4-22) – (4-24) represent a phasor model of a three-phase synchronous generator, where the terminal voltage and current magnitudes are not necessarily equal.

Equations (4-22) – (4-24) are the basis for the DR simulation to be performed in the Section 4.2.3. The terminal voltages of the DR are determined from a multi-phase power flow solution. The phase currents injected by the DR is a function of the terminal voltages and may be calculated from Equations (4-22) – (4-24).

**B) Unbalanced Circuit Simulation**

A power flow for a radial system is run for this simulation. For each iteration of the power flow solution, the currents injected by the DR may be calculated given the terminal voltages and the machine field voltage. Considered in this form, the DR is looked upon as a voltage-dependent, current source. The terminal voltages of the DR vary with each iteration of the power flow solution, and the currents injected by the DR depend upon the terminal voltages. The currents injected by the DR will affect the results from the next iteration of the power flow solution. This is continued until convergence occurs.

Figure 4.3 shows the circuit to be used for the simulation studies. The circuit consists of a substation, five cable sections, and a distributed resource.
that is connected to the circuit through the switch labeled “SW1.” Thus, the DR injects power into the circuit at Node 2.

![Diagram of circuit](image)

**Figure 4.3 Distributed Resource Modeled as Voltage-Dependent Current Source**

Each cable section shown in Figure 4.3 consists of a three-phase cable, and the length of all cable sections is the same. The impedance matrix for each cable section is given by

\[
\begin{bmatrix}
0.20 + j0.50 & 0.10 + j0.10 & 0.10 + j0.10 \\
0.10 + j0.10 & 0.20 + j0.50 & 0.10 + j0.10 \\
0.10 + j0.10 & 0.10 + j0.10 & 0.20 + j0.50
\end{bmatrix}
\]  

(4–25)

The loads for each cable section shown in Figure 4.3 are given in Table 4-3. Note that there is a significant imbalance in the loading on phases A, B, and C. This loading imbalance will also cause a voltage imbalance to exist among the phases at each node. Thus, the DR located at node 2 works against a set of unbalanced phase voltages.

<table>
<thead>
<tr>
<th>Cable</th>
<th>Phase A (kW + j kVAR)</th>
<th>Phase B (kW + j kVAR)</th>
<th>Phase C (kW + j kVAR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,1)</td>
<td>1250 + j1250</td>
<td>625 + j625</td>
<td>312.5 + j312.5</td>
</tr>
<tr>
<td>(1,2)</td>
<td>0 + j0</td>
<td>0 + j0</td>
<td>0 + j0</td>
</tr>
<tr>
<td>(2,3)</td>
<td>1250 + j1250</td>
<td>625 + j625</td>
<td>312.5 + j312.5</td>
</tr>
<tr>
<td>(3,4)</td>
<td>1250 + j1250</td>
<td>625 + j625</td>
<td>312.5 + j312.5</td>
</tr>
<tr>
<td>(1,5)</td>
<td>1250 + j1250</td>
<td>625 + j625</td>
<td>312.5 + j312.5</td>
</tr>
</tbody>
</table>
Two simulation studies are to be considered. In the first study the switch SW1, shown in Figure 4.3, is open and thus the DR is switched out of the circuit. In the second study the switch SW1 is closed and the DR supplies power to the circuit. The size of the DR was set at 4.5 MW, and thus represents approximately 50% of the existing circuit power requirements.

The results of the simulation runs are presented in Table 4-4. The first column of the table presents the node number, corresponding to the nodes shown in Figure 4.3, for which the row results apply. The next three columns present the customer level voltage (in volts) for each of the phases A, B, and C, where the DR is not switched into the circuit. Following that, the next three columns present customer level voltage for each of the phases A, B, and C, where the DR is switched into the circuit. The final three columns show the change in the voltage magnitude for each phase that occurs as a result of switching the DR into the circuit.

**Table 4-4 Customer Level Phase Voltages for Circuit Shown in Figure 4.3 With and Without DR Operating**

<table>
<thead>
<tr>
<th>Node</th>
<th>SW1 Open</th>
<th>SW1 Closed</th>
<th>Change of V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>A  B  C</td>
</tr>
<tr>
<td>0</td>
<td>127.02</td>
<td>127.02</td>
<td>0.00 0.00</td>
</tr>
<tr>
<td>1</td>
<td>120.13</td>
<td>124.46</td>
<td>126.77</td>
</tr>
<tr>
<td>2</td>
<td>116.70</td>
<td>123.19</td>
<td>126.66</td>
</tr>
<tr>
<td>3</td>
<td>113.29</td>
<td>121.94</td>
<td>126.54</td>
</tr>
<tr>
<td>4</td>
<td>111.58</td>
<td>121.32</td>
<td>126.48</td>
</tr>
</tbody>
</table>

Assume that the nominal customer level voltage is 120 volts, and that the voltage limits at loads are set at 110 volts on the low side and 128 volts on the high side. In light of these voltage constraints, consider the simulation results for the case where SW1 is open. In this case no voltage constraints are violated.
After the DR is switched into the circuit and supplies power, the phase C voltage at nodes 1-4 goes out of limits. Hence, switching the DR into the circuit caused voltage limits to be violated, where no violations existed previously.

One way the voltage violations could be caused in practice is via a voltage magnitude control that has a feedback measurement existing on only a single-phase. This type of control, where only a single-phase measurement is made, is common in distribution systems. For instance, in the simulation study considered here, assume that it is desired that the DR control the phase A voltage to 120 volts. Such a control would result in the phase C voltage at nodes 1-4 going out of limits.

It should be noted that the DR may be used to raise the voltage magnitudes along the line. This is due to the changes in the line flows upstream (i.e., toward the substation) from the DR. Note that downstream of the DR the voltages are pulled up uniformly. An analytical formula for evaluating such a voltage control is derived in the next section. The formula may be used to check the simulation results presented here.

C) Analytical Formulas for Voltage Changes after Connecting DRs

Consider Figure 4.4, where voltages and power flows are shown for a power line with a DR installed at node 2.
For the present, assume that Figure 4.4 represents a single-phase line. Given node 1 as reference, i.e., \( V_1 = V e^{j0} \), the node 2 voltage of Figure 4.4 may be expressed as

\[
V_2 = V - \frac{1}{P+Q} \left[ P \cdot R + Q \cdot X \right] + j \frac{1}{P+Q} \left[ Q \cdot R - P \cdot X \right] = (V + \Delta V) + j(\delta V) \quad (4-26)
\]

where \( R + jX = \) line impedance.

Figure 4.5 shows the phasor relationships between the voltages of Equation (4-26). Note that the change of \( V_2 \) is primarily due to \( \Delta V \) and not \( \delta V \). Thus, from Equation (4-26), \( V_2 \), the magnitude of voltage at node 2, is given by

\[
V_2 = V + \Delta V = V - \frac{1}{V} \left[ P \cdot R + Q \cdot X \right] = \text{Re}(V_2) \quad (4-27)
\]

From Equation (4-27), it may be noted that the magnitude of the ending node voltage primarily depends upon \( \Delta V \), the real part of the voltage drop. The angle of the ending node voltage primarily depends upon \( \delta V \), the imaginary part of the voltage drop (see Figure 4.5).
Assuming $V_1$ stays constant and that there is a decrease in the power flows in the line, $\Delta P$ and $\Delta Q$, then the voltage magnitude at node 2 is given by

$$
V_2' = V - \frac{1}{V} \left[ (P-\Delta P)R + (Q-\Delta Q)X \right]
$$

(4-28)

Subtracting Equation (4-27) from Equation (4-28)

$$
dV_2 = V_2' - V_2 = \text{Re} \left[ r_2' \right] - \text{Re} \left[ r_2 \right] = \text{Re} \left[ r_2' - r_2 \right] = \frac{1}{V} \left[ \Delta P R + \Delta Q X \right]
$$

(4-29)

If there is a power injection at node 2, for instance from a DR, then the P and Q line flows will decrease in proportion to the power injection. Hence, $(V_2' - V_2)$ will be positive and the voltage magnitude at node 2 will increase.

Now assume that the power line in Figure 4.4 is a three-phase line with symmetric mutual coupling, where the impedance matrix for the power line may be written as

$$
Z = \begin{bmatrix}
Z_s & Z_m & Z_m \\
Z_m & Z_s & Z_m \\
Z_m & Z_m & Z_s
\end{bmatrix}
$$

(4-30)

Also assume that $V_1$ for phases A, B and C has the same magnitude and has angles of 0, -120 and 120 degrees, respectively. We also assume that $V_1$ stays constant, such as the voltage at the substation. Let $V_1$ be given as

$$
V_1 = \begin{bmatrix}
r_{_a} \\
r_{_b} \\
r_{_c}
\end{bmatrix}
$$

(4-31)

Then, the voltage at node 2 is given as
We first analyze the voltage at node 2 for phase A. The voltages for phases B and C are similar except for phase angle shifts of –120 and 120 degrees, respectively. From Equation (4-31), the voltage at node 2 for phase A is given by

\[
\begin{bmatrix}
V_2^a \\
V_2^b \\
V_2^c 
\end{bmatrix} = \begin{bmatrix}
Z_s & Z_m & Z_m \\
Z_m & Z_s & Z_m \\
Z_m & Z_m & Z_s 
\end{bmatrix} \begin{bmatrix}
P_a-jQ_a \\
P_b-jQ_b \\
P_c-jQ_c 
\end{bmatrix}
\]

(4-32)

Now assume the DR injects \(\Delta P + j\Delta Q\) for each phase. This decreases the line flow by \(\Delta P + j\Delta Q\) for each phase, where we neglect the effects of power loss. This can be expressed as

\[
P_k' + jQ_k' = (P_k - \Delta P) + j(Q_k - \Delta Q)
\]

(4-34)

where \(P_k' + jQ_k'\) = line flow of phase \(k\)

Therefore, the voltage change at node 2 for phase A is given by

\[
\begin{align*}
\Delta V_2^a &= V_2^a - V_2^a \\
&= \frac{1}{V} \left[ Z_s (\Delta P - j\Delta Q) + Z_m (\Delta P - j\Delta Q) e^{-j120} + Z_m (\Delta P - j\Delta Q) e^{j120} \right] \\
&= \frac{1}{V} \left[ (Z_s + Z_m e^{-j120} + Z_m e^{j120}) \Delta P - j\Delta Q \right] \\
&= \frac{1}{V} \left[ (Z_s - Z_m) \Delta P - j\Delta Q \right] \\
&= \frac{1}{V} \left[ (R_s - R_m) \Delta P + (X_s - X_m) \Delta Q \right] + j \frac{1}{V} \left[ (X_s - X_m) \Delta P - (R_s - R_m) \Delta Q \right] 
\]

(4-35)
The change of voltage magnitude at node 2 of phase A is the real part of \((\bar{V}_{2_{-a}}' - \bar{V}_{2_{-a}})\). Thus, the change of voltage magnitude is given by

\[
dv_{2_{-a}} = v_{2_{-a}}' - v_{2_{-a}} = \text{Re} \left( \bar{V}_{2_{-a}}' - \text{Re} \left( \bar{V}_{2_{-a}} \right) \right) = \text{Re} \left( \bar{V}_{2_{-a}}' - \bar{V}_{2_{-a}} \right) = \frac{1}{\mathcal{P}} \Delta P \left( \mathcal{R}_s - \mathcal{R}_m \right) + \Delta Q \left( \mathcal{X}_s - \mathcal{X}_m \right)
\]

Equations (4-36) and (4-37) give the same results for each phase. This is quite reasonable because everything for phases B and C are identical to phase A except that there is a –120 or 120 degree phase angle shift. In the above analysis, we assumed the voltage at node 1 shown in Figure 4.4 stays constant.

Now consider Figure 4.6, with a DR connected at node 3. The voltages at nodes 1 and 2, upstream of node 3, will change due to a change in the DRs power injection. Then, the overall voltage change at node 3 is the accumulation of voltage changes at nodes 1, 2, and 3.

![Figure 4.6 Accumulation of Voltage Change in DR’s Upstream Lines](image)

For a general node n where a DR injects \(\Delta P + j \Delta Q\), assume that the power flow in all upstream lines of node n is decreased by \(\Delta P + j \Delta Q\) (where changes
in power loss are neglected). Then, the expected change in voltage magnitude at node $n$ is given by

$$dV_{n,a} = dV_{n,b} = dV_{n,c} = \frac{\Delta P}{V} \cdot \sum_{j \in U} (R_{s,j} - R_{m,j}) + \frac{\Delta Q}{V} \cdot \sum_{j \in U} (\chi_{s,j} - \chi_{m,j}) \quad (4-38)$$

where

$V$ = the rated voltage level or voltage base

$U$ = the set of all lines upstream of node $n$.

If we consider all lines from node 0 (the substation) to node $n$ (the DR node) as a “long” power line, Equation (4-38) is exactly Equation (4-37).

By substituting the appropriate parameter and variable values into equation 38, the simulation results obtained earlier in this chapter may be verified. Also, Equation 38 may be used by a system planner to place a DR in a distribution circuit for voltage magnitude control.

### 4.2.3 Impacts on Imbalance from Two Models

This section has addressed the development of DR models for use in unbalanced power flow analysis of electrical distribution circuits.

The results from Section 4.2.1 show that a DR, modeled as a balanced voltage source, in a distribution circuit with perfect load balance can result in unbalanced power flows due to the unbalanced line impedances. The voltage source approximation of the DR is only valid for small variations, but illustrates power flow interactions that would occur between the substation, viewed as a voltage source, and a DR whose terminal voltages deviate from that of the substation. An analytical formula for a three-phase
system was developed that relates changes in real and reactive power flows to changes in DR source voltage magnitudes and angles.

A three-phase model for a DR of type synchronous generator was developed. The model was then used in a simulation study. The simulation showed that in a distribution circuit operating in an unbalanced but legal state, a DR being switched on can cause circuit constraints to be violated at some phases. Hence, the control of DRs under unbalanced circuit conditions could be important.

### 4.3 Extending PowerFrame to Model Distributed Resources

#### 4.3.1 DistRes: An Extended Module of PowerFrame

*PowerFrame* is designed to be extensible. That means *PowerFrame* could solve current and future problems in the domain of power system analysis. *PowerFrame* obtains extensibility through interfaces provided by each layer. Since interfaces of classes are defined as the responsibilities known publicly, new modules can be attached to *PowerFrame* seamlessly as long as they follow the specified interfaces. This extension does not require any explicit knowledge of the framework, since all it needs to know is the interfaces. Also, the extension does not require any recompilation within the existing framework.
DRs here serves as a test bed for extensibility of PowerFrame. Figure 4.7 depicts the extended package named DistRes to model DRs. AbstractDR is extended from AbstractElement that implements two interfaces, IConnElem and ICktLaws, as indicated in Chapter 2. ConstVoltDR and VoltDepCurrDR are two classes inherited from AbstractDR to simulate the two different DR models, constant voltage source and voltage-dependent current source.

The extended package of DistRes can be viewed as a unified part of the Element layer, since all classes in DistRes inherit AbstractElement and implement interfaces IConnElem and ICktLaws. The extended boundary of Element layer is shown as the dashed line in Figure 4.7.

In Section 4.3.2, it is shown that the ICktLaws interface is sufficient to extend PowerFrame to model Distributed Resources. In Section 4.3.3, a
more general analysis is given to show that ICktLaws is sufficient for modeling any new elements in power systems.

4.3.2 Interactions with PowerFrame

Different Distributed Resource models perform differently in the real world. In the computer simulation calculation, different software models of DRs interact with the rest of PowerFrame modules differently. This section discusses the interaction of the DR modules with other modules in PowerFrame for a typical power system calculation, power flow.

A) Modeled as Constant Voltage Source by ConstVoltDR

Since DRs are modeled as constant voltage sources, the power flow calculation considers a DR as a root element like a substation in the system. For this model, the presence of a DR may create an additional loop in the system. Since the system is always modeled as a tree for a radial system or a collection of trees for a looped system, the additional loop created by the DR must be virtually eliminated. That is, a new co-tree component must be identified. Hence, the system can be represented as a collection of radial trees. To accomplish this, the following two approaches can be applied.

If there is a normally-closed switch, it can be set as a co-tree component.

If there is no switch in the loop, a virtual switch can be inserted into the system as a co-tree element.
Hence, the system with a \textit{ConstVoltDR} has one extra root element, one extra (virtual) co-tree switch, and one more tree in the collection of trees. The system can be consistently represented by a collection of trees and co-tree elements. There is no change for all existing algorithm classes which interact with subclasses of \textit{AbstractElem}, including \textit{ConstVoltDR}.

\textbf{B) Modeled as Voltage Dependent Current Source --\textit{VoltDepCurrDR}}

When a DR is modeled as Voltage-dependent current source, the DR is not considered as a root element. They are treated as an intermediate, regular element in the tree-based approach.

For a backward traverse, the difference between the desired DR voltage and the voltage from the forward traverse is determined first. Then, the current is calculated using the voltages from Equations (4-22) to (4-24). For a forward traverse, voltage drops are calculated as usual.

The voltage magnitude of the DR will eventually meet the desired limits by this approach. This is checked after a backward and a forward traverse. Also, the convergence of power flow needs to be checked. If not, the repetition process of backward and forward traverse must be further executed.

The UML diagram in Figure 4.8 illustrates the above process of the power flow calculation when there is \textit{VoltDepCurrDR} in the system.
In Figure 4.8, the messages are self-explanatory. The objects and classes are described as follows.

* \textit{pf} is an object of \textit{PowerFlow} class;
* \textit{bi} is an object of \textit{BackwardIterator} class;
* \textit{fi} is an object of \textit{ForwardIterator} class;
* \textit{dr} is an object of \textit{VoltDepCurrDR} class;
* \textit{e} is an object of any subclasses of \textit{AbstractElem} class.
4.3.3 Why Interfaces are Sufficient for Extensions

The interface $IConnElem$ is sufficient for traversing a tree-based system. The reason is that it provides the methods to find the three connected elements, which are sufficient connection knowledge for the whole system [34].

The $ICktLaws$ is sufficient for power flow analysis in the aspect of circuits with such assumptions.

- Elements must have two ports. (For most cases, this is true. If not, we can transform it to three 2-port elements.)
- Traversing-based approach rules. Then, traversing can be handled through providing methods at two ports, such as $I()$, $V1()$ and $V2()$, $S1()$ and $S2()$.
- Circuit Laws hold for two-port elements.

Since every element has two internal ports, every element has “across” features, which describe the information about the internal ports. Also, since every element contains a lumped physical model (impedances, transformer windings, or switches), it has “through” features. Physically, from the point of view of electric circuit laws, the “across” features are the voltages at internal ports, and the through features are the currents and powers through an element. They are all modeled as methods in the $ICktLaws$ interface, which is supported by the $Element$ layer.

In the $ICktLaws$ interface, methods of $I()$, $S1()$ and $S2()$ are associated with “through” features, and methods of $V1()$ and $V2()$ are associated with “across” features. Hence, interface $ICktLaws$ provides sufficient methods to
model the responsibilities of elements from the point of view of electrical circuit laws.

4.4 Summary

The models of DRs demonstrate the impact on imbalance when operating in an unbalanced power distribution system. When modeled as a constant voltage source, the DR could worsen the imbalance of the system. When modeled as a voltage-dependent current source, the DR could raise the upstream voltages of some phases to violate the constraints, while keeping voltages of other phases within voltage constraints.

The DR model also serves as a conceptual test for the extensibility of PowerFrame. Following the same interface of the Element layer, a module can be attached to the Element layer to model different DRs. These interfaces are sufficient to model DRs, even new elements, in the scope of tree-based circuit analysis.
Chapter 5
Distributed Algorithms for Radial and Looped Power Flows

As illustrated by Figure 2.16 in Chapter 2, the fourth layer in PowerFrame, DistributedAlgorithm layer, provides an abstraction of distributed algorithms for power system analysis. Even though many analyses can be designed and implemented in parallel or distributed, this chapter focuses on distributed algorithms for the power flow calculation, which is a typical, fundamental calculation for power system analysis.

First, a general distributed algorithm for a tree-based system is presented in this chapter. Secondly, a distributed algorithm for Radial Power Flow is presented along with experimental investigations of execution speed. Thirdly, a distributed algorithm for Looped Power Flow is presented again with experimental investigations of execution time.
5.1 A General Approach to Distributed Algorithms for Tree-based System

5.1.1 Controller and Workers

Distributed computation usually involves multiprocessors without a centralized controller or shared memory. Processors exchange information through message passing over communication links. A typical architecture for distributed computation is an Ethernet LAN, in which every workstation has its own CPU and memory. Figure 5.1 depicts this architecture, which is used for the proposed distributed algorithms in this chapter.

![Figure 5.1 Ethernet LAN with Multiple Workstations](image)

In the general approach for distributed algorithms, there are two types of processes, controllers or workers. A controller is a process with the following responsibilities:

- To accept user input
- To assign initial data (job) to workers
- To invoke workers to execute tasks
• To send/receive intermediate data to/from workers
• To do some system-wide, non-intensive calculations
• To terminate the algorithm at the appropriate time and notify workers to stop.

A worker is a process with the following responsibilities:
• To receive initial data from a controller
• To do intensive calculation upon a controller’s request
• To send/receive intermediate data to/from a controller
• To stop when notified.

In the proposed approach here there is only one controller, while there are usually multiple workers. The controller coordinates all workers, while workers are responsible for intensive computations. Also, the controller has system-wide knowledge, while a worker has knowledge of part of the system, i.e., the job assigned by the controller. In this sense, a controller behaves like a manager in a company, while a worker behaves like a technician, a salesman, or an engineer.

5.1.2 General Approach

A general approach is presented here for distributed algorithms dealing with tree-based topologies. This idea can be applied to power flow, short circuit or reliability assessment calculations for power distribution systems. The steps of this approach, assuming there is one controller and \( p \) workers, is as follows:
1. The controller breaks a tree into p sub-trees.

2. The controller sends each sub-tree to a remote worker along with initial conditions needed to start calculations.

3. The controller invokes remote calculations on workers.

4. Each worker processes its assigned sub-tree locally.

5. After completion, the intermediate results are sent back from workers to the controller.

6. After collecting results from workers, the controller performs some system-wide, non-intensive calculations related to sub-tree boundaries.

7. The controller sends updated boundary conditions to workers.

8. The controller asks all workers to check the convergence of their own sub-trees.

9. If the controller is informed that all workers have converged, the controller considers the whole system is converged and notifies all workers to stop. (Otherwise, the controller goes to step 4.)

Figure 5.2 illustrates the above steps using Unified Modeling Language (UML) notation [6], where CController is the controller class and CWorker is the worker class. Also, c is an object of class CController, and w is an object of class CWorker. The numbers in Figure 5.2 correspond to the numbers of the above steps.
5.1.3 Assumption of Tree Partition

In step 1 of the above approach, a tree is evenly partitioned into $p$ sub-trees. The work here does not address the method used to partition the system into $p$ sub-trees. In the following discussion, it is assumed that the partition is known. It is also assumed that each sub-tree has roughly the same number of elements.

Figure 5.3 illustrates the outcome of the job assignment for a radial power distribution system, where the “tree” is broken into 4 sub-trees and assigned to 4 workers, $w1$ to $w4$. 
5.2 Distributed Algorithm for Radial Power Flow

5.2.1 Review of Sequential Algorithm based on Tree Approach

The term sequential is used to indicate one processor, while the term distributed is used to indicate multiple processors.

A review of the topology of radial power systems follows. Chapter 2 describes a tree-based approach that considers each electric device as a two-port element. Hence, the radial distribution system can be viewed as a tree. In a tree, an element may have multiple succeeding elements. Except for the root element, every element has one and only one preceding element [34]. Here, a root is a power source such as a substation. A preceding element is
the connecting upstream element close to the root. A succeeding element is one of the connecting downstream elements that are close to ending loads.

The Radial Power Flow algorithm is based on the two-port element model and the tree traverse [38]. It is carried out by several iterations. Every iteration consists of a backward traverse, followed by a forward traverse of all the elements. The backward traverse calculates the currents through elements. The forward traverse will calculate the voltage drops across elements. These calculations are represented by the following equations, respectively.

\[
I = \sum I_m + \frac{S_{load}}{V_2^*} \quad (5-1)
\]

\[
V_2 = V_1 - IZ \quad (5-2)
\]

where

- \( I \) = current through the element;
- \( m \in \) all preceding elements;
- \( S_{load} \) = load attached to the element;
- \( V_2 \) = voltage at downstream port of the element;
- \( V_1 \) = voltage at upstream port of the element;
- \( Z \) = the impedance of the element.

This sequential algorithm for the Radial Power Flow is given as follows.

1. Starting from an ending element, backward traverse the tree element-by-element. Equation (5-1) is applied to calculate the current for each element.
2. Starting from the root element, forward traverse the tree element-by-element. Equation (5-2) is applied to calculate voltages for each element.

3. Check the convergence criteria of power flow. If converged, stop; otherwise, go back to step 1.

Note that the implementation of the traversing methods and the associated data structure could vary [38-39]. However, they must satisfy the following traversing principles.

- To calculate the current by Equation (5-1) at an element during a backward traverse, all of its succeeding elements should have completed current calculations.

- To calculate the voltage drop by Equation (5-2) at an element during a forward traverse, its preceding element should have completed its voltage calculation.

An example will briefly illustrate the above traversing principles. Considering a system as depicted in Figure 5.4, the following forward traverse sequences are legal.

0-1-2-3-4-5-6
0-1-2-5-6-3-4
0-1-2-3-5-4-6
0-1-2-5-3-6-4

The following backward traverse sequences are also legal.

6-5-4-3-2-1-0
4-3-6-5-2-1-0
5.2.2 Design of Distributed Algorithm for Radial Power Flow

A distributed algorithm for Radial Power Flow is presented here. It is transformed from the sequential algorithm for Radial Power Flow.

The UML sequence diagram in Figure 5.5 illustrates the algorithm. In this diagram, $c$ is an object of class CController, and $w1$ and $w2$ are two objects of class CWorker. Objects $w1$ and $w2$ correspond to those in Figure 5.3. Objects $w3$ and $w4$ are not shown here for simplicity.

After the job assignment, $w1$ and $w2$ will start backward traverses concurrently. Then, $w2$ sends the current information at its sub-tree root to $c$. CWorker $w1$ needs this information from $c$ to complete the backward traverse. Object $c$ serves like a data collector and dispatcher here.
Next, \( w_1 \) and \( w_2 \) start forward traverse concurrently. \( CWorker w_2 \) needs the voltage information at its sub-tree root to start the traverse, so \( CWorker w_1 \) must send such information to \( c \), the data collector and dispatcher. Then, \( c \) forwards this information to \( w_2 \).

After convergence criteria are met in both sub-trees handled by \( w_1 \) and \( w_2 \), respectively, the algorithm terminates.
Figure 5.5 Sequential Diagram for Distributed Algorithm of Radial Power Flow
Assume there is unique controller and \( p \) worker processes. The full description of the distributed algorithm is given as follows, where the step numbers match those in Figure 5.5.

1. Initialization of tree-breaking and job assignment
   1.1 The controller breaks the tree into \( p \) sub-trees.
   1.2 The controller sends each sub-tree to a remote worker along with initial conditions to start the load flow calculation.
   1.3 The controller invokes remote calculations on workers.

2. Backward traverse to calculate currents
   2.1 For workers
      2.1.1 Each worker traverses backward on its own sub-tree in the same way as the sequential algorithm.
      2.1.2 After completing one backward traverse, the current information at the root of its sub-tree will be reported to the controller.
   2.2 For the controller – The controller forwards the received current information of sub-tree roots to corresponding workers, which are waiting for the current information to complete backward traverses.

3. Forward traverse to calculate voltage drops
   3.1 For workers
      3.1.1 Each worker traverses forward along its own sub-tree in the same way as the sequential algorithm.
      3.1.2 After it completes the voltage at an element that is parent of another sub-tree, it will report the voltage information at this element to the controller.
3.2 For the controller - The controller forwards the received voltage information of connecting elements to corresponding workers, which are waiting for the voltage information at its sub-tree root to complete forward traverses.

4. Checking convergence
4.1 Each worker is requested to check the convergence criteria of its own sub-tree.
4.2 If the controller is informed all workers are converged, it will notify workers to stop. (Otherwise, it notifies all workers to iterate from Step 2.)

The next section discusses the convergence, complexity and efficiency of this distributed algorithm.

5.2.3 Convergence, Complexity and Efficiency of the Distributed Algorithm for Radial Power Flow

In many engineering and scientific calculations, convergence takes higher priority than speed or complexity. The design of a distributed algorithm should preserve the convergence characteristics of the original sequential algorithm. This also applies to the power flow calculation.

The distributed algorithm for Radial Power Flow does not change the mathematical kernel of the original sequential algorithm. All it does is to split the computation into small parts. Each part is carried out in a separate computer in parallel and the results are re-assembled through message
passing. Hence, the convergence characteristics are preserved. Next, the time complexity and efficiency are discussed.

Assume that the sequential power flow converges within $k$ iterations (from experiments typically $k=4\sim12$), each having one backward and one forward traverse. Hence, $2k$ traverses are needed. Also assume that the time to traverse an element is the same in the forward and backward calculations. The complexity for the sequential algorithm, involving $2k$ traverses of a tree system, is given as

$$T_{seq} = 2kan$$  \hfill (5-3)

where

- $k =$ the number of forward traverses;
- $= =$ the number of backward traverses;
- $a =$ the time to backward traverse one element
- $= =$ the time to forward traverse one element;
- $n =$ the number of elements in the system.

Assume there are $p$ processors for the distributed algorithm. Since the controller process is computationally lightly loaded, one processor runs the controller process and a worker process for maximum CPU usage. Each of the remaining $p-1$ processors runs a worker process. Hence, there are $p+1$ processes in $p$ processors.

The time complexity consists of two parts. One is for computation, and the other is for communication. Since the traverse is carried out in parallel at $p$ workers, the complexity for computation is $1/p$ of the sequential execution. Hence, we have

$$T_{dist\_comp} = \frac{2kan}{p}$$  \hfill (5-4)
Next consider the communication overhead during the initial job assignment. During initialization, the controller sends \( p \) sub-trees to each worker. Each sub-tree takes roughly \( n/p \) elements (and \( n/p \) unit information). So the communication overhead for initialization is given as

\[
T_{\text{dist\_comm\_init}} = p \times \left( b \times n/p \right) = bn \quad (5-5)
\]

where \( b \) = the time of sending/receiving the information of one element.

The communication overhead during the computation is much smaller. In the backward traverse, each worker needs to pass one unit of information current at its sub-tree root to the controller. Then, the controller forwards it to other workers. Hence, the communication overhead for \( k \) backward traverses is given as

\[
T_{\text{dist\_comm\_bt}} = k \times \left( b \times p + b \times p \right) = 2kbp \quad (5-6)
\]

The forward traverse has the same communication overhead as in Equation (5-4). So, we have

\[
T_{\text{dist\_comm\_ft}} = 2kbp \quad (5-7)
\]

The overall communication overhead is given as

\[
T_{\text{dist\_comm}} = T_{\text{dist\_comm\_init}} + T_{\text{dist\_comm\_bt}} + T_{\text{dist\_comm\_ft}} = bn + 4kbp \quad (5-8)
\]

Practically, \( n >> 4kp \) for \( p < 64 \). This gives

\[
T_{\text{dist\_comm}} = bn \quad (5-9)
\]

The overall time complexity for the distributed algorithm is given by

\[
T_{\text{dist}} = T_{\text{dist\_comp}} + T_{\text{dist\_comm}} = 2kan/p + bn \quad (5-10)
\]
The speedup [24] is given as
\[ S_R = \frac{T_{seq}}{T_{dist}} = \frac{p}{1 + \frac{bp}{2ka}} \]  (5-11)

The efficiency [24] is given as
\[ E_R = \frac{S_R}{p} = \frac{1}{1 + \frac{bp}{2ka}} \]  (5-12)

The next section presents the test results of sequential and distributed algorithms for Radial Power Flow.

5.2.4 Test Results

Based on a prototype of PowerFrame, two applications are developed here. One is for the sequential Radial Power Flow algorithm and the other is for the distributed Radial Power Flow algorithm. Test results for these two programs are presented. The power flow presented here is a single-phase power flow. Computation for a three-phase power flow is nine times as intensive as a single-phase power flow.

A) Development and Testing Environment

Both sequential and distributed programs are developed with Microsoft Visual C++ 6.0 on a Windows NT 4.0 platform.

Tests are run on the following eight machines, connected in the 10Based-T Ethernet LAN at Electrical Distribution Design Inc.

101: Intel P2-300 MHZ, 128 MB RAM
102: Intel P2-300 MHZ, 128 MB RAM
103: Intel P2-200 MHZ, 256 MB RAM
104: Intel P2-200 MHZ, 64 MB RAM
105: Intel P3-700 MHZ, 128 MB RAM
106: Intel P3-700 MHz, 128 MB RAM
107: Intel P2-200 MHZ, 64 MB RAM
108: Intel P2-300 MHZ, 128 MB RAM

The sequential program is installed and run on each machine individually. The test data are averaged.

The distributed power flow program consists of a controller part and a worker part, distributed among the above machines. Machine 101 runs a controller process and a worker process simultaneously. At each of the remaining machines, only a worker process is run. These workers communicate with machine 101 through TCP/IP protocol. Tests are run with 2, 4, and 8 machines.

B) Test Systems

A system containing 40,000 elements is used. Figure 5.6 illustrates the configuration of this system. Each substation has four feeders. Each feeder has an electric load of approximately 1.0 MVA and consists of
approximately 5,000 elements. It is quite straightforward to break this system into two, four, or eight parts to fit the distributed computing environment. All co-tree switches are open in this case.

C) Test Results for Sequential and Distributed Algorithms

The following results come from four tests. Results presented are averaged. The averaged data is used for comparison.

First, the sequential power flow is run on all machines. It takes 12 iterations to converge, because this is a heavily loaded system. The running time in seconds is shown in Table 5-1.

<table>
<thead>
<tr>
<th>Test</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing</td>
<td>25.64</td>
<td>25.90</td>
<td>25.46</td>
<td>26.01</td>
<td>25.75</td>
</tr>
</tbody>
</table>

Tables 5-2, 5-3 and 5-4 show the running time for the distributed Radial Power Flow algorithm with 2, 4, and 8 computers. All tests take 12 iterations to converge as in the sequential algorithm.

Table 5-2 Running Time of the Distributed Algorithm for Radial Power Flow with 2 Processors

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>0.67</td>
<td>0.68</td>
<td>0.69</td>
<td>0.82</td>
<td>0.72</td>
</tr>
<tr>
<td>Computation</td>
<td>12.77</td>
<td>12.83</td>
<td>12.87</td>
<td>12.74</td>
<td>12.80</td>
</tr>
<tr>
<td>Total</td>
<td>13.44</td>
<td>13.51</td>
<td>13.56</td>
<td>13.56</td>
<td>13.52</td>
</tr>
</tbody>
</table>
Table 5-3 Running Time of the Distributed Algorithm for Radial Power Flow with 4 Processors

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>1.03</td>
<td>1.01</td>
<td>1.02</td>
<td>1.04</td>
<td>1.03</td>
</tr>
<tr>
<td>Computation</td>
<td>6.49</td>
<td>6.49</td>
<td>6.38</td>
<td>6.37</td>
<td>6.43</td>
</tr>
<tr>
<td>Total</td>
<td>7.52</td>
<td>7.50</td>
<td>7.40</td>
<td>7.41</td>
<td>7.46</td>
</tr>
</tbody>
</table>

Table 5-4 Running Time of the Distributed Algorithm for Radial Power Flow with 8 Processors

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>1.46</td>
<td>1.20</td>
<td>1.20</td>
<td>1.19</td>
<td>1.26</td>
</tr>
<tr>
<td>Computation</td>
<td>3.63</td>
<td>3.28</td>
<td>3.30</td>
<td>3.30</td>
<td>3.38</td>
</tr>
<tr>
<td>Total</td>
<td>5.09</td>
<td>4.48</td>
<td>4.50</td>
<td>4.49</td>
<td>4.64</td>
</tr>
</tbody>
</table>

Speed-up and efficiency are given in Table 5-5.

Table 5-5 Speed-up and Efficiency of the Distributed Algorithm for Radial Power Flow

<table>
<thead>
<tr>
<th># of processors</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Assignment (sec)</td>
<td>--</td>
<td>0.72</td>
<td>1.03</td>
<td>1.26</td>
</tr>
<tr>
<td>Computing (sec)</td>
<td>25.75</td>
<td>12.80</td>
<td>6.43</td>
<td>3.38</td>
</tr>
<tr>
<td>Total CPU Time (sec)</td>
<td>25.75</td>
<td>13.52</td>
<td>7.46</td>
<td>4.64</td>
</tr>
<tr>
<td>Speedup</td>
<td>1</td>
<td>1.90</td>
<td>3.45</td>
<td>5.55</td>
</tr>
<tr>
<td>Efficiency (%)</td>
<td>100</td>
<td>95</td>
<td>86</td>
<td>69</td>
</tr>
</tbody>
</table>

From Table 5-5 the following observations can be drawn.

- The overall running time is significantly decreased with the distributed algorithm.
- The computation time dominates the communication overhead time. So, this is a computationally intensive algorithm.
- The computation time is linearly scaled down.
- The time of communication overhead may go up slightly when more processors are involved.
5.3 Distributed Algorithm for Looped Power Flow

5.3.1 Sequential Algorithm based on Sensitivity Matrix

In power distribution systems, there may be closed switches that connect two or more radial power distribution feeders. Those closed switches create loops in the systems. They are referred to as co-tree elements [38]. The number of co-tree elements in a distribution feeder or network may reach 50.

To calculate the power flow in this scenario, many power flow algorithms have been presented. The method proposed in [38] is used as our sequential algorithm here and will be transformed to a distributed algorithm. This method is based on the Radial Power Flow method in conjunction with a Sensitivity Matrix. The description of this sequential Looped Power Flow algorithm follows.

1. Break the network into a number of radial trees by disconnecting the co-tree elements, usually switches.
2. Converge each radial tree using the Radial Power Flow method.
3. Inject one ampere into the radial circuits at each co-tree element.
5. Calculate the Sensitivity Matrix elements as the change in voltage divided by the current injection.
6. Calculate the change of injection current at each co-tree element.
7. Calculate the expected change in voltage and add this to the connection points.
8. Converge each of the connecting circuits using the Radial Power Flow.

9. Check convergence. If converged, stop; else, go to step 5.

This algorithm is highly parallelizable because the key idea here is to break the looped tree into many radial trees. Hence, each radial tree can be solved locally and independently in parallel with the other radial trees.

5.3.2 Design of Distributed Algorithm for Looped Power Flow

A distributed version of the above algorithm is presented here. It is similar to the key idea of the distributed version of the Radial Power Flow in Section 5.2. In the following description, assume there are a unique controller and $p$ workers, as defined in Section 5.1.

1. The controller breaks the circuit into $p$ radial sub-tree circuits by disconnecting co-tree elements. Then, the controller assigns sub-trees to each worker.

2. The controller and the workers work together until all Radial Power Flow calculations converge. Here, the distributed algorithm of Radial Power Flow is used.

3. The controller sets the injected current at each co-tree switch to 1 ampere.

4. The controller and workers again complete the Radial Power Flow distributed algorithm.
5. The controller calculates the Sensitivity Matrix elements as the change in voltage divided by the current injections.

6. The controller calculates the change of injection current at each co-tree element.

7. The controller calculates the expected change in voltage. It adds this to the connection points and sends to workers.

8. The controller and workers complete the Radial Power Flow distributed algorithm.

9. The convergence criteria are checked. If converged, stop; otherwise, return to step 5.

In this algorithm, the controller has more responsibility than in the distributed Radial Power Flow algorithm. It needs to calculate the Sensitivity Matrix, which is required by the Looped Power Flow. The controller is responsible for this since it has global knowledge of the system, while a worker has only local knowledge of its own sub-tree circuit.

5.3.3 Convergence, Complexity and Efficiency of Distributed Algorithm for Looped Power Flow

As stated in Section 5.2.3, the distributed algorithm for the Looped Power Flow does not change the mathematical characteristics. Hence, it does not affect the convergence features. Practical experience demonstrates that the sequential program converges in about 4~6 iterations of the Sensitivity Matrix calculation [38]. Please note that within each Sensitivity Matrix calculation, a Radial Power Flow needs to be run. Also, the last iteration of
the power flow may take a less number of traverses than the first iteration of the power flow, due to the different injection currents and initial system states. The total number of backward or forward traverse is 8~35.

Assume there are $k'$ forward and $k'$ backward traverses in total. Therefore, there are $2k'$ system traverses in the Looped Power Flow. Then we have the complexity of the sequential algorithm given by

$$T_{\text{seq}} = 2jk'n \quad (5-13)$$

With the same assumption as in Section 5.2.3, there are $p$ processors running $p+1$ processes in the distributed algorithms. Similarly, the time complexity for $T_{\text{dist comp}}$, $T_{\text{dist comm ft}}$, and $T_{\text{dist comm bt}}$ are given in following equations.

$$T_{\text{dist comp}} = 2k'an/p \quad (5-14)$$
$$T_{\text{dist comm bt}} = T_{\text{dist comm ft}} = 2k'bp \quad (5-15)$$
$$T_{\text{dist comm init}} = bn \quad (5-16)$$

$$T_{\text{dist comm}} = T_{\text{dist comm init}} + T_{\text{dist comm ft}} + T_{\text{dist comm bt}} = bn + 4k'bp \quad (5-17)$$

Since $n >> 4k'p$ for $p < 64$, $T_{\text{dist comm ft}}$ and $T_{\text{dist comm bt}}$ are much smaller than $T_{\text{dist comm init}}$. Equation (5-17) can be simplified as

$$T_{\text{dist comm}} = bn \quad (5-18)$$

The overall time complexity of the distributed algorithm is given as

$$T_{\text{dist}} = T_{\text{dist comp}} + T_{\text{dist comm}} = 2k'an/p + bn \quad (5-19)$$

The speedup and the efficiency are given as

$$S_L = T_{\text{seq}}/T_{\text{dist}} = p/(1 + bp/2k'a) \quad (5-20)$$
$$E_L = S_L/p = 1/(1 + bp/2k'a) \quad (5-21)$$
Compared with Equation (5-12), $E_L$ is greater than $E_R$ because of the number of traverse $k'$ is 2~4 times as great as $k$. This matches the test results presented in the following section.

### 5.3.4 Test Results

As in Section 5.2.4, two applications are developed to solve the Looped Power Flow based on the prototype of **PowerFrame**. One is for the sequential Looped Power Flow algorithm and the other is for the distributed Looped Power Flow algorithm. Test results for these two programs are presented here. The testing environment and testing system are the same as those in Section 5.3.4. However, the open switches in the system shown in Figure 5.6 are all closed for the Looped Power Flow.

The power flow for this looped system takes 4 iterations of Sensitivity Matrix calculation (4 Radial Power Flows are needed). These 4 Radial Power Flows take 12, 8, 6 and 5 backward/forward traverses to converge, respectively.

First, the sequential algorithm for the Looped Power Flow is run. The running time in seconds is shown in Table 5-6.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>61.49</td>
<td>62.62</td>
<td>61.94</td>
<td>61.67</td>
<td>61.93</td>
</tr>
</tbody>
</table>
Tables 5-7, 5-8 and 5-9 show the running time of the distributed algorithm for Looped Power Flow with 2, 4 and 8 processors. For each case, one processor runs a controller and a worker for efficient usage of CPU, while each of the remaining $p-1$ processors runs a worker only.

### Table 5-7 Running Time of the Distributed Algorithm for Looped Power Flow with 2 Processors

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>0.99</td>
<td>1.03</td>
<td>0.97</td>
<td>0.90</td>
<td>0.97</td>
</tr>
<tr>
<td>Computation</td>
<td>32.71</td>
<td>33.01</td>
<td>31.98</td>
<td>32.19</td>
<td>32.47</td>
</tr>
<tr>
<td>Total</td>
<td>33.70</td>
<td>34.04</td>
<td>32.95</td>
<td>33.09</td>
<td>33.44</td>
</tr>
</tbody>
</table>

### Table 5-8 Running Time of the Distributed Algorithm for Looped Power Flow with 4 Processors

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>1.26</td>
<td>1.28</td>
<td>1.22</td>
<td>1.02</td>
<td>1.20</td>
</tr>
<tr>
<td>Computation</td>
<td>16.23</td>
<td>17.02</td>
<td>16.88</td>
<td>16.54</td>
<td>16.67</td>
</tr>
<tr>
<td>Total</td>
<td>17.49</td>
<td>18.30</td>
<td>18.10</td>
<td>17.56</td>
<td>17.87</td>
</tr>
</tbody>
</table>

### Table 5-9 Running Time of the Distributed Algorithm for Looped Power Flow with 8 Processors

<table>
<thead>
<tr>
<th></th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
<th>Test 4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>1.54</td>
<td>1.65</td>
<td>1.22</td>
<td>1.30</td>
<td>1.43</td>
</tr>
<tr>
<td>Computation</td>
<td>7.91</td>
<td>7.84</td>
<td>7.84</td>
<td>7.88</td>
<td>7.87</td>
</tr>
<tr>
<td>Total</td>
<td>9.45</td>
<td>9.49</td>
<td>9.06</td>
<td>9.18</td>
<td>9.30</td>
</tr>
</tbody>
</table>

Speed-up and efficiency is given as in Table 5-10.

### Table 5-10 Speed-up and Efficiency of the Distributed Algorithm for Looped Power Flow

<table>
<thead>
<tr>
<th># of processors</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Assignment (sec)</td>
<td>--</td>
<td>0.97</td>
<td>1.20</td>
<td>1.43</td>
</tr>
<tr>
<td>Computing (sec)</td>
<td>61.93</td>
<td>32.47</td>
<td>16.67</td>
<td>7.87</td>
</tr>
<tr>
<td>Total Time (sec)</td>
<td>61.93</td>
<td>33.44</td>
<td>17.87</td>
<td>9.30</td>
</tr>
<tr>
<td>Speedup</td>
<td>1.00</td>
<td>1.85</td>
<td>3.47</td>
<td>6.66</td>
</tr>
<tr>
<td>Efficiency (%)</td>
<td>100</td>
<td>93</td>
<td>87</td>
<td>83</td>
</tr>
</tbody>
</table>

120
Similar conclusions can be drawn from Table 5-10 as from Table 5-5. Also, from these two tables, the ratio of $b/a$ can be calculated in the range of (1.0~2.0). Taking 1.5 as the average value of $b/a$, efficiency can be verified roughly based on Equations (5-12) and (5-21).

The comparison of results in Tables 5-5 and 5-10 follows. The Loop Power Flow calculation is more computationally intensive than the Radial Power Flow, while the communication overhead for these two cases is close. Therefore, the communication overhead weights much less in the Loop Power Flow. Thus, a better speed-up and efficiency are expected for the Loop Power Flow calculation. This is also shown in Figure 5.7, which displays the efficiency curves of distributed algorithms for the Radial Power Flow and the Loop Power Flow. The Loop Power Flow has a flatter curve when the number of processors goes up.

![Efficiency Comparison of Distributed Algorithms](image)

**Figure 5.7 Efficiency Comparison of Distributed Algorithms of Radial Power Flow and Loop Power Flow**
5.4 Summary

The following summary can be drawn from this chapter:

- Tree-based power flow is a computationally intensive engineering problem that is naturally suited for parallel and distributed computing.
- Communication delay in Ethernet LANs is unlikely to hurt the overall performance.
- The distributed algorithm of the Looped Power Flow is more computationally intense and more efficient than the distributed algorithm for Radial Power Flow.

Some additional comments follows:

- The communication overhead would be even smaller if Fast Ethernet and Gigabit Ethernet are applied for the distributed algorithms. They are 10 and 100 times faster, respectively, than 10BASE-T Ethernet.
- The distributed algorithms are more efficient for three-phase power flows, because their computational scale is 9 times as large as that of single-phase power flows. Hence, in the three-phase power flows, computation time dominates communication time more than in the case of single-phase power flows.
- The Power Flow distributed algorithms can be applied to some time-critical computations such as real-time power flows.
- The Power Flow distributed algorithms would be valuable for time-consuming computations such as network reconfigurations and optimizations.
Chapter 6
Conclusions

6.1 Concluding Remarks

This dissertation presents the development of four inter-related works:

A framework for power distribution system analysis, referred to as *PowerFrame*. This work presents a high-level design of *PowerFrame*, a software framework abstracting the generic rules in power distribution system analysis. *PowerFrame* has a layered architecture, which aims to achieve reusability and extensibility for application development. All classes in each layer implement the same interface so that the whole layer performs uniformly from the point of view of the outside world. Also, the layered architecture follows a strict top-down dependency relation, which can mitigate the coupling between different layers.

Impedance model for composite cables, integrated into *PowerFrame* as a subsystem. As part of the detailed design of *PowerFrame*, this work presents a design of a subsystem to model composite cables placed in parallel-run ducts. The Exact Method for composite cables considers all possible couplings between any two conductors. The calculation shows considerable errors of an Approximate Method used in common practice. The subsystem design for cables in *PowerFrame* employs the Composite
design pattern to model complicated cables. Flexibility and efficiency are achieved in this design.

**Three-phase, unbalanced model for Distributed Resources as an extension of PowerFrame.** Distributed Resources (DRs) are used as a test bed for PowerFrame in terms of extensibility and reusability. This work presents two different DR models that consider DRs as constant voltage sources or voltage-dependent current sources, respectively. Imbalance issues are addressed in the analytical models. Then, a design extending PowerFrame to implement DR models is described. This design can work with the original framework in a seamless fashion without breaking the semantic boundary.

**Distributed algorithms for Radial and Looped Power Flow calculations.** PowerFrame models distributed computing which could speed up power system calculations significantly. Here distributed algorithms are proposed for Radial and Looped Power Flows, which are the fundamental calculations of power system analysis. The sequential power flow algorithms based on a tree approach are adapted to distributed algorithms. Preserving the convergence characteristics of the sequential algorithms, the distributed algorithms can achieve significant speedup over sequential algorithms due to low communication overhead.

### 6.2 Contributions

The contributions of this work are summarized as follows:
A design of a software framework for power distribution system analysis with a layered architecture. This is the first proposal of using a framework to achieve reusability and extensibility for application development in the domain of power system analysis. The proposed framework, *PowerFrame*, has a four layered architecture and contains generic characteristics of the domain. *PowerFrame* focuses on the software design reuse for the entire domain, while previous works focus on a specific problem (application).

An exact impedance model of composite cables designed as the Composite pattern. An impedance model of cables in parallel run ducts, referred to as composite cables, is presented here. This work considers the coupling between any two conductors, while previous works often ignore the coupling between conductors in separated ducts. This is the first report of employing the Composite design pattern to model composite cables.

Unbalanced model of Distributed Resource generation. This is the first work presenting unbalanced, three-phase models for Distributed Resources (DRs). The imbalance studies show that DRs may have a negative effect on imbalance if modeled as constant voltage sources. It also shows that DRs may in certain situations raise the voltage over the limits if modeled as voltage-dependent current sources.

Distributed Algorithms of Radial and Looped Power Flow calculations. This is the first report of distributed algorithms for Radial and Looped Power Flow calculations in power distribution systems. Previous works focus on the power flow for transmission systems with a matrix approach, while this work addresses power distribution systems with a tree approach.
Also, most previous works deal with parallel computing carried out in high-performance supercomputers. In addition, this work presents theoretical derivations and testing results of efficiency and speedup, while previous works present testing results only.

6.3 Future work

The following work is recommended as possible future improvements.

A more detailed design and implementation of PowerFrame. This work presents a high-level, architectural design of PowerFrame. There are many more details not covered here. A more detailed design is necessary to make the concept of framework widely accepted. Also, an implementation can directly benefit all application developers working in the domain of power engineering.

Testing of composite cable impedance within large systems. Current work checks the errors of the Approximate Method for impedance calculations and power flows for a simple power distribution feeder. Tests with real, complicated power distribution systems with thousands of elements will make the investigation stronger.

Testing of DR models within large systems. Current work is based on simple, unbalanced systems. Tests with large systems using DRs may uncover further discoveries on the imbalance issue of DRs.

Testing of distributed algorithms for power flows at different network communication loads. The current work does not address the communication loads of the Ethernet LAN. It is certain that communication...
overhead can be greater if the computer network is loaded more heavily. It is interesting to find the relation between the overall performance (efficiency) and the communication network load.

**Applying distributed algorithms for power flows to solve discrete optimization problems, such as reconfigurations of power distribution systems.** Many of the discrete optimization problems involve intensive power flow calculations. Since power flow is a fundamental base of such problems, distributed Power Flow algorithms are likely to be beneficial for solving problems, such as reconfigurations.
Reference


Fangxing Li was born on November 18, 1973 in Tongcheng, Anhui Province, China. He received his BSEE and MSEE degrees from Southeast University, Nanjing, Jiangsu Province, China, in July 1994 and April 1997, respectively. He is working for his PHD degree at Virginia Tech, under the supervision of Dr. Robert Broadwater in the areas of computer applications for power distribution analysis. Since February 2001, he has been working at ABB as a Senior Engineer in the area of computer applications in power systems.