Analysis and Control System Techniques for Electric Power Systems

Part 2 of 4

CONTROL AND DYNAMIC SYSTEMS

Advances in Theory and Applications

Volume 42

CONTRIBUTORS TO THIS VOLUME

ROSS BALDICK
ANJAN BOSE
G. S. CHRISTENSEN
MARIESA L. CROW
M. J. DAMBORG
J. ENDRENYI
MOTOHISA FUNABASHI
ANIL K. JAMPALA
P. KUNDUR
KWANG Y. LEE
Y. MANSOUR
TAKUSHI NISHIYA
YOUNG MOON PARK
S. A. SOLIMAN
DANIEL J. TYLAVSKY
S. S. VENKATA
LU WANG

CONTROL AND DYNAMIC SYSTEMS

ADVANCES IN THEORY AND APPLICATIONS

Edited by
C. T. LEONDES

Department of Electrical Engineering
University of Washington
Seattle, Washington
and
School of Engineering and Applied Science
University of California, Los Angeles
Los Angeles, California

VOLUME 42: ANALYSIS AND CONTROL SYSTEM TECHNIQUES FOR ELECTRIC POWER SYSTEMS
Part 2 of 4

ACADEMIC PRESS, INC.
Harcourt Brace Jovanovich, Publishers
San Diego New York Boston
London Sydney Tokyo Toronto

Academic Press Rapid Manuscript Reproduction

This book is printed on acid-free paper. ♾

Copyright © 1991 By ACADEMIC PRESS, INC.
All Rights Reserved.
No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

Academic Press, Inc.
San Diego, California 92101

United Kingdom Edition published by
ACADEMIC PRESS LIMITED
24-28 Oval Road, London NW1 7DX

Library of Congress Catalog Card Number: 64-8027

ISBN 0-12-012742-3 (alk. paper)

PRINTED IN THE UNITED STATES OF AMERICA
91 92 93 94 9 8 7 6 5 4 3 2 1

CONTENTS

CONTRIBUTORS

Numbers in parentheses indicate the pages on which the authors' contributions begin.

Ross Baldick (245), *Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720*

Anjan Bose (1), *Department of Electrical Engineering, Arizona State University, Tempe, Arizona 85287*

G. S. Christensen (371), *Department of Electrical Engineering, University of Alberta, Edmonton, Alberta, Canada T6G 2G7*

Mariesa L. Crow (1), *Department of Electrical Engineering, Arizona State University, Tempe, Arizona 85287*

M. J. Damborg (57), *Department of Electrical Engineering, University of Washington, Seattle, Washington 98195*

J. Endrenyi (163), *Ontario Hydro Research Division, Toronto, Ontario M8Z 5S4, Canada*

Motohisa Funabashi (341), *Systems Development Laboratory, Hitachi, Ltd., Kawasaki, 215 Japan*

Anil K. Jampala (57), *ESCA Corporation, Bellevue, Washington 98004*

P. Kundur (111), *Ontario Hydro Research Division, Toronto, Ontario M8Z 5S4, Canada*

Kwang Y. Lee (293), *Department of Electrical and Computer Engineering, Pennsylvania State University, University Park, Pennsylvania 16802*

Y. Mansour (111), *PowerTech Laboratories, Inc., Surrey, British Columbia V3W 7R7, Canada*

Young Moon Park (293), *Department of Electrical Engineering, Seoul National University, Seoul 151, Korea*

S. A. Soliman (371), *Electrical Power and Machines Department, Ain Shams University, Abbassia, Cairo, Egypt*

Daniel J. Tylavsky (1), *Department of Electrical Engineering, Arizona State University, Tempe, Arizona 85287*

S. S. Venkata (57), *Department of Electrical Engineering, University of Washington, Seattle, Washington 98195*

Lu Wang (163), *Ontario Hydro Research Division, Toronto, Ontario M8Z 5S4, Canada*

PREFACE

Research and development in electric power systems analysis and control techniques has been an area of significant activity for decades. However, because of increasingly powerful advances in techniques and technology, the activity in electric power systems analysis and control techniques has increased significantly over the past decade and continues to do so at an expanding rate because of the great economic significance of this field. Major centers of research and development in electrical power systems continue to grow and expand because of the great complexity, challenges, and significance of this field. These centers have become focal points for the brilliant research efforts of many academicians and industrial professionals and the exchange of ideas between these individuals. As a result, this is a particularly appropriate time to treat advances in the many issues and modern techniques involved in electric power systems in this international series. Thus, this is the second volume of a four volume sequence in this series devoted to the significant theme of "Analysis and Control System Techniques for Electric Power Systems." The broad topics involved include transmission line and transformer modeling. Since the issues in these two fields are rather well in hand, although advances continue to be made, this four volume sequence will focus on advances in areas including power flow analysis, economic operation of power systems, generator modeling, power system stability, voltage and power control techniques, and system protection, among others.

The first contribution to this volume, "Concurrent Processing in Power System Analysis," by Mariesa L. Crow, Daniel J. Tylavksy, and Anjan Bose, deals with the application of parallel processing to power system analysis as motivated by the requirement for faster computation. This is due to interconnected generation and transmission systems that are inherently very large and that result in problem formulations tending to have thousands of equations. The most common analysis problem, the power flow problem, requires the solution of a large set of nonlinear algebraic equations, approximately two for each mode. Other important problems of very substantial computational complexity include the optimal power flow problem, transient stability. In the case of transient stability problems, a 2,000 bus power network with 300 machines can require on the order of 3,000 differential

equations and 4,000 (nonlinear) algebraic equations. Other application areas include short circuit calculations, steady-state stability analysis, reliability calculations, production costing, and other applications. This contribution focuses on techniques for the application of parallel computer methods to these large-scale power system problems which require such methods. Other contributions to this four volume sequence that treat the large-scale power system present methods and algorithms that are potentially applicable to parallel computers.

The next contribution, "Power System Protection: Software Issues," by S.S. Venkata, M. J. Damborg, and Anil K. Jampala, provides a rather comprehensive review and analysis of the past, present, and future of power system protection from a software point of view. Next generation power systems and beyond will operate with minimal spinning margins, and energy transportation will take place at critical levels due to environmental and economic constraints. These factors and others dictate that power systems be protected with optimum sensitivity, selectivity, and time of operation in order to assure maximum reliability and security at minimal costs. Naturally, one of the keys to all this and more will be the associated software issues, as treated in this contribution.

The voltage stability phenomenon has emerged as a major problem currently being experienced by the electric utility industry. The next contribution, "Voltage Collapse: Industry Practices," by Y. Mansour and P. Kundur, presents a rather comprehensive review and analysis of this problem of voltage stability. Major outages attributed to this problem have been experienced on a worldwide basis, and two in-depth surveys of this phenomenon have been conducted on the international scene. Consequently, major challenges in establishing sound analytical procedures and quantitative measures of proximity to voltage are issues facing the industry. This contribution will be an invaluable source reference for researchers and practicing engineers working in this problem area of major significance.

In the next chapter, "Reliability Techniques in Large Electric Power Systems," by Lu Wang and J. Endrenyi, an overview is given of the techniques used in the reliability evaluation of large electric power systems. Particular attention is paid to the reliability assessment of bulk power systems which are the composite of generation and high-voltage transmission (hence often called composite systems). Reliability modeling and solution methods used in these systems are unusually complex. This is partly because of the sheer size of bulk power systems, which usually consist of hundreds, possibly thousands, of components, and partly because of the many ways these systems can fail and the multiplicity of causes for the failures. At an EPRI-sponsored conference in 1978, the observation was made that while reliability methods for other parts of the power system were reasonably well developed, the methods for transmission and composite systems were still in an *embryonic* stage. The reasons were the same difficulties as those mentioned above. Impressive efforts have been made since then to close the gap, and this review attempts to reflect this development. In fact, this chapter can be considered an

update of the relevant chapters in the 1978 book by the second author, *Reliability Modeling in Electric Power Systems*, published by John Wiley & Sons.

External forces such as higher fuel costs, deregulation, and increasing consumer awareness are changing the role of electric utilities and putting pressure on them to become more "efficient." Until recently, increases in efficiency were mostly due to improving generation technology; however, the potential for such improvements has been almost completely exploited. Efficiency improvements are increasingly due to nongeneration technologies such as distribution automation systems, which increase the options for real-time computation, communication, and control. This technology will prompt enormous changes in many aspects of electric power system operation. The next contribution, "Coordination of Distribution System Capacitors and Regulators: An Application of Integer Quadratic Optimization," by Ross Baldick, investigates the potential of such technology to improve efficiency in a radial electric distribution system through the coordination of switched capacitors and regulators. System performance criteria and constraints are, of course, examined in depth, and the relationship between the capacitor and regulation expansion design problem and the coordination problem is carefully considered.

The optimal operation of a power system requires judicious planning for use of available resources and facilities to their maximum potential before investing in additional facilities. This leads to the operational planning problem. The purpose of the operational planning problem is to minimize the fuel costs, system losses, or some other appropriate objective functions while maintaining an acceptable system performance in terms of voltage profile, contingencies, or system security. The operational planning problem was first formulated as an optimal power flow problem by selecting the fuel cost as the objective function and the network or load-flow equations as constraints. The problem was solved for an optimal allocation of real power generation to units, resulting in an economic dispatch. Recently, voltage stability or voltage collapse has been an increasingly important issue to utility as the power system is approaching its limit of operation due to economical and environmental constraints. Generators alone can no longer supply the reactive power that is needed to maintain the voltage profile within the allowed range throughout the power system. Additional reactive power or var sources need to be introduced and coordinated with generators. This has motivated many researchers to formulate optimal reactive power problems, wherein the system loss is used as an objective function, resulting in an economic reactive power dispatch. The next contribution, "Optimal Operational Planning: A Unified Approach to Real and Reactive Power Dispatches," by Kwang Y. Lee and Young Moon Park, is an in-depth treatment of these issues which are substantially complicated as a result of the large system scale nature of these problems.

The development of optimization methods has a long history. However, algorithmic innovation is still required particularly for operating large-scale dynamic plants, which are characteristic of electric power systems. Because of the high

dimensionality of the plants, technological bases for the problem in operating such large plants are usually found in the area of linear programming. In applying linear programming for optimal dynamic-plant operation, the constraint appears in the form of a staircase structure. In exploiting this structure, several attempts at devising efficient algorithms have been made. However, when applied to the large-scale system problem area of operational scheduling in electric power systems, significantly greater improvements in speed are required. The next contribution, "Multistage Linear Programming Methods for Optimal Energy Plant Operation," by Takushi Nishiya and Motohisa Funabashi, presents techniques for achieving these requisite speed improvements, which are so essential to electric power systems.

The hydro optimization problem involves planning the use of a limited resource over a period of time. The resource is the water available for hydro generation. Most hydroelectric plants are multipurpose. In such cases, it is necessary to meet certain obligations other than power generation. These may include a maximum forebay elevation not to be exceeded because of the danger of flooding and a minimum plant discharge and spillage to meet irrigational and navigational commitments. Thus, the optimum operation of the hydro system depends upon the conditions that exist over the entire optimization interval. Other distinctions among power systems are the number of hydro stations, their location, and special operating characteristics. The problem of determining the optimal long-term operation of multireservoir power systems has been the subject of numerous publications over the past forty years, and yet no completely satisfactory solution has been obtained, since in every publication the problem has been simplified in order to be solved. The next contribution, "Optimization Techniques in Hydroelectric Systems," by G.S. Christensen and S.A. Soliman, presents an in-depth treatment of issues on effective techniques in this broadly complex area.

This volume is a particularly appropriate one as the second of a companion set of four volumes on analysis and control techniques in electric power systems. The authors are all to be commended for their superb contributions, which will provide a significant reference source for workers on the international scene for years to come.

Concurrent Processing in Power System Analysis

Mariesa L. Crow, Daniel J. Tylavsky, and Anjan Bose

Electrical Engineering Department
Arizona State University
Tempe, Arizona 85287-5706

I Introduction

The application of parallel processing to power systems analysis is motivated by the desire for faster computation. Except for those analytical procedures that require repeat solutions, like contingency analysis, there are few obvious parallelisms inherent in the mathematical structure of power system problems. Thus, for a particular problem a parallel (or near-parallel) formulation has to be found that is amenable to formulation as a parallel algorithm. This solution has then to be implemented on a particular parallel machine keeping in mind that computational efficiency is dependent on the suitability of the parallel architecture to the parallel algorithm.

The interconnected generation and transmission system is inherently large and any problem formulation tends to have thousands of equations. The most common analysis, the power flow problem, requires the solution of a large set of nonlinear algebraic equations approximately two for each node. The traditional method of using successive linearized solutions (Newton's method) exploits the extreme sparsity of the underlying network connectivity to gain speed and conserve storage. Parallel algorithms for handling dense matrices are not competitive with sequential sparse matrix methods, and,

CONTROL AND DYNAMIC SYSTEMS, VOL. 42
Copyright © 1991 by Academic Press, Inc.
All rights of reproduction in any form reserved.

since the pattern of sparsity is irregular, parallel sparse matrix methods have been difficult to develop. The power flow problem defines the steady state condition of the power network and thus, the formulation (or some variation) is a subset of several other important problems like the optimal power flow or transient stability. Hence an effective parallelization of the power flow problem is essential if these other problems are to be handled efficiently.

The transient stability program is used extensively for off-line studies but has been too slow for on-line use. A significant speed up by parallel processing, in addition to the usual efficiencies, will allow on-line transient stability analysis, a prospect that has spurred research in this area. The transient stability problem requires the solution of differential equations that represent the dynamics of the rotating machines together with the algebraic equations that represent the connecting network. This set of differential algebraic equations (DAE) have various nonlinearities and some sort of numerical method is usually used to obtain a step-by-step time domain solution. Each machine may be represented by two to twenty differential equations, and so a 2000 bus power network with 300 machines may require 3000 differential equations and 4000 algebraic equations. In terms of structure, the differential equations can be looked upon as block diagonal (one block for each machine) and the sparse algebraic equations as also providing the interconnection between the machine blocks.

This block diagonal structure has made the transient stability problem more amenable to parallel processing than the power flow problem. Research results to date seem to bear this out. Other power system analysis problems are slowly being subjected to parallel processing by various researchers. Short circuit calculations require the same kind of matrix handling as the power flow and the calculation of electromagnetic transients is mathematically similar to the transient stability solution although the models can be more complicated. Steady-state stability (or small disturbance stability) analysis requires the calculation of eigenvalues for very large matrices. The optimal

power flow (OPF) optimizes some cost function using the various limitations of the power system as inequality constraints and the power flow equations as equality constraints. Usually the OPF refers to optimization for one operating condition while unit commitment and hydro-thermal coordination requires optimization over time. This optimization problem, especially if there are many overlapping water and fuel constraints, can be extremely large even without the power flow constraints. Reliability calculations, especially when considering generation and transmission together, can be quite extensive and may require Monte Carlo techniques. Production costing is another large example.

It is the size of these above problems and the consequent solution times that encourages the search for parallel processing approaches. Even before parallel computers became a potential solution, the concept of decomposing a large problem to address the time and storage problems in sequential computers has been applied to many of these power system problems. In fact, there is a rich literature of decomposition/aggregation methods, some more successful than others, that have been specifically developed for these problems. The use of parallel computers can take advantage of these decomposition/aggregation techniques but usually a certain amount of adaptation is necessary. Much of the research in applying parallel processing to power systems has its roots in this literature. This report, however, is confined to examining the efforts that apply parallel computers to specific power system problems rather than the much larger area of methods and algorithms that are potentially applicable to parallel computers.

II Classification of Parallel Architectures

Parallel processing is a type of information processing in which two or more processors, together with an interprocessor communication system, work cooperatively on the solution of a problem [1]. This cooperative arrangement typically takes the form of *concurrent processing,* or the performance of operations simultaneously with some transfer of information between processors. Two distinct methods of performing parallel tasks have emerged:

- Pipelining
- Replication

Pipelining is the overlapping of parts of operations in time, while replication implies that more than one functional unit is applied to solving the desired problem. These two methods underlie the development of a variety of diverse computer architectures and software algorithms. Unfortunately, progress often breeds complexity. Traditional serial programs are generally transportable from machine to machine with, at most, only a nominal amount of software alteration. However, the diversity of the parallel techniques that makes parallel processing so attractive, also tends to make the software difficult to transport from machine to machine. Therefore, algorithms are usually developed for a specific architecture, rather than striving for general application. Before describing the algorithms which have been developed, or are in development at the time of this writing, it is instructive to characterize the various types of parallel hardware currently available. Most of the parallel processors which are commercially available at this time fall into one of three architectural categories: *vector computers*, which exploit pipelining, *processor arrays*, and *multiprocessors*, both of which utilize replication by containing up to a thousand or more interconnected processing units. The difference between these latter processors lies in the means by which the processing units are managed.

II.A Vector (Pipelined) Computers

Pipelining is essentially the overlapping of subtasks in time. An often used analogy to pipeline processing is that of assembly lines in an industrial plant [1]. Pipelining is achieved if the input task may be divided into a sequence of subtasks, each of which can be executed by a specialized hardware stage that operates concurrently with other stages in the pipeline. Ideally, all the stages should have equal processing speed, otherwise the stream of tasks will form a bottleneck at the point of the slowest task stage. For a linear sequence of S pipeline stages, the total transversal time for a given job will be S_τ, where τ is the time required for the completion of the slowest task stage. After the pipeline is filled (and before it is drained), the steady-state throughput will be one job per cycle. Given enough jobs of a similar nature, this will represent the average throughput, insensitive, and certainly not proportional to the number of pipeline segments [2]. Whenever a change of operation occurs, the pipeline must be emptied, reconfigured, and then refilled for the new operation. One of the disadvantages of exploiting parallelism by pipelining is that the problem must contain large amounts of identical, repetitive sets of instructions. This requirement stems from the considerable amount of overhead time required to set up, fill, and drain a functional pipeline. One type of problem which is conducive to pipelining, is one in which there exists a high percentage of vector calculations. A specific class of processors, known as vector processors, has been developed to solve these types of problems.

A vector processor is a computer that recognizes instructions involving vector variables as well as scalar variables. It is a natural step to implement vector computers as pipelined processors, since operations on vectors often involve a series of identical, repetitive tasks. It is common for applications to manipulate vectors with 50,000 or more elements [3]. Three of the most common pipelined vector processors are the Cyber-205, built by Control Data

Corporation, the Cray-1, and the Cray X-MP, both by Cray Research, where the X-MP model is an improved version of the Cray-1.

II.B Processor Arrays

Processor arrays are distinguished by several features unique to their architecture. The processor array is designed to consist of a group of processing modules led by a single centralized control unit. This architecture is especially well suited for applications in which it is desired to perform a series of repetitive calculations on various sets of data, using one set of instructions. All of the processors in an array architecture have identical hardware structure and are programmed to execute the same set of instructions on differing sets of data either in an "enabled" or "disabled" mode of operation. This allows each processing element to respond to conditional statements so that it need not participate in all instruction cycles dictated by the control unit, but otherwise the individual processing units do not have the capability of independent operation. This approach has several practical advantages. By replicating identical models, the design process is simplified because fewer overall designed are needed. The repair problems are reduced by needing only one set of diagnostic tools and replacement parts can service a larger area. Another design advantage is the simplification of control. Since the control unit treats all array modules identically, the task of controlling a large number of modules is logically the same as a small number [4]. Two well known processor arrays are the Burrough's PEPE and the Goodyear Aerospace Massively Parallel Processor (MPP).

II.C Multiprocessor Systems

In contrast to processor arrays, a multiprocessor system consists of a group of processors which are capable of independent operation, and are synchronized to work in unison on a common problem. The categorical characterization of a multiprocessor system can be described by two primary

attributes: first, a multiprocessor is a single computer that includes multiple processors, and second, processors may communicate and cooperate at different levels in solving a given problem [1]. Multiprocessor systems are usually referred to as MIMD (Multiple-Input-Multiple-Data) machines, where "input" refers to the stream of instructions given to the individual processors. Since each processor has the ability to operate independently, the input stream to each processor may be different. The most significant difference between types of multiprocessor systems is the difference in architecture of their storage, communication, and data recovery devices. The two most commonly used categories of interconnection are the *tightly coupled* and *loosely coupled* multiprocessor architectures.

In the general tightly coupled system configuration, the individual processors have access to common memory resources and exchange data among themselves via successive read and write operations performed on designated memory locations. This is commonly known as a *shared memory* multiprocessor. The rate at which data can be communicated from one processor to another is on the order of, and limited by, the bandwidth of the memory. When multiple requests to a shared memory must be served simultaneously, the memory is typically partitioned into modules in a hierarchical fashion so that simultaneous conflicting requests to a single module are rare and are handled in a predetermined manner. Access conflicts are manipulated by appropriately configuring the packet-switch paths to memory [5]. However, if there is a large amount of conflict, this conflict management procedure can result in long access delays. This problem may be partially alleviated by supplying each processor with a small individual memory, or cache, which is controlled by the processor only. Examples of shared memory systems are Sequent's Balance 8000, the Alliant FX/8, Denelcor's HEP, Carnegie-Mellon's C.mmp, and the Cray X-MP (note that this is multiprocessor comprised of a set of vector processors.)

In contrast to shared main memory, loosely coupled multiprocessors utilize a main memory which is comprised of the combination of the local

memories of the processors. When no shared memory is used, variables that are required by several processors must be accessed through a message passing system that allows one processor to request some other, possibly distant, processor to modify or transmit the requested data. As a result, this type of multiprocessor is often called a *message passing* processor. These systems do not generally encounter the degree of memory conflicts experienced by shared memory system [1], but the length of time needed to access a particular memory location will depend on the distance of the requesting processor to the memory location. A disadvantage of this type of system is that in order to move data from the memory of one processor into the memory of another processor, both the transmitting and receiving processors must be involved for the duration of the transfer. The time required to pass information among elements that are not adjacent to each other is therefore of concern in selecting a specific topology . The *hypercube* is one approach to minimizing the "maximum distance" between processing elements in large networks [6]. Other interconnection topologies include the *ring, butterfly, hypertrees*, and *hypernets*..

II.D Multicomputer Systems (Distributed Processing)

Parallel processing and distributed processing are closely related. The primary difference between these two classifications is that distributed processing is achieved by a multiple computer system with several autonomous computers which may or may not communicate with each other [1]. Each autonomous computer has its own memory, therefore there is no global memory referencing, and all communication and synchronization of processes between individual processors is achieved via message passing, or through memory shared between pairs of processors. In the most general case, each processor/computer will have its own operating system and may execute programs written in a different computer language from some or all of the other processors/computers in the distributed system.

Distributed processing is an attractive alternative to centralized computing systems fueled by the decreased cost of standalone computing facilities, improved reliability, and high computing speeds for relatively independent, parallel tasks [7]. A disadvantage to distributed processing is the difficult nature of analysis and control of these systems as compared to centrally controlled parallel processing systems.

Although there has been progress in applying vector processors to power system simulation [8], as well as processor arrays [9], [10], and distributed processing [11], the majority of research efforts have been in the application of multiprocessors for power system simulation. The remainder of this chapter will concentrate on algorithm development for these specific types of machine architectures.

III Parallel Algorithms for Power System Analysis

There are several methods for designing a parallel algorithm to solve a problem. One possibility is to determine and exploit any inherent parallelism in an existing sequential algorithm. Several compilers exist which can perform simple identifications and may work quite well for some programs. Unfortunately, there exist a significant number of sequential algorithms which have no obvious parallelism. It then becomes necessary to approach the design process from a new direction.

III.A Factors Which Impact Algorithm Efficiency

Parallelism may be exploited at four different levels:

1. job level
2. task level
3. interinstruction level
4. intrainstruction level

The last two levels require the user to have an intimate knowledge of the system hardware on which the job is to be processed, with the intrainstruction parallelism being directly exploited by the hardware measures. The job level parallelism is typically the type of parallelism utilized in distributed computing. Therefore, this chapter will address only the exploitation of task parallelism.

Two important measures used in determining how well a parallel algorithm performs when implemented on MIMD machines are speedup and efficiency. The *speedup* achieved by a parallel algorithm running on p processors is the ratio between the time taken by that parallel computer executing the fastest serial algorithm and the time taken by the same parallel computer executing the parallel algorithm using p processors. The *efficiency* of a parallel algorithm running on p processors in the speedup divided by p [3], which is a measure of the percentage of time that all processors are operating. It is desired to minimize any "idle" time, i.e., the time any of the processors are forced to wait for communication delays or synchronization. A number of factors contribute to the upper limit of speedup obtainable by a particular parallel algorithm on a specific MIMD machine. In the following sections,

several items which affect the speedup and efficiency of parallel algorithms will be discussed.

Many early parallel algorithms were proposed using the assumption that the time required for communication and memory access would be negligible compared to computation time. This assumption has been shown to be erroneous, with communication time often comprising a large percentage of the total run-time. Unfortunately, predicting communication time is probabilistic in nature and can be considered analogous to predicting when a customer will arrive at a store and how much service they will demand. In order for two or more processors to work in unison on a problem, they must be able to communicate and synchronize their combined efforts.

Synchronization refers to the control of deterministic aspects of computation. The objective of synchronization is to guarantee the correctness of parallel computations such that the results obtained from a parallel execution of a program are the same as those of a sequential execution [6]. Synchronization serves two purposes. Firstly, synchronization insures the integrity of the data being shared between processors in a shared memory system. In a system where local or cache memory is used, it is possible for caches to differ. For example, if cache A and cache B contain the same copy of the data in central memory, and processor 1 updates A, then if processor 2 accesses cache B for the same variable, it will retrieve the incorrect value. There are several schemes available for avoiding this problem. These schemes generally utilize some method of "testing" and "flagging" data to ensure that the data being retrieved is the most recent copy. In message passing systems only one processor "owns" a variable, therefore the problem of data inconsistencies is alleviated.

The second purpose of synchronization is to determine the precedence and sequentiality of a set of tasks. In almost all parallel programs there are critical sections that must be performed serially. Only one processor should be

executing the section at a time. Similarly, if all or part of a certain task A must precede task B, the processor to which task B is assigned must wait until the critical portion of task A is completed and the updated information is available to B.

Since testing and waiting is expensive in terms of system efficiency, algorithm design should strive to make the length of time between required synchronizations as large as possible. This is often achieved by making the *grain size* of the the algorithm as large as possible.

The *grain size* of a parallel algorithm is the relative number of operations done between synchronizations in a MIMD algorithm [3]. If an algorithm has a fine grain size, then the completion of tasks will often require information from other tasks [12]. It is important that an algorithm of this type be implemented on a computer that requires relatively little interprocessor time. If, on the other hand, the computer has a relatively large communication time, then it is advantageous to be able to divide, or *partition*, the problem into coarse grain tasks which require little interprocessor communication. In most situations, designing a particular algorithm for a specific machine will involve determining how to partition the algorithm into task sizes to match the communication capabilities of the machine. A mismatch in granularity and machine type may result in an unbalanced increase in communication overhead which will subsequently reduce the maximum obtainable speedup. In general, loosely coupled multiprocessors are best suited to fine grain task parallelism. Tightly coupled multiprocessors exhibit a higher communication time to access the main memory and are therefore better suited to coarse grain tasks.

The remainder of this chapter will be devoted to the study of the characteristics of various parallel algorithms which have been proposed for the solution of traditional power system problems and the implementation of these algorithms on the two main types of multiprocessors previously discussed.

III.B Parallel Processing Techniques for Power Flow Analysis

Nonlinear Equation Solution

The power flow problem might be most generally stated as a set of simultaneous algebraic equations of the form,

$$f(x) = 0, \text{ or } f_k(x_1, \ldots, x_n) = 0, \quad 1 \leq k \leq n$$

The Newton algorithm solves nonlinear equations of this form using the iteration scheme,

$$x^{k+1} = x^k - Df(x^k)^{-1} f(x^k)$$

where D is the derivative operator. Of the many methods for solving simultaneous equations Newton's method has been traditionally used for several reasons. First, generally, a sufficiently good initial estimate of the solution is known apriori so that the behavior of the power flow equations in the region encompassing the solution and the initial guess is approximately quadratic. This means that the a solution can usually be found reliably and quickly. (The problem on multiple solutions and nonexistent real solutions is of interest in general but will not be addressed here.) Second, and because of this, it is faster (and more reliable) on systems of equations which characterize utility applications (i.e., non-radial networks). Third, quasi-Newton methods are usually faster, and in certain cases more robust. Quasi - Newton algorithms deal with modifications to the above iteration schemes in the following general way

$$x^{k+1} = x^k - (A^k)^{-1} f(x^k)$$

where A^k are approximations to the Jacobian matrices $Df(x^k)$. These algorithms are usually implemented via the update rules,