Parallel and Distributed Processing Applications in Power System Simulation and Control

Djalma M. Falcão

COPPE*
Universidade Federal do Rio de Janeiro

Abstract

Recent advances in computer technology will certainly have a great impact on the methodologies used in power system expansion and operational planning as well as in real-time control. Parallel and distributed processing are among the new technologies that present great potential for application in these areas. Parallel computers use multiple functional or processing units to speed up computation while distributed processing computer systems are collections of computers joined together by high speed communication networks having many objectives and advantages. This paper discusses the potential application areas of parallel and distributed processing in power system and presents an overview of the current research effort in this subject. The paper also presents comments regarding the research work in this area presently in developmewnt in Brazilian institutions and parallel computation facilities available in Brazil.

1 Introduction

Power system planning and operation in the 1990s face new challenges imposed by a new economic context (deregulation, privatization, etc.), the practical utilization of fast acting electronic control devices (FACT's and HVDC controls), and the stretching of the electric performance by operation close to system limits. Computation tools to simulate and control power systems with these characteristics must be powerful enough to deal with increasingly complex models and solution possibilities overlooked in the past. Moreover, the reduced number of power engineers graduated in the past decade indicates a likely shortage of power system experts in the near future. To overcome this personnel deficiency, the few available experts will need efficient computation tools to properly perform their duties.

Power system computation methods for simulation and control have undergone a great deal of development in the last three decades or so. Techniques for handling large sparse matrices, clever adaptations like the fast decoupled power flow method, etc., made possible the analysis of large scale power system models. A widely accepted guess is that the introduction of new mathematical or programming methodologies will produce only marginal increase in the performance of these methods [1]. Moreover, single processing capacity, although dramatically increased in the last years, seems to be approaching physical limits [2, 3, 4]. Therefore, substantial improvements are more likely to come from the adaptation of the already well developed power system computation methodologies to the new hardware and software developments being offered by the computer industry. Parallel and distributed processing appear to be among the most promising ones of these new developments [5]-[7]. Parallel processing consists in the use of multiple hardware components to exploit concurrency in the computation job. The main advantage of parallel processing in power system applications is the speed up of computations in order to make viable the solution of problems intractable in conventional computers. Distributed computer systems are collections of independent computers joined together by high speed communication networks having many objectives and advantages. Power system applications can benefit from this form of decentralized computer architecture due to its geographically distributed nature and, as in many other application areas, from its flexibility, scalability, cost advantage, etc.

This paper introduces and discusses some ideas for the use of parallel and distributed processing in power system simulation and control. Initially, it is presented a general description of parallel and distributed processing schemes, their advantages and disadvantages, as well as promising power system application areas. Next, an overview of the research work currently in development in these topics is presented together with a perspective view of future developments in this area. Finally, the paper presents comments regarding the research work presently in development in Brazilian institutions and parallel computation facilities available in Brazil.

*The author can be reached at COPPE/UFRJ, Caixa Postal 68504, 21945-970 Rio de Janeiro RJ, Brazil, Phone: (021) 260 5010, Fax: (021) 290 6626, E-mail. falcao@coep.ufrj.br.

2 Power System Simulation and Control

Simulation of power system behavior in the presence of several types of predictable disturbances is a fundamental tool in expansion and operational planning and real-time control of these systems. Owing to the complexity of the phenomena involved in power system operation, simulation studies are usually performed within different time frames: starting with the very fast electromagnetic transients caused by switching operations and lightning, passing through the slower electromechanical oscillations resulting from power unbalance on rotating machines, and eventually reaching the steady-state analysis corresponding to different loading conditions during the daily, weekly or yearly load cycle. Different models are used to represent power system components in these simulations but some basic characteristics appear in all of them. For instance, the dynamic interaction between apparatus (generators, compensators, etc.) occur via the transmission network whose representation introduces some similar structural properties into the models [8, 9, 10]. These models usually consist of sets of non-linear ordinary differential equations, presenting direct interaction only to other few equations, and/or sets of algebraic equations whose linearized approximations exhibit similar sparsity patterns.

The power system simulation problems, when compared to simulation problems found in other fields of engineering, may be classified as medium size problems. However, simulation cases are seldom performed only once. In general, practical expansion and operational planning applications require hundreds or thousands of simulation cases. In some conjectured probabilistic assessment studies, this figure may reach several thousands. Other stringent requirement of the power system simulation problem is in the case of real-time simulators: unfolding the power system's response in the same rate the physical system would do. These requirements make power system simulation a very time consuming and/or numerically intensive computer application. To circumvent the lack of enough computer power to satisfy these high requirements, the usual practice so far has been to sacrifice accuracy by the introduction of simplifying assumptions and approximated models. This procedure sometimes leads to meaningless results very often misunderstood by engineers not well acquainted with the limitations of the used models. Dedicated analog or hybrid devices (TNA's and HVDC simulators) have been used for the simulation of electromagnetic transients but they are expensive and less flexible than fully digital simulators.

The real-time control environment, apart from specific computation tools, also rely extensively on power system simulation. The preventive control strategy, by and large adopted in modern Energy Management Systems (EMS), tries to avoid catastrophic occurrences by a periodic simulation of credible contingencies and system rescheduling and reconfiguration. Obviously, in this case a fast solution time is essential to allow corrective measures to be taken in due time. The state-of-the-art in real-time control hardware and software is so far only able to deal with steady-state models. Important dynamic phenomena, like transient, small signal, and voltage stability, are almost completely ignored due to the lack of powerful enough computation resources. Probabilistic analysis is also usually left out of the real-time control strategy for the same reason.

3 Parallel and Distributed Processing

Parallel and distributed processing are similar but rely on different information processing concepts. Parallel computers are usually single frame units in which the idea of concurrent processing is exploited mainly to speed up computations. Distributed computer systems are collections of computers joined together with multiple objectives depending on the envisaged applications. These data processing approaches have achieved practical implementation mainly due to the relative large increase in microprocessor performance as can be seen in Figure 1 [4] and Table 1.

3.1 Parallel Processing

Parallel processing can be broadly defined as the utilization of multiple hardware components to perform a computation job. Several levels of parallelism can be found in present day computers but the ones most important for the applications discussed in this paper are multiprocessors and multicomputers [3].

A *multiprocessor* is a computer composed of several processors sharing a common memory. A *multicomputer* is a computer composed of several processor-memory pairs which communicate between themselves by message passing. Figure 2

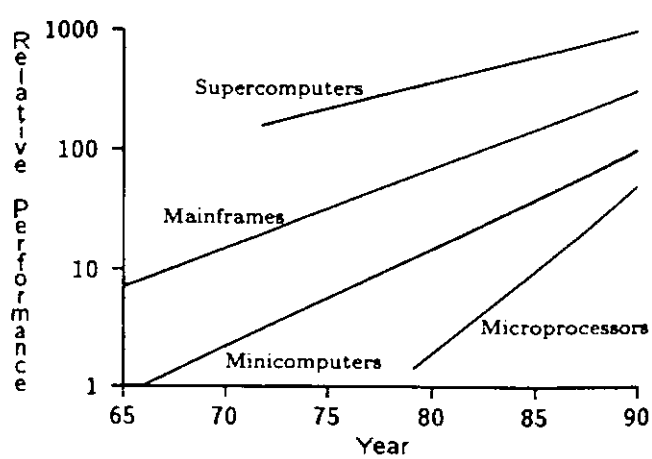

Fig. 1: Trends in computers and microprocessors performance

Table 1: Microprocessors used in microcomputers, workstations and parallel computers

Micropro-cessors	Manufac-turer	Type	Clock (MHz)	Operating systems	Peak performance (MIPS/MFLOPS)
Alpha 21064	DEC	RISC	200	WNT,OSF1,VMS	
MIPS R4400SC	MIPS	RISC	150	IRIX,SRVR45	
PA7100	HP	RISC	100	HP-UX	
Pentium	Intel	CISC	66	WNT,OS2,Unix,etc.	
PowerPC 601	IBM/Motorola	RISC	80	OS2,AIX,MacOS	
SuperSparc	SUN/Texas	RISC	60	Unix SVR4, Sun OS	
i860	Intel	RISC	66	WNT,OS2,Unix,etc.	
Transputer	Inmos				

presents a schematic view of these two forms of parallel computer architecture. Other interesting forms of parallelism are the pipelined processors, used in *vector supercomputers*, specially tailored for concurrent operation of loops present in vector and matrix operations, and the *superscalar* processors that uses special compilers to optimize the execution of conventional programs by executing concurrently some instructions in a single instruction stream.

Multicomputers, and less extensively multiprocessors, can be built using commodity hardware in contrast to, for instance, vector supercomputers that require specially designed processors and memories. In particular, the microprocessors used in microcomputers and workstations are becoming standard components of commercially available parallel computers as can be seen in Table 2. For this reason, multiprocessors are able to present a price/performance advantage over supercomputers based on one or a few customized processors. This fact is driving the high performance computing industry towards *massive parallelism*, i.e., computers with hundreds or thousands of processors. Scalability is another interesting feature of parallel computers. Table 2 presents a sample of presently commercially available parallel computers..

Programs developed for conventional computers usually require relatively extensive modifications to be used in multiprocessors and multicomputers. Moreover, the best performance of these machines are usually achieved only with specially developed algorithms and programs. Automatic program parallelization tools are rare in the massive parallel environment and it is doubted weather they will be commonly available in the foreseeable future. Fortunately, a strong research effort in universities and industry around the world in the last years made available a large number of algorithms suitable for parallel implementation [6]. Moreover, parallel computer manufacturers are at present offering development tools much more efficient than they were a few years ago. On the other hand, it is usually easier to convert programs to run on supercomputers and superscalar processors. Automatic vectorization compilers are often available for supercomputers although the best performance are still achieved only by careful hand customization.

The gain obtained in moving an application to a parallel computer has been usually measured in terms of speedup and efficiency of the parallel implementation when compared to the *best* sequential version. *Speedup* is usually defined as the ratio between the execution time of the best sequential code in one processor of the parallel machine to the time to run the parallel code in P processors. *Efficiency* of the parallelization process is defined as the ratio between the speedup achieved on P processors to P. In the early stages of applications development for parallel computers these two indexes were almost exclusively the determinants of the quality of parallel algorithms. As more parallel machines became commercially available, and practical applications begin to be actually implemented, other aspects of the problem started to become important. For instance, the cost/performance ratio (Mflops/$; Mflops=$10^6$ floating point operations) attainable in real-life applications. In other cases, although speedup and efficiencies are not so high, the implementation in a parallel machine is the only way to achieve the required speed in the computations.

3.2 Distributed Processing

A distributed computing system is a collection of autonomous computers, interconnected by a communication network, that work together to satisfy the information processing needs of a company, department, laboratory, etc. [6, 7]. Figure 3 presents a generic view of a distributed computing system. This type of decentralized computing approach is feasible today due to the availability of very powerful desktop computers (microcomputers and workstations) which can be interconnected by very fast networks. Distributed systems are replacing mainframe based computer systems in both commercial and scientific applications generating a phenomena known as *downsizing* of applications.

Distributed computing systems may include a few computers located in a single room, connected by a

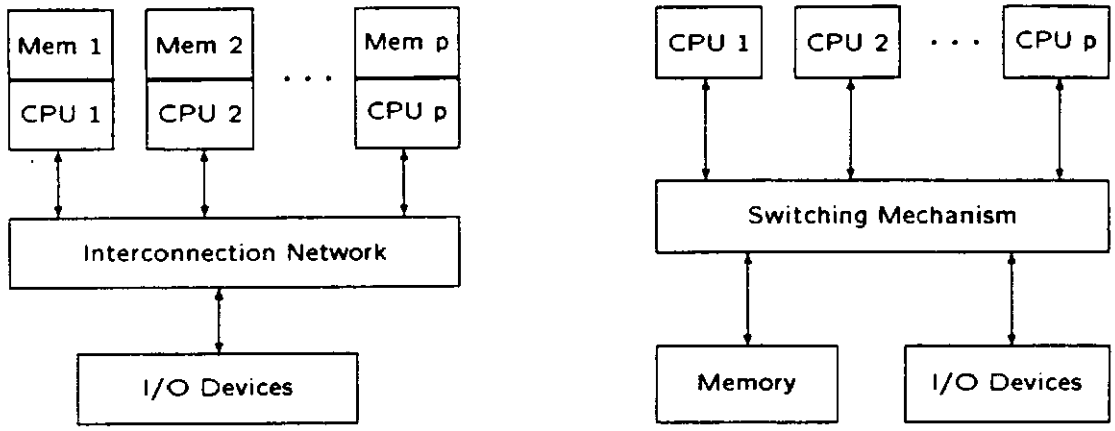

Fig. 2: Parallel Computer Architectures

local area network using coaxial cable as interconnection media, or a large number of computers scattered hundred of kilometers apart, interconnected by a wide area network based on telephone lines, microwave channels, optical fiber cables, etc. The computers may be of different types and capacities. No single operational system need to be shared between processors.

Some of the advantages of distributed computing systems over conventional ones are [7]:

- *Cost*: the average cost for MIP (million instructions per second) on a mainframe is almost 100 times the one for a workstation.
- *Scalability*: distributed systems can be expanded incrementally.
- *Flexibility and Configurability*: distributed systems have improved performance and reliability due to redundant data and processing.
- *Exploitation of special hardware*: special graphic devices, highly efficient number crunching modules (vector or parallel units), etc., can be added easily to the system.

On the other hand, distributed processing is a new and unmarked field. The lack of standards, security and integrity control difficulties, the availability of too many options, etc., raise several technical and management concerns.

Distributed applications can be developed according to several different paradigms [7]. In the simplest one, a *terminal emulator* makes a computer look like a terminal which is connected to another computer. A *file transfer package* allows files to be transferred between different computers. A more elaborate model is the *client-server* system in which remotely located programs can exchange information in real-time. A *distributed data and transaction management system* is a sophisticated version of the client-server system which allows a user to store, access and manipulate data transparently from many computers. This type of system will eventually evolve to a truly distributed operating system so that the entire network appears to the user as a large computer system.

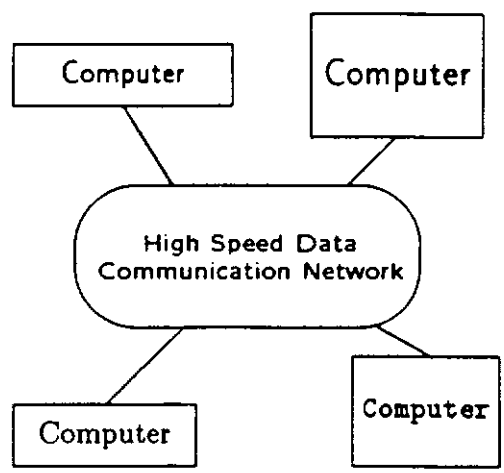

Fig. 3: Distributed processing

In the last few years, *workstation clusters* have been gaining acceptance as a platform to implement parallel computing in distributed systems. These are groups of workstations connected by a local area network which can exchange information by message passing. Parallel execution of programs can be achieved using client-server programming languages and tools (Remote Procedure Call-RPC, for instance) or specially developed tools like PVM (Parallel Virtual Machine). The communication speed between workstations is usually low what makes these systems indicated only for applications in which computation on the processors are much more intense than communication between processors. It is claimed that these systems may be able to use the idle time of a cluster of workstations to perform highly intensive computer jobs in the background.

Table 2: Commercially available parallel computers.

Name	Manu-facturer	Classi-fication	Proces-sor	Topology	No. of Proc.	Peak Performance (GFLOPS)*
Paragon	Intel	MIND	i860 XP	Mesh 2d	8-480	36
nCube 2	Ncube	MIMD	Custom	Hipercube	2-1024	2.5
CM-5	Tinking Machine	MIMD	Sun Sparc	Fat Tree	1-16000	1000
SP1	IBM	MIMD	PowerRisc		1-64	8
T3D	Cray	MIMD SIMD	Alpha 21064	Torus 3d	32-2048	307
Exemplar SMM	Convex	MIMD	PA7100	Tightly Coupled Crossbar	1-128	25

4 The Impact of Parallel and Distributed Processing in Power System Applications

From the point of view of potential for parallel processing, power system applications can be classified as:

- *Obviously Parallelizable:* applications that consist in the solution of almost independent and computationally intensive numerical problems as static contingency analysis, Monte Carlo simulations, eigenvalue analysis, security constrained optimal power flow, etc.

- *Not Obviously Parallelizable:* applications that require ingenious problem decomposition in order to achieve a reasonable degree of concurrency in the solution algorithms. Most of the applications in this category require the parallel solution of large sets of sparse linear algebraic equations originated from different power network modeling approaches. Examples of this class of applications are the power flow computations, dynamic simulation for transient and long term stability analysis, etc.

Some applications, like electromagnetic transients computation and state estimation, although conceptually belonging to the second category, may be included in the first one due to a nice structure of the equation set to be solved as will be discussed later in this paper. It should be noted that the border between the two categories above is nebulous and likely to be changing with advancements in parallel processing methods and architectures.

The brief history of parallel processing application to power system problems registers an apparent paradox: although several important obviously parallelizable applications are available, most of the research effort has been concentrated on applications in the not obviously parallelizable category. An explanation for this phenomenon may be found in the fact that most of the research in this field has been conducted in universities, were parallel machines have been available for quite a long time. Applications with not obvious parallelism present more interesting challenges from both the theoretical and algorithm design points of view and, therefore, are more attractive to academic researchers. More recently, this trend has been changing, even among academics, as the obviously parallelizable applications have shown to present as big challenges as the other category although seldom in the numerical algorithmic level.

Major impacts of parallel processing in practical terms are expected to occur in areas in which conventional computer methods and architectures have failed so far to produce adequate results or in areas that are likely to increase their complexity due to the need of modeling newly introduced equipment. Some candidates are:

- *Real-Time Control:* dynamic security assessment and correction, voltage stability assessment, security constrained optimal power flow, etc. [11, 12].

- *Real-Time Simulators:* electromechanical and electromagnetic transient simulators to be used in design and test of new equipment, control and protection schemes, operators training, etc. [13]-[15].

- *Probabilistic Assessment:* composite reliability and other probabilistic techniques based on Monte Carlo and enumeration methods using realistic static and dynamic models [16]-[18].

Another field that can benefit from parallel processing is the automation of power engineering analysis and synthesis studies required both in expansion and operational planning. These studies require long and tedious cycles of case preparation, runs, and analysis of load flow, transient stability, short-circuit, etc. As the availability of qualified engineers to perform these studies is diminishing, a

result of both the utilities personnel policies and reduced interest of electrical engineering students for the power engineering area, the development of automated tools to help engineers to speed up the power engineering studies is becoming an important requirement. This type of application exhibits a large amount of obvious parallel computations (each power flow or transient stability case is almost completely independent from each other) but requires a central coordination tool most likely based on *intelligent processing techniques* like expert systems, neural networks, fuzzy logic, etc.

Two distinct reasons motivate the utilization of distributed processing in power system applications: the geographically distributed nature of the power system and the recognition by the electrical energy industry of the cost and operational advantages of using networks of microcomputers and/or workstations, based on the open system concept, instead of the expensive and less flexible proprietary mainframe based systems.

Two applications are already established as viable and being implemented in practice:

- *New Architectures for EMS:* the traditional dual and similar proprietary architectures of control centers, using mainframe or supermini computers, are being substituted by more versatile and cost effective networks of high performance workstations [5, 19, 20].

- *Distribution Automation:* distribution automation projects have been proposed based on the availability of processing power distributed alongside the power distribution network with the objective of supervised switching operations, voltage/var control, load management, customer interface automation functions, etc. [5, 21, 22].

New areas which may benefit from distributed processing are [23]:

- *Distributed EMS:* integration of the traditional supervision and control system of the main transmission network with the distribution automation function and down to the customer level.

- *Corporation Integrated Information System:* integration of real-time operations with planning functions and other corporation functions, such as accounting, customer services, and management information systems.

5 Power System Applications

This section introduces a review of parallel and distributed processing techniques applied to power system problems. This review contains mainly the work developed by the author and his associates. Research work developed by other groups are also mentioned and reviewed to some extent.

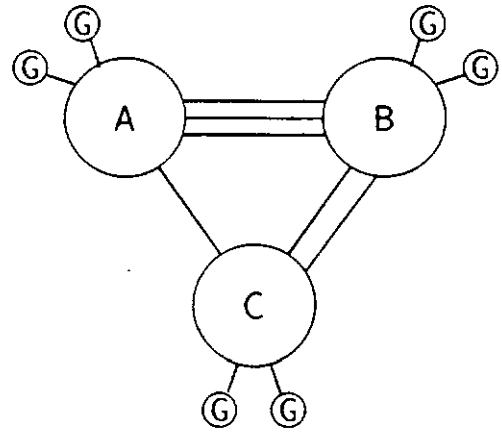

Fig. 4: Dynamic Simulation Model Decomposition

5.1 Dynamic Simulation for Transient Stability Analysis

The mathematical model usually adopted for transient stability analysis consists of a set of ordinary non-linear differential equations, associated to the synchronous machine rotors and their controllers, and a set of non-linear algebraic equations associated to the transmission network, synchronous machine stators, and loads [9]. These equations can be expressed as:

$$\dot{x} = f(x, z) \quad (1)$$

$$0 = g(x, z) \quad (2)$$

where f and g are non-linear vector functions; x is the vector of state variables; and z is the vector the algebraic equations variables.

In the model defined in (1) and (2), the differential equations representing one machine present interaction with the equations representing other machines only via the network equations variables. From a structural point of view, this model can be visualized as shown in Figure 4: clusters of generators are connected by local transmission and subtransmission networks and interconnected among themselves and to load centers by tie-lines.

In the sequential computer context, several solution schemes have been used to solve the dynamic simulation problem. The main differences between these schemes are in the numerical integration approach (implicit or explicit) and the strategy to solve the differential and algebraic set of equations (simultaneous or alternating). Implicit integration methods, particularly the trapezoidal rule, have been mostly adopted for this application. The most used schemes are [9]:

1) Alternating Implicit Scheme (AIS): this solution scheme is better understood if (1) and (2) are rewritten as:

$$\dot{x} = Ax + Bu \quad (3)$$

$$I(E, V) = YV \quad (4)$$

$$u = h(E, V) \quad (5)$$

where A is a square, real valued, block diagonal matrix in which each block is associated to one machine; B is a rectangular, real valued, blocked matrix in which each block is associated to one machine; u is a vector of interface variables; I is a vector of injected nodal currents; Y is a square, complex valued, nodal admittance matrix; V is a vector of complex nodal voltages representing the steady-state network behavior; E is a subvector of x required to calculate current injections in generation nodes; and h is a nonlinear vector function.

The alternating implicit scheme consists in transforming the differential equations in difference equations, by the application of an implicit integration method, and to solve these equations iteratively and alternately with the algebraic equations. A modified version of this scheme, called the Interlaced Alternating Implicit scheme [9, 31, 32], is obtained relaxing the convergence requirements on the network equation solution. The convergence test is performed in the state variables. This method usually presents better results than the conventional approach.

2) Simultaneous Implicit Scheme (SIS): in this approach, the network algebraic equations and the algebrized differential equations

$$F(x, V^e) = 0 \quad (6)$$
$$G(x, V^e) = 0 \quad (7)$$

where F is a vector function associated with the difference algebraic equations, G is a vector function associated with the algebraic equations, and V^e is a vector of nodal voltages expanded in its real and imaginary components, are lumped together in an enlarged set of equations which are, then, solved simultaneously. The Newton method is the usual choice of method to solve this set of equations. In the $k-th$ iteration of the Newton algorithm, the following set of linear equations has to be solved:

$$\begin{bmatrix} F(x, V^e) \\ G(x, V^e) \end{bmatrix}^k = - \begin{bmatrix} Q & R \\ S & \bar{Y} \end{bmatrix}^k \begin{bmatrix} \Delta x \\ \Delta V^e \end{bmatrix}^{k+1} \quad (8)$$

where $Q, R, S,$ and $\bar{Y}$ are Jacobian submatrices, The Jacobian matrix in (6) can be ordered in a Block Bordered Diagonal Form (BBDF) [25, 26] and the system of equations solved by Block Gauss elimination.. The elements of the Jacobian matrix may be kept constant for several iterations and/or integration steps in a process known as the Very Dishonest Newton Method (VDHN).

The difficulties for the parallelization of the dynamic simulation problem in the AIS are concentrated on the network solution. The differential equations associated with the synchronous machines and their controllers are naturally decoupled and easily parallelizable. On the other hand, the network equations constitute a *tightly coupled* problem requiring ingenious decomposition schemes and solution methods suitable for parallel applications. The SIS also requires the parallel solution of sets of linear algebraic equations in every integration step, with difficulties similar to the ones described for the AIS.

A basic numerical problem in both simulation schemes is the parallel solution of sets of linear algebraic equations. Direct methods, like LU factorization, have been dominating this application in conventional computers. If parallel computers are considered, however, the hegemony of direct methods is no more guaranteed. In several other engineering and scientific fields, parallel implementations of iterative methods have shown superior performance. Among the most successful iterative methods are the ones belonging to the Conjugate Gradient (CG) category [6]. The parallelization of the solution of the network equations requires the decomposition of the set of equations in a number of subsets equal to the number of processors used in the simulation. An adequate decomposition is fundamental to the success of the parallel solution and need to take into consideration factors like computation load balancing, convergence rate of the iterative algorithms, etc.

In the last decade or so, several parallel methods were proposed for the solution of the dynamic simulation problem. In the following sections, some of these methods are reviewed.

5.1.1 Spatial Parallelization

Methods in this category exploit the structural properties of the the network or Jacobian equations to be solved in each integration step of conventional simulation schemes (AIS or SIS). Four methods are briefly described below:

1) The Parallel VDHN [27]: it consists in a straightforward parallelization of the VDHN method simply identifying tasks that can be performed concurrently and allocating them among the processors. This method was implemented in the parallel computers Intel iPSC/2 (distributed memory) and Alliant FX/8 (shared memory) and tests performed with the IEEE 118 bus and US Midwestern system with 662 buses. The results show speedups slightly superior for the iPSC/2 with a strong saturation with the increase in the number of processors. The maximum obtained speedup was 5.61 for 32 processors (efficiency = 17.5%).

2) The Parallel Newton-W matrix Method [28]: the main feature of this approach is the use o a parallel version of the Sparse Matrix Inverse Factors [29, 30] in the SIS. The methodology was tested in the shared memory Symmetry parallel computer the same test system used in the work cited in the previous item. The results show a worse performance of this method when compared to the parallel VHDN with an slowdown of 10% to 30% depending on the chosen partitions.

3) The Parallel Real-Time Digital Simulator [14]: The main objective of this work was the development of a real-time power system simulator for equipment testing based on a massive parallel architecture. The parallel algorithm is based on the AIS using the trapezoidal integration method. One processor is associated to each network bus. The differential equations corresponding to each generator and its controllers are solved in the processor assigned to the bus in which the generator is connected. The network equations are solved by a Gauss-Seidel like method also allocating one equation to each processor. Therefore, the number of processors required to perform the simulation is equal to the number of buses in the network. Reported results with a 261 bus network in a 512 node nCube parallel computer show that the required cpu time is not affected by the system dimensions. However, it is doubtful weather this property can be kept valid for larger system taking into consideration that the number of iterations required by the Gauss-Seidel algorithm increases considerably with system size. This approach exhibits low speedup and efficiency measured by the traditional indexes. However, it has had a tremendous impact in the research community because it has demonstrated the usefulness of parallel processing in solving a real world problem.

4) The Hybrid CG-LU Approach [31, 32]: This method is based on the AIS, the decomposition of the network equations in a BBDF, and a hybrid solution scheme using LU decomposition and the CG method. An efficient parallel implementation of the CG method depends on the existence of a convenient structure of the coefficient matrix. One such structure is the BBDF which is shown in (9) for equation (4).

$$\begin{bmatrix} I_1(E_1,V_1) \\ I_2(E_2,V_2) \\ \vdots \\ I_p(E_p,V_p) \\ I_s(E_s,V_s) \end{bmatrix} = \begin{bmatrix} Y_1 & & & & \bar{Y}_1 \\ & Y_2 & & & \bar{Y}_2 \\ & & \ddots & & \vdots \\ & & & Y_p & \bar{Y}_p \\ \bar{Y}_{1_s}^t & \bar{Y}_2^t & \dots & \bar{Y}_p^t & Y_s \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_p \\ V_s \end{bmatrix} \quad (9)$$

The BBDF of the network equations can be achieved by re-ordering the network nodes in $p+1$ sets in which the p first sets correspond to subnetworks connected to each other by the *boundary buses* in the $(p+1)^{th}$ set. A semiautomatic decomposition method to obtain the BBDF for large networks is described in the next section of this paper.

Equation (9) can be solved in a two-phase scheme by Block-Gaussian Elimination as follows:

Phase 1: Solve

$$\hat{Y}_s V_s = \hat{I}_s \quad (10)$$

where:

$$\hat{Y}_s = Y_s - \sum_{i=1}^{p} \bar{Y}_i^t Y_i^{-1} \bar{Y}_i \quad (11)$$

$$\hat{I}_s = I_s - \sum_{i=1}^{p} \bar{Y}_i^t Y_i^{-1} I_i \quad (12)$$

Phase 2: For $i = 1, 2, \dots, p$, solve

$$Y_i V_i = I_i - \bar{Y}_i V_s \quad (13)$$

The solution of the network equations in the form (10-13) can be performed in different ways. Considering the implementation in parallel machines, the main issue is how to solve *Phase 1* efficiently since *Phase 2* presents an inherently parallel structure. The explicit formation of $\hat{Y}_s$ and $\hat{I}_s$ and the solution of (10) by a direct method could be a straightforward approach to this problem. However, it should be noted that $\hat{Y}_s$ usually presents a less sparse structure than Y_s. Moreover, as stated before, the parallel implementation of direct methods is not an easy task.

The solution of (10) by the CGM does not require the explicit formation of $\hat{Y}_s$ but only of $\hat{Y}_s V_s^0$ and $\hat{I}_s$ in the calculation of the residual r^0 and $\hat{Y}_s d^k$ ($d^k = 0, 1, \dots,$ are the conjugate directions) in each iteration. Moreover if the p independent systems given in (13) are solved by a direct method like LU factorization, the factors of Y_i, $i = 1, \dots, p$, can be efficiently used to calculate $\hat{Y}_s V_s^0$, $\hat{I}_s$ and $\hat{Y}_s d^k$ in parallel.

5) The Full CG Approach [33]: In this method the network equations are solved as a whole by a block-parallel version of the Preconditioned CG method. The network matrix is decomposed in such a way that the blocks in the diagonal are weakly coupled to each other, i.e., in a Near Block Diagonal Form (NBDF) [25, 26]. The NBDF is equivalent to the decomposition of the network in subnetworks weakly coupled. A block-diagonal matrix, obtained from the NBDF neglecting the off-diagonal blocks, is used as a preconditioner. To optimize the algorithm performance, the NBDF should be obtained according to the following:

- The number of subnetworks is fixed and equal to the number of processors used in the simulation.
- The number of subnetwork connections is kept to a minimum in order to reduce communication requirements.
- The subnetworks have approximately the same dimensions and complexity.

Two approaches were used to obtain the NBDF's with the above characteristics: a trial and error method based on a careful analysis of the system

one-line diagram and the semiautomatic methodology described in the next section.

The Hybrid CG-LU method has shown a fairly good performance on actual parallel implementations but presents the disadvantages of applying the CG method to a system of equations with relatively small dimensions and the need of an optimized BBDF. The Full CG method an attempt to overcome these difficulties. The CG gradient method is supposed to present best performance, both in parallel and vector computers, for very large problems.

The use of preconditioning techniques is advised for a better performance of the Hybrid CG-LU method and essential for an adequate performance of Full CG method. Truncated Series preconditioning [40] is used for Hybrid CG-LU method. For the Full CG method, the Incomplete LU Factorization [6] was chosen as the preconditioning approach after preliminary tests with several methods available in the literature.

5.1.2 Waveform Relaxation

This method [35, 36] consists in the decomposition of the set of equations describing the power system dynamics into subsystems weakly coupled and to solve each subsystem independently for several integration steps to get a first approximation of the time response. The results are, then, exchanged and the process repeated. The advantages of this method are the possibility of using different integration steps for each subsystem (multirate integration) and to avoid of the need to solve large sets of linear algebraic equations.

5.1.3 Space and Time Parallelization

This class of methods follows the idea introduced in [37] in which the differential equations given in (1) are algebrized for several integration steps, called integration windows, and solved together with the algebraic equations of this window by the Newton method, similarly to the conventional one-step approach shown in (56). The resulting enlarged system of linear equations has the following structure:

$$\xi = -\Lambda\delta \qquad (14)$$

where

$$\xi = \begin{bmatrix} \hat{F}_1 & \hat{G}_1 & \hat{F}_2 & \hat{G}_2 & \cdots & \hat{F}_p & \hat{G}_p \end{bmatrix}^T$$

$$\delta = \begin{bmatrix} \Delta\hat{x}_1 & \Delta\hat{V}_1^e & \Delta\hat{x}_2 & \Delta\hat{V}_2^e & \cdots & \Delta\hat{x}_p & \Delta\hat{V}_p^e \end{bmatrix}^T$$

$$\Lambda = \begin{bmatrix} Q_1 & R_1 & & & & & \\ S_1 & Y_1 & & & & & \\ Q_{21} & R_{21} & Q_2 & R_2 & & & \\ 0 & 0 & S_2 & Y_2 & & & \\ & & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & \\ & & & & Q_{pp-1} & R_{pp-1} & Q_p & R_p \\ & & & & 0 & 0 & S_p & Y_p \end{bmatrix}$$

where p is the number of integration steps taken simultaneously. Matrix Λ is asymmetric and very sparse. Assuming that each one of the m synchronous machines and its controllers is modeled by κ differential equations and that the network has n nodes, the dimension of (7) is $(\kappa m + 2n)$.

1) The Space and Time CG Approach [33]: The calculation of the elements of the mismatch functions $\hat{F}_i$ and $\hat{G}_i$, the elements of the Jacobian matrix, and the updating of the components of $\hat{x}_i$ and $\hat{V}_i^e$ in (14) can be performed with perfect parallelism. Therefore, an obvious mapping of these operations in the multiprocessor system is to allocate the equations corresponding to each integration step to a single processor. The decomposition of (14) should be done in the same way. The advantages of this decomposition approach are:

- Perfect processing load balance.
- Low communication requirements. Except for the convergence test, each processor needs to communicate only with processors assigned to integration steps before and after itself.
- Each processor may be a cluster of q other processors which could be used to solve each subset of equations in parallel in the so called space and time parallelization.

Two CG methods suitable for asymmetric sets of linear equations were experimented in the implementation of the this method: These implementations are referred to as:

- The Bi-Conjugate Gradient method (Bi-CG) in which two sequences of mutually orthogonal residuals are generated by simple relations similar to the one used in the CG method [41].
- the recently introduced more robust and efficient variant of the Bi-CG method called the Bi-CGSTAB [42].

Both methods were implemented using preconditioning. The preconditioning matrix used is the following block-diagonal matrix:

$$M = -\,\text{blockdiag}\begin{bmatrix} Q_1 & Y_1 & Q_2 & Y_2 & \cdots & Q_p & Y_p \end{bmatrix}$$

2) The Space and Time Gauss-Jacobi-Block-Newton Approach [38, 39]: In this approach the waveform relaxation concept is used with a slightly different formulation. The discretization of the differential equations is performed for all integration steps simultaneously resulting in an extremely large set of algebraic equations. In a first work [38], this set of equations is solved by a parallel version of the Gauss-Jacobi method with a poor performance. In a second work [39] a method called Gauss-Jacobi-Block-Newton Approach that consists essentially in

the of the VDHN method to the equations associated to each integration step and, then, to apply the Gauss-Jacobi globally to all integration steps, is used with better results. Both works present results only for simulations of parallel implementation.

5.1.4 Conjugate Gradient Approach Results

The Hybrid CG-LU, Full CG, and Space and Time CG methods described above were tested using different test systems, including a representation of the South-Southern Brazilian interconnected system with 80 machines and 616 buses. The tests were performed on the iPSC/860 computer and in a prototype parallel computer using the Transputer T800 processor. Despite the difficulties in parallelizing this application, the results obtained in these tests showed a considerable reduction in computation time. The CG methods presented adequate robustness, accuracy, and computation speed establishing themselves firmly as an alternative to direct methods in parallel dynamic simulation. Moderate efficiencies and speedups were achieved, particularly in the tests performed on the iPSC/860, which are partially explained by the relatively low communication/computation speed ratio of the machines used in the tests. It is believed that in other commercially available parallel machines, the studied algorithms will be able to achieve higher levels of speedup and efficiency.

5.2 Network Decomposition for Parallel Block-Iterative Solutions

A basic step in the solution of sets of linear equations by block-iterative methods is the decomposition of the coefficient matrix in order to have it mapped onto the parallel architecture. In the case of electrical networks, this is equivalent to decompose or *tear* apart the network into subnetworks of smaller size.

Reference [24] introduces an enhanced version of a semiautomatic network decomposition method for block-iterative solutions on parallel computers preliminary reported in [34]. The basic operating principle of the proposed method is *the build up of subnetworks from a given number of starting nodes (seed nodes)*. The subnetworks are formed by aggregating nodes one by one to these seed nodes. The choice of which node to aggregate to a given seed node depends on a node ranking criterion based on weights assigned previously to all network nodes. Fig. 5 shows an overview of the decomposition method.

The weights used for the node ranking are calculated using the following criterion:

$$w_i = \sum_{l \in \Omega_i} B_l^{\frac{B_l}{M}} \quad , \quad i = 1, ..., n_b \qquad (15)$$

$$M = \sum_{l \in \Omega_T} \frac{B_l}{b} \qquad (16)$$

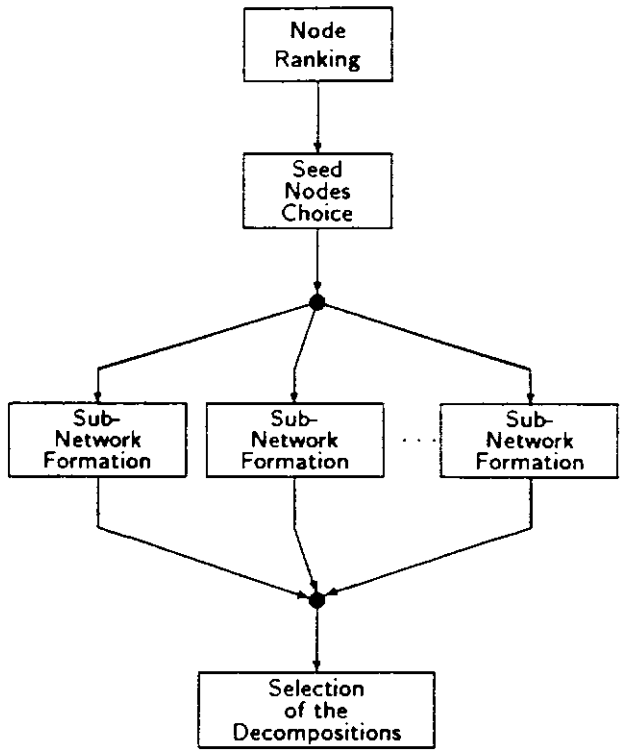

Fig. 5: Decomposition method overview

where Ω_i is the set of all branches connected to node i; Ω_T is the set of all network branches; B_l is the susceptance of branch l; n_b is the number of network branches. The value of w_i indicates the magnitude of the electrical interconnection between a node and its neighbors. The exponent $\frac{B_l}{M}$ is used to increase the influence of branches with high susceptance values in order to differentiate between nodes with several low susceptance value interconnections from nodes with few interconnections with high susceptance value.

In each step of the process, the next node to be aggregated to the subnetwork is the one with the largest weight and still not yet aggregated to other subnetworks. The process terminates when the subnetworks can aggregate no more nodes. In the algorithm to generate the NBDF, all nodes are aggregated to the subnetworks while in the one for the BBDF, the nodes left out of the subnetworks form the interconnection subnetwork.

The choice of the seed nodes set is a relevant step in the decomposition method outlined above. The number of seed nodes determines the number of subnetworks and their location strongly influences the suitability of the decompositions for parallel solution. Most of the generated decompositions are likely to be adequate for a solution by a block-iterative method. However, some of these decompositions require less iterations and computation time than the others, for the same solution method and convergence tolerance. Therefore, it is desirable to be able to spot the most promising decompositions before the effective network equation solution. This can be achieved using a Decomposition Selection

method.

The decomposition method outlined above, using a node ranking based on weights calculated according with (15) and (16), is conducive to block-diagonal dominance of the NBDF's and BBDF's and, therefore, to the convergence of associated iterative processes for the network equation solutions.

5.3 Simulation of Electromagnetic Transients

In the usual model of the power network for electromagnetic transient studies all components, except transmission lines, are modeled by lumped parameter equivalent circuits composed of voltage and current sources, linear and non-linear resistors, inductors, capacitors, ideal switches, etc. These elements are described in the mathematical model by ordinary differential equations which are solved by step-by-step numerical integration, often using the trapezoidal rule, leading to equivalent circuits consisting of resistors and current sources.

Transmission lines often have dimensions comparable to the wave-length of the high frequency transients and, therefore, have to be modeled as distributed parameter elements described mathematically by partial differential equations (wave equation). For instance, in a transmission line of length ℓ, the voltage and current in a point at a distance x from the sending end, at a time t, are related through the following equation:

$$-\frac{\partial E(x,t)}{\partial x} = L\,\frac{\partial I(x,t)}{\partial t} + R\,I(x,t) \quad (17)$$

$$-\frac{\partial I(x,t)}{\partial x} = C\,\frac{\partial E(x,t)}{\partial t} + G\,E(x,t) \quad (18)$$

where $E(x,t)$ and $I(x,t)$ are $p \times 1$ vectors of phase voltage and currents (p is the number of phases); R, G, L and C are $p \times p$ matrices of the transmission line parameters.

The wave equation does not have an analytic solution in the time domain, in the case of a lossy line, but it has been shown that it can be adequately represented by a traveling wave model consisting of two disjoint equivalent circuits containing a current source in parallel with an impedance in both ends of the line. The value of the current sources are determined by circuit variables computed in past integration steps (*history* terms).

This model is nicely structured for parallel processing: subnetworks of lumped parameter circuit elements connected by transmission lines, representing a group of devices in a substation for instance, can be represented by sets of nodal equations that interface with other groups of equations by the variables required to calculate the current sources in the transmission line equivalent circuits. The exploitation of this characteristic of the network model, in the partitioning of the set of equations for parallel processing, often correspond to a geographical mapping of the power system onto the multiprocessor topology as shown below for a two subnetwork example:

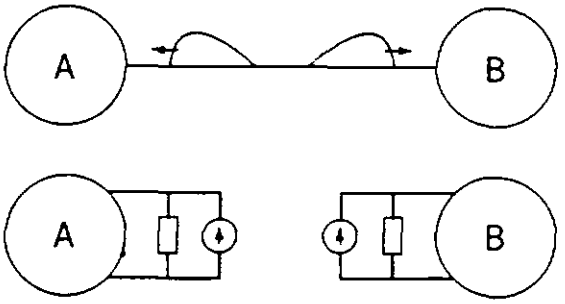

Fig. 6: *Natural* Decomposition of the Electromagnetic Transients Problem Model

$$\begin{bmatrix} G_A & 0 \\ 0 & G_B \end{bmatrix} \begin{bmatrix} E_A \\ E_B \end{bmatrix} + \begin{bmatrix} \mathcal{F}_A(E_A) \\ \mathcal{F}_B(E_B) \end{bmatrix} = \quad (19)$$

$$\begin{bmatrix} I_A^S \\ I_B^S \end{bmatrix} + \begin{bmatrix} I_A^L \\ I_B^L \end{bmatrix} + \begin{bmatrix} I_A^C \\ I_B^C \end{bmatrix} + \begin{bmatrix} I_A^H \\ I_B^H \end{bmatrix} \quad (20)$$

where G_A and G_B are conductance matrices related to linear branch elements; $\mathcal{F}_A$ and $\mathcal{F}_B$ are non-linear functions related to non-linear branch elements; E_A and E_B are vectors of the unknown node voltages; I_A^S, I_B^S are nodal current injections corresponding to independent sources, I_A^L, I_B^L, I_A^C, I_B^C are the nodal injection currents related to the equivalent circuits of inductors and capacitors, and I_A^H, I_B^H are the nodal current injections present in the transmission line models. Figure 6 is a graphical representation of this model.

Since $I^S(t)$ is known and $I^H(t)$, $I^L(t)$, and $I^C(t)$ depend only on terms computed in previous integration steps, $E_A(t)$ and $E_B(t)$ can be computed independently in different processors. The computation of the terms $I_A^S(t)$, $I_B^S(t)$, $I_A^L(t)$, $I_B^L(t)$, and $I_A^C(t)$, $I_B^C(t)$ can also be executed in parallel, since the equations related to branches in a particular subnetwork depend only on nodal voltages belonging to the same subnetwork. However, the term $I_A^H(t)$ depend on the past terms $I_B^H(t-\tau)$ and $E_B(t-\tau)$, as well as $I_B^H(t)$ depend on the past terms $I_A^H(t-\tau)$ and $E_a(t-\tau)$. Since such terms have already been evaluated in previous integration steps, the processors must exchange data in order to each one be able to compute its part of the vector $I^H(t)$.

An electromagnetic transients simulation methodology, based on the above model, was developed and implemented on a prototype of a parallel machine using the Transputer T800 processor [43]. Using a careful handicraft load balancing procedure, it was possible to achieve very high efficiencies (98% in some cases) in the simulation of real size networks.

5.4 Small-Signal Stability

Operational problems caused by electromechanical oscillations weakly damped, or even negatively damped, must be detected and eliminated to avoid deterioration in the quality of the supplied electrical energy and damage to equipment. This type of problem can be studied using linearized versions of the power system dynamic model. The great advantage of this approach is the possibility of the performance assessment of control schemes without time simulation. This assessment is conducted through linear control systems analysis and design methods. A large scale numerical problem resulting from the application of these techniques is the computation of eigenvalues and eigenvectors associated with the state matrix of the linearized system model.

A linearized version of (1) and (2) at an operating point (x_0, z_0) is given by

$$\begin{bmatrix} \Delta\dot{x} \\ 0 \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \end{bmatrix} \qquad (21)$$

where $J_1, \ldots, J_4$ are Jacobian matrices evaluated at the linearization point. The power system state transition equation can be obtained eliminating Δz from (21):

$$\Delta\dot{x} = (J_1 - J_2 J_4^{-1} J_3)\, \Delta x = A\, \Delta x \qquad (22)$$

where A is the system state matrix whose eigenvalues provide information on the singular point stability of the nonlinear system. Efficient algorithms to obtain the dominant eigenvalues and eigenvectors of A for large scale systems do not require the explicit calculation of this matrix [44, 45]. These algorithms can be directly applied to (21), named the *augmented* system matrix, whose sparse structure can be fully exploited to reduce both cpu time and memory requirements. These methods require repeated solutions of linear equation sets of the form [46]:

$$\begin{bmatrix} J_1 - qI & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix}^{(k)} = \begin{bmatrix} r \\ 0 \end{bmatrix}^{(k)} \qquad (23)$$

where w, v are unknown vectors; q is a complex shift used to make dominant the eigenvalues close to q; I is the identity matrix; r is a complex vector; and k is the iteration counter. These equation sets are independent and their solution can be obtained concurrently in different processors. This property make the eigenvalue problem well suited for parallel processing.

In the work reported in [46] and [47], algorithms for the parallel solution of the eigenvalue problem for small-signal stability assessment, using the above formulation, are described and the results of tests with models of a large practical power systems are presented. A first investigatory line of research was based on the parallelization of the Lop-sided Simultaneous Iterations method [46]. The obvious parallel stratagem used was to carry out each trial vector solution on a different processor. Results obtained in tests performed on the iPSC/860 parallel computer, using two large scale representations of the Brazilian South-Southern interconnected power system, presented computation efficiencies around 50%. A second approach to the problem uses a Hybrid Method [47] resulting from the combination of the Bi-Iteration version of the Simultaneous Iteration algorithm and the Inverse Iteration method. The Hybrid algorithm exploits the fast eigenvalue estimation of the Bi-Iteration algorithm and the fast eigenvector convergence of the Inverse Iteration algorithm whenever the initial shift is close to an eigenvalue. In the Inverse Iteration stage, the Hybrid algorithm allows perfect parallelization. The results obtained indicate a superior performance of this method both in terms of computation speedup and robustness.

5.5 State Estimation

State estimation is a basic module in the Energy Management System (EMS) advanced application software. Its main function is to provide reliable estimates of the quantities required for monitoring and control of the electric power system. In almost all state estimation implementations, a set of measurements obtained by the data acquisition system throughout the whole supervised network, at approximately the same time instant, is centrally processed by a static state estimator at regular intervals or by operator's request. Modern high speed data acquisition equipment is able to obtain new sets of measurements every 1-10 seconds but the present EMS hardware and software allow state estimation processing only every few minutes. It has been argued that a more useful state estimation operational scheme would be achieved by shortening the time interval between consecutive state estimations to allow a closer monitoring of the system evolution particularly in emergency situations in which the system state changes rapidly. Another industry trend is to enlarge the supervised network by extending state estimation to low voltage subnetworks. These trends pose the challenge of performing state estimation in a few seconds for networks with thousands of nodes.

The higher frequency in state estimation execution requires the development of faster state estimation algorithms. The larger size of the supervised networks will increase the demand on the numerical stability of the algorithms. Conventional centralized state estimation methods have reached a development stage in which substantial improvements in either speed or numerical robustness are not likely to occur. These facts, together with the technical developments on distributed EMS, based on fast data communication network technology, opens up the

possibility of parallel and distributed implementations of the state estimation function.

In the work reported in [48], the possibility of parallel and distributed state estimation implementation was exploited leading to a solution methodology based on conventional state estimation algorithms and a coupling constraint optimization technique.

The information model used in power system state estimation is represented by the equation

$$z = h(x) + \omega \tag{24}$$

where z is a $(m \times 1)$ measurement vector, x is a $(n \times 1)$ true state vector, $h(.)$ is a $(m \times 1)$ vector of nonlinear functions, ω is a $(m \times 1)$ measurement error vector, m is the number of measurements, and n is the number of state variables. The usual choice for state variables are the voltage phase angles and magnitudes while the measurements are active and reactive power flows and node injections and voltage magnitudes.

Like in load flow calculations, it has been found that state estimation algorithms based on decoupled versions of (24) behave quite adequately for the usual power networks. Therefore, the following decoupled model has been mostly adopted:

$$z_p = h_p(\theta, v) + \omega_p \tag{25}$$
$$z_q = h_q(\theta, v) + \omega_q \tag{26}$$

where θ $(n_\theta \times 1)$ and v $(n_v \times 1)$ are the vectors of true voltage magnitudes and phase angles; p and q are subscripts indicating partitions of vectors and matrices corresponding to active and reactive measurements, respectively; $n_\theta = n - 1$, $n_v = n$, and n is the number of network nodes.

Assume that the power network is decomposed in r areas. The areas are connected through boundary buses which belongs simultaneously to both adjacent areas. Therefore, there are overlapping areas which may be observed using both adjacent measurement sets. The number of boundary buses may be kept to a minimum or incorporate a few extra buses in order to facilitate some estimation functions like bad data processing, for instance. A further assumption is that there are no injection measurements in the overlapping area buses. This assumption does not necessarily represent a practical limitation to the proposed methodology as actual injection measurement buses in overlapping areas can be replaced by fictitious buses with no injection measurement connected to the actual buses, now placed outside the overlapping area, by zero impedance lines.

Under the above assumptions, the state estimation problem introduced in (25) and (26) can be decomposed as follows

$$z_p^k = h_p^k(\theta^k, v^k) + \omega_p^k, \quad k = 1, ..., r \tag{27}$$
$$z_q^k = h_q^k(\theta^k, v^k) + \omega_q^k, \quad k = 1, ..., r \tag{28}$$

where

z_p^k and z_q^k are vectors of active and reactive measurements in area k; dimensions: $(m_p^k \times 1)$ and $(m_q^k \times 1)$, respectively.

θ^k and v^k are vectors of voltage phase angles and magnitudes in area k, including the ones corresponding to the boundary buses; dimensions: $(n_\theta^k \times 1)$ and $(n_v^k \times 1)$, respectively.

The WLS state estimate for the distributed estimation problem depicted in (27) and (28) can be obtained solving a constrained optimization problem with a separable objective function and a set of linear constraints introduced to force the state variables in the overlapping areas to assume the same values. The WLS problem is stated as follows:

$$\min_{\theta^k, v^k} \sum_{k=1}^{r} 1/2\{ [z_p^k - h_p^k(.)]^T [R_p^k]^{-1} [z_p^k - h_p^k(.)] + [z_q^k - h_q^k(.)]^T [R_q^k]^{-1} [z_q^k - h_q^k(.)] \} \tag{29}$$

$$\text{s. to} \sum_{k=1}^{r} A_p^k \theta^k = 0 \tag{30}$$

$$\sum_{k=1}^{r} A_p^k v^{k'} = 0 \tag{31}$$

where A_p^k and A_q^k are $(l \times n_\theta^k)$ and $(l \times n_v^k)$, respectively, matrices (l is the number of boundary buses) whose nonzero elements are either 1 or -1.

The necessary conditions for the solution of the above problem, derived from the corresponding Lagrangian function L, are:

$$\frac{\partial L}{\partial \theta^k} = -[H_p^k]^T [R_p^k]^{-1} [z_p^k - h_p^k(\theta^k, v^k)] + [A_p^k]^T \gamma_p = 0, \quad k = 1, ..., r \tag{32}$$

$$\frac{\partial L}{\partial v^k} = -[H_q^k]^T [R_q^k]^{-1} [z_q^k - h_q^k(\theta^k, v^k)] + [A_q^k]^T \gamma_q = 0, \quad k = 1, ..., r \tag{33}$$

$$\frac{\partial L}{\partial \gamma_p} = \sum_{k=1}^{r} A_p^k \theta^k = 0 \tag{34}$$

$$\frac{\partial L}{\partial \gamma_q} = \sum_{k=1}^{r} A_q^k v^k = 0 \tag{35}$$

where γ_p and γ_q are $(l \times 1)$ vectors of Lagrange multipliers and

$$H_p^k = \frac{\partial h_p^k(\theta^k, v^k)}{\partial \theta^k} \quad \text{and} \quad H_q^k = \frac{\partial h_q^k(\theta^k, v^k)}{\partial v^k}$$

calculated at the solution point.

As in the integrated state estimation approach, the Gauss-Newton method combined with the usual

Fast Decoupled assumptions can be used to solve (32)-(35) leading to the following algorithm:

$$\tilde{\theta}^k(i+1) = \theta^k(i) + [C_p^k]^{-1}\Delta b_p^k(i), \quad k = 1,...,r \qquad (36)$$

$$\gamma_p(i+1) = N_p^{-1}\sum_{k=1}^{r} A_p^k \tilde{\theta}^k(i+1) \qquad (37)$$

$$\theta^k(i+1) = \tilde{\theta}^k(i+1) - [C_p^k]^{-1}[A_p^k]^T\gamma_p(i+1), \quad k = 1,...,r \qquad (38)$$

and

$$\tilde{v}^k(i+1) = v^k(i) + [C_q^k]^{-1}\Delta b_q^k(i), \quad k = 1,...,r \qquad (39)$$

$$\gamma_q(i+1) = N_q^{-1}\sum_{k=1}^{r} A_q^k \tilde{v}^k(i+1) \qquad (40)$$

$$v^k(i+1) = \tilde{v}^k(i+1) - [C_q^k]^{-1}[A_q^k]^T\gamma_q(i+1), \quad k = 1,...,r \qquad (41)$$

where

$$\Delta b_p^k(i) = [H_p^k]^T[R_p^k]^{-1}[z_p^k + - h_p^k(\theta_k(i), v_k(i))] \qquad (42)$$

$$\Delta b_q^k(i) = [H_q^k]^T[R_q^k]^{-1}[z_q^k + - h_q^k(\theta_k(i+1), v_k(i))] \qquad (43)$$

$$C_p^k = [H_p^k]^T[R_p^k]^{-1}H_p^k \qquad (44)$$

$$C_q^k = [H_q^k]^T[R_q^k]^{-1}H_q^k \qquad (45)$$

$$N_p = \sum_{k=1}^{r} A_p^k[C_p^k]^{-1}[A_p^k]^T \qquad (46)$$

$$N_q = \sum_{k=1}^{r} A_q^k[C_q^k]^{-1}[A_q^k]^T \qquad (47)$$

and H_p and H_q are calculated at nominal conditions and kept constant in the iterative process.

Matrices N_p and N_q defined in (46) and (47) play a key role in the distributed state estimation formulation. These matrices present the following relevant characteristics:

- *Structure*: represented by a graph whose nodes correspond to the boundary buses; from a particular node, there are links to all nodes corresponding to boundary buses of the adjacent areas.
- *Elements*: the diagonal elements are the sum of the diagonal elements of the two local area inverse gain matrices (C_p^k and C_q^k) involved in the corresponding constraints; the nonzero off-diagonal elements correspond to the off-diagonal elements of the inverse local area gain matrices; in the case of more than one boundary bus in the overlapping area, the off-diagonal elements related to the interconnection of these buses are the sum of the corresponding elements of the two local area inverse gain matrices.

A straightforward implementation of the algorithm given in (36)-(47) leads to a *hierarchical estimator*. As explained earlier, this kind of estimator is not suitable for parallel or distributed processing. A version of the algorithm more adequate to this type of processing can be obtained neglecting the off-diagonal elements in matrices C_p^k and C_q^k in equations (37,38) and (40,41). In this case, these equations can be merged and rewritten, respectively, as

$$\theta^k(i+1) = \tilde{\theta}^k(i+1) + \Delta\tilde{\theta}^k(i+1) \qquad (48)$$

$$v^k(i+1) = \tilde{v}^k(i+1) + \Delta\tilde{v}^k(i+1) \qquad (49)$$

where all the elements of $\Delta\tilde{\theta}^k(i+1)$ are null except the ones corresponding to boundary buses that are given by

$$\Delta\tilde{\theta}_r^k(i+1) = \pm\frac{g_{rr}^k}{g_{rr}^k + g_{rr}^j}[\tilde{\theta}_r^k(i+1) - \tilde{\theta}_r^j(i+1)] \qquad (50)$$

where g_{rr}^k and g_{rr}^j are diagonal elements corresponding to bus r of the inverse gain matrices of the neighboring areas k and j, respectively, and the sign is set to + or − according to A_p^k. The elements of $\Delta\tilde{v}^k(i+1)$ are calculated similarly.

The algorithm introduced in the previous section calculates the θ and v updates at every iteration in a synchronous way, i.e., it has to wait until the state vector is updated in all areas (or processors) before it starts a new iteration. This approach presents drawbacks regarding both parallel and distributed processing. In parallel processing, synchronous algorithms usually cannot achieve high efficiency due to processor idle time unless a perfect load balancing is obtained which is not easily achievable in most practical applications. In distributed processing, it requires difficult synchronizing operations and reliable communication systems which is not usually present in geographically distributed systems. Asynchronous algorithms, on the other hand, are more flexible for the implementation of both parallel and distributed computation processes.

To achieve asynchronous mode operation, the algorithm defined above has to be slightly modified. The modifications concern the way the components of the vectors defined in (48) and (49) are updated. In the synchronous version, these vectors are calculated using values of the state variables obtained in the *i-th* iteration. In the asynchronous algorithm, these vectors are updated using the *latest available* value of the state variables. Therefore, whenever a particular processor finishes the updating of its state vector, it transmits this information to its adjacent processors (logical adjacency), reads in the values of the boundary buses state variables from its buffer, *regardless of the age of that information*, and proceeds to the calculation of the next area state vector update.

Owing to the local based nature of the state estimation algorithm, simulated experiments have shown that the computation can continue even in the absence of information from other areas. This fact can easily be understood considering that (36) and (39) alone are in fact a local decoupled state estimator. Obviously, in this case the matching of boundary bus state variables would not be achieved. This property make the algorithm suitable to asynchronous implementation. Based on the above property, several iterative schemes can be derived, incorporating different degrees of asynchronism, as will be discussed in latter sections.

The results of computational experiments indicate that the algorithm described above is accurate, robust and efficient in terms of reducing the required computation time. An important by-product of the work is the indication that state estimation is probably a naturally decoupled problem. Important parallel and distributed state estimation issues, like data base access and communication between processors, were not addressed in this work. Also, bad data processing and observability analysis, in the distributed context, were not studied. These issues are topics well suited for further research. Finally, a strong practical advantage of the methodology introduced in this paper is the use of standard state estimation algorithms.

5.6 Load Flow Computations

Load-flow is a fundamental tool in power system studies. It is by far the most often used program in evaluating system security, configuration adequacy, etc., and as a starting point for other computations such as short circuit, dynamic simulation, etc. Its efficient solution is certainly a fundamental requirement for the overall efficiency of several integrated power system analysis and synthesis programs. Therefore, it should be expected a great research effort in the parallelization of the load-flow algorithms. That has not been the case, however, for two main reasons:

- The practical load-flow problem is much more difficult to parallelize than other similar problems owing to constraints added to the basic system of non-linear algebraic equations.

- Very efficient algorithms are already available which can solve large load-flow problems (more than 1000 nodes) in a few seconds on relatively inexpensive computers [1].

More interesting investigatory lines are the parallelization of multiple load-flow solutions (contingency analysis, for instance) [49, 50] and the speed up of load-flow programs in vector processors[51].

5.7 Team Algorithms

Parallel asynchronous implementations of iterative algorithms are now establishing themselves as good choices for high performance computation in distributed-memory environments, in view of several attractive features that they possess, such as ease of implementation, facility with load balancing, shorter convergence times and so on [6]. A new context for asynchronism arose with the introduction of the so-called *Team Algorithms*, which are hybrids of different methods. Such combinations of algorithms were first proposed in the context of power system applications in [52], for sequential computers, and later again in [53, 54, 55], where they were first called *Team Algorithm*, in the context of asynchronous parallel computing, in which such methods have a natural implementation.

The main idea behind a Team Algorithm (TA) is to partition a complex problem in several subproblems and to solve each subproblem in a different processor of a distributed computer system. choosing for each subproblem one or more methods that best solve it. That way, if there are subproblems with different characteristics for which it has been chosen different methods, the resulting combination of methods is a TA.

As an illustration for the TA approach, consider the problem of solving a set of n nonlinear algebraic equations given by:

$$\Phi(x) = 0 \quad , \quad x \in R^n \tag{51}$$

Assume that the elements of (51) can be partitioned as:

$$\Phi(x) = \left[\begin{array}{cc} \Phi_1(x) & \Phi_2(x) \end{array} \right]^T \tag{52}$$

$$x = \left[\begin{array}{cc} x_1 & x_2 \end{array} \right]^T \tag{53}$$

Then, equation (51) may be rewritten as:

$$\Phi_1(x) = 0 \tag{54}$$

$$\Phi_2(x) = 0 \tag{55}$$

An asynchronous implementation of a TA to solve (51), implemented in a distributed memory computing system, can be represented as:

$$\begin{array}{rcl} \chi_1(k+1) & = & \mathcal{G}_1[x^1(k)] \\ \chi_2(k+1) & = & \mathcal{G}_2[x^2(k)] \\ x(k+1) & = & c\,x(k) + w_1\,\chi_1^3(k) + w_2\,\chi_2^3(k) \end{array} \tag{56}$$

where, $\mathcal{G}_i$ $(i = 1, 2)$ are functions representing the iterative processes used in each processor, respectively; $x^i(k)$ represents the most recently received value of vector x in processor i at iteration k, while $\chi_i^3(k)$, $(i = 1, 2)$, represents the most recent value of χ_i at iteration k, available to processor 3 (the Administrator).

The TA given by (56) works as follows: each processor i $(i = 1, 2)$ tries to solve problem (4) using

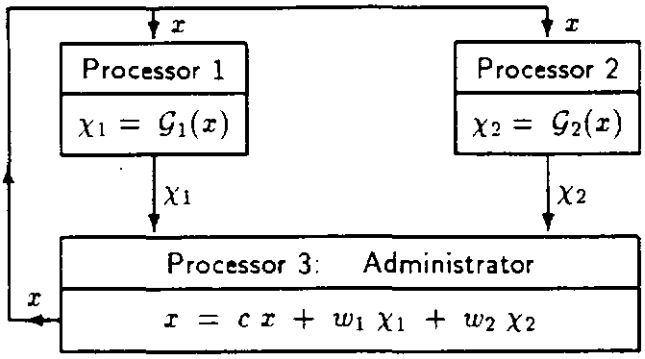

Fig. 7: Example of Team Algorithm.

an iterative algorithm represented by the map $\mathcal{G}_i$, using the most recent value of x received from the Administrator, and transmitting the updated value χ_i. At the same time, the Administrator updates x combining most recently received values of χ_i, using properly chosen non-negative *weights* (w_1, $w_2 \geq 0$). The constant c is chosen to ensure that for a solution x^*, the following condition holds:

$$x = \chi_1 = \chi_2 = x^* \in R^n,$$

which, in turn, implies that:

$$c = 1 - w_1 - w_2.$$

This solution process is illustrated in figure (7).

The advantages of using a TA in distributed load flow computations is demonstrated in the research work reported in [56, 57, 58]. In this work, two load flow example problems are studied. For these examples, neither the Fast Decoupled method nor the block Jacobi version of the Y matrix method can properly solve the problem; however, a TA combining both methods solves the problems with the additional advantage of being easily parallelized in an asynchronous environment with an excellent speedup. That way, it was possible to exploit all the potential of distributed memory computers, as the Intel iPSC/860 hypercube of 8 nodes used for the implementations described in the referred work.

6 Research Activities in Brazilian Institutions

In Brazil, research activities in parallel processing application to power system problems have been showing a fairly fast pace in the last ten years or so. At least three Brazilian institutions have research groups working in this area: COPPE[1], CEPEL[2], and UNICAMP[3].

[1] The Postgraduate School of Engineering of the Federal University of Rio de Janeiro.

[2] The Brazilian Utilities Research Center.

[3] The University of Campinas.

COPPE started a research project in the application of parallel processing to engineering problems in 1986. The project joined together professors, researchers and graduate students of the Computer Science, Civil, and Electrical Engineering Departments. The objective of the project is the development of a complete cycle of parallel processing technology comprising hardware, basic and application software. The project has been financed by the Brazilian Government Agencies FINEP and CNPq/RHAE. An Intel iPSC/860 parallel computer, with hypercubic topology and eight processing nodes, has been used in this project. An IBM/SP-2 parallel computer with eight cpu's and a Cray J90/8 supercomputer are in the process of acquisition by COPPE and are planned to become available for research activities in early 1995. As a result of this project, several software methodologies and programs for parallel processing have been developed. The ones related to power system applications have already been mentioned in previous section of this paper. Also, a prototype of a parallel machine based on the Transputer T800 processor was built and has been used in the development of applications.

CEPEL and UNICAMP started their activities in this research area using a parallel machine known as PP (Preferential Processor) developed at the research center of the Brazilian Telecomunication Companies (CPqD/TELEBRAS). This machine is based on a hybrid shared/distributed memory architecture and the Intel 80286 processor. A few prototypes of this machine were built, some of which are still in operation, but the project was later discontinued.

CEPEL developed a parallel programming environment for the PP, based on the MS-DOS operating system extended with facilities for communication and synchronization, which was used in power system applications such as multi-area reliability evaluation, hydrothermal production costing and reliability evaluation, and security constrained optimal power flow [59, 60]. This programming environment was also used at UNICAMP to develop a parallel methodology to solve sets of network equations based on the matrix inverse factors (W-matrix) approach [29]. After the discontinuity of the PP project, the UNICAMP power system group acquired a nCUBE-2 with 64 processors that has been used in research projects in security constrained optimal power flow, probabilistic short-circuit analysis, etc. [18, 61]. CEPEL is also developing a long term project in the application of distributed architectures and open system concepts to EMS design [20].

The availability of parallel machines for research and development activities in Brazilian institutions has also increased considerably in the last few years.

Table 3: Parallel computers and supercomputers available in Brazil

Make Model	Number of cpu's	Institution
Intel iPSC/860	8	COPPE
nCube	64	UNICAMP
IBM SP-1	16	LNCC
IBM SP-1		UNICAMP
Cray YMP	2	UFRGS
Cray J90/8	8	COPPE*
IBM SP-2	8	COPPE*

*Available in early 1995.

Table 3 gives a partial relation of parallel computers and supercomputers available in Brazil. Most of these machines are open to use for the academic community and accessible via computer communication networks.

7 Concluding Remarks

Parallel processing may be the only way to make viable some power system applications requiring high performance computing like real-time dynamic security assessment, security constrained optimal power flow, real-time simulation of electromagnetic and electromechanical transients, composite reliability assessment using realistic models, etc. Parallel computers are presently available in a price range compatible with power system applications and presenting the required computation power.

Two main factors are still impairments to the wide acceptance of these machines in power system applications: the requirements for reprogramming or redevelopment of applications and the uncertainty about the prevailing parallel architecture. The first problem is inevitable, as automatic parallelization tools are not likely to become practical in the near future, but has been minimized by the research effort in parallel algorithms and the availability of more efficient programming tools. The second difficulty is destined to disappear as parallel computers leave the laboratories and arrive in the showrooms of computers' manufacturers.

Likewise in the history of sequential computer evolution, a unique and overwhelming solution to the parallel computer architecture problem is not to be expected. It is more likely that a few different architectures will be successful in the next few years and the users will have to decide which one is the most adequate for their application. Moreover, it is probably that commercial processing applications, which are now turning towards parallel processing, are the ones that will shape the future parallel computer market. However, to make this scenario a little bit less uncertain, it should be pointed out the tendency in the parallel computer industry to make their products follow open system standards and the possibility of developing applications less dependent of a particular architecture.

Distributed processing is a prevailing technology in many information processing applications. Combined with the concept of open system, it offers a better performance/cost ratio, more reliability and flexibility, scalability, etc. In the power utility environment, it has been used in distribution automation and is substituting the minis and mainframes in the supervisory and control centers. In the future, networks of microcomputers and workstations will probably integrate operation and planning information systems with accounting and billing processing systems allowing direct information transfer and resource sharing.

The most likely scenario for the electric utility information processing system of the future will be a conglomerate of local area networks serving specific functions (control centers, expansion and operational planning, billing, substation control, etc), built around standard hardware and software, in which each computer has specific characteristics like sophisticated graphic terminals, data base servers, etc. Some of the computers may actually be parallel machines that will act as *number crunching servers* for highly intensive numerical applications. These local area networks will be joined together by a wide area network connecting computers in the corporation headquarters and widespread in the whole area serviced by the utility.

Acknowledgments

Most of the ideas contained in this paper are the result of a long and fruitful collaboration with Prof. E. Kaszkurewicz and Prof. A. Bhaya (COPPE) and our present and former students I.C. Decker, H.L.S. Almeida, M.H.M. Vale, B. Baran, J.M. Campagnolo, C.L.T. Borges, and D.Q. Siqueira. More recent work has been developed in collaboration with Prof. F.F. Wu and Dr. L. Murphy (U.C. Berkeley, USA) in Parallel and Distributed State Estimation, and Dr. Nelson Martins (CEPEL) and Prof. J.L.R. Pereira (UFJF) in Parallel Small Signal Stability Assessment. Part of this paper text and some figures were extracted from thesis and papers written by or with people referred to above.

References

[1] J.V. Mittsche, "Stretching the Limits of Power System Analysis", *IEEE Computer Application in Power*, vol. 6, no. 1, pp. 16-21, January 1993.

[2] *IEEE Spectrum, Special Issue: Supercomputers*, September 1992.

[3] T.G. Lewis and H. El-Rewini, *Introduction to Parallel Computing*, Prentice Hall, New York, 1992.

[4] M.J. Quinn, *Parallel Computing: Theory and Practice*, McGraw-Hill, New York, 1994.

[5] D.J. Tylavsky, A. Bose, et al., "Parallel Processing in Power System Computation", *IEEE Transactions on Power Systems*, vol. 7, no. 2, pp. 629-637, May 1992.

[6] D.P. Bertsekas and J.N. Tsitsikilis, *Parallel and Distributed Computation*, Prentice Hall, New York, 1989.

[7] A. Umar, *Distributed Computing: A Practical Synthesis*, Prentice-Hall, New Jersey, 1993.

[8] B. Stott, "Review of Load Flow Calculation Methods", *Proceedings of the IEEE*, vol. 62, no. ??, pp. 916-929, July 1974.

[9] B. Stott, "Power System Dynamic Response Calculations", *Proceedings of the IEEE*, vol. 67, no. 2, pp. 219-241, February 1979.

[10] H.W. Dommel and W.S. Meyer, "Computation of Electromagnetic Transients", *Proceedings of the IEEE*, vol. 62, no. ???, pp. 983-993, July 1974.

[11] B. Stott, O. Alsac, and A. Monticelli, "Security Analysis and Optimization", *Proceedings of the IEEE*, vol. 75, no. 12, pp. 1623-1644, December 1987.

[12] N.J. Balu, et al., "On-Line Power System Security Analysis", *Proceedings of the IEEE*, vol. 80, no. 2, pp. 262-280, February 1992.

[13] Y. Sekine, K. Takahashi, and T. Sakaguchi, "Real-Time Simulation of Power System Dynamics", *Proceedings of the 11th Power Systems Computation Conference*, Avignon, France, Aug. 30 - Sep. 3 , 1993.

[14] H. Taoka, I. Iyoda, H. Noguchi, N. Sato, and T. Nakazawa, "Real-Time Digital Simulator for Power System Analysis on a Hypercube Computer", *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 1-10, February 1992.

[15] D. Brandt, R. Wachal, R. Valiquette, and R. Wierckx, "Closed Loop Testing of A Joint VAR Controller Using a Digital Real-Time Simulator", *IEEE Transactions on Power Systems*, vol. 6, no. 3, pp. 1140-1146, August 1991.

[16] M.V.F. Pereira and N.J. Balu, "Composite Generation/Transmission Reliability Evaluation", *Proceedings of the IEEE*, vol. 80, no. 4, pp. 470-491, April 1992.

[17] A.M. Leite da Silva, J. Endrenyi and L. Wang, "Integrated Treatment of Adequacy and Security in Bulk Power System Reliability Evaluation", *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 275-285, February 1993.

[18] F. Sato, A.V. Garcia, and A. Monticelli, "Parallel Implementation of Probabilistic Short-Circuit Analysis by the Monte Carlo Approach", *Proceedings of the 18th Power Industry Computer Applications Conference*, Scottsdale, AZ, May 1993.

[19] *IEEE Tutorial Course on Fundamentals of Supervisory Systems*, Publication 91EH0337-6-PWR, 1991.

[20] L.C. Lima et al., "Design and Development of an Open EMS", *Proceedings of the Joint International Power Conference (Athens Power Tech)*, Athens, Greece, September 1993.

[21] *IEEE Tutorial Course on Distribution Automation*, Publication 88H0280-8-PWR, 1988.

[22] W.R.Cassel, "Distribution Management Systems: Functions and Payback", *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 796-801, August 1993.

[23] L. Murphy and F.F. Wu, "An Open Design Approach for Distributed Energy Management Systems", *IEEE Transactions on Power Systems*, vol. 8, no.3, pp. 1172-1179, August 1993.

[24] M.H.M Vale, D.M. Falcão, and E. Kaszkurewicz, "A Semiautomatic Network Decomposition Method for Block-Iterative Solutions on Parallel Computers", submitted to the 1995 IEEE Power Industry Computer Applications Conference, Salt Lake, USA, May 1995.

[25] W.L. Hatcher, F.M. Brasch, and J.E. Van Ness, "A Feasibility Study for the Solution of Transient Stability Problems by Multiprocessor Structures", *IEEE Transactions on Power Apparatus and Systems*, vol. 96, no. 6, pp. 1789-1797, Nov./Dec. 1977.

[26] F.M. Brasch, J.E. Van Ness, and S.C. Kang, "Simulation of a Multiprocessor Network for Power System Problems", *IEEE Transactions on Power Apparatus and Systems*, vol. 101, no. 2, pp. 295-301, 1982.

[27] J.S. Chai, N. Zhu, A. Bose, and D.J. Tylavsky, "Parallel Newton Type Methods for Power System Stability Analysis Using Local and Shared Memory Multiprocessors", *IEEE Transactions on Power Systems*, vol. 6, no. 4, pp. 1539-1545, November 1991.

[28] J.S. Chai and A. Bose, "Bottlenecks in Parallel Algorithms for Power System Stability Analysis", *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 9-15, February 1993.

[29] A. Padilha and A. Morelato, "A W-Matrix Methodology for Solving Sparse Network Equations on Multiprocessor Computers", *IEEE Transactions on Power Systems*, vol. 7, no. 3, pp. 1023-130, August 1992.

[30] A.F.C. Canto e J.L.R. Pereira, "Solução de Equações da Rede Elétrica Utilizando Processamento Sequencial e Paralelo", *Anais do 10 Congresso Brasileiro de Atomática*, Rio de Janeiro, RJ , Setembro 1994.

[31] I.C. Decker, D.M. Falcão, and E. Kaszkurewicz, "An Efficient Parallel Method for Transient Stability Analysis", *Proceedings of the 10th Power Systems Computation Conference*, Graz, Austria , pp. 509-516, August 1990.

[32] I.C. Decker, D.M. Falcão, and E. Kaszkurewicz, "Parallel Implementation of a Power System Dynamic Simulation Methodology Using the Conjugate Gradient Method", *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 458-465, February 1992.

[33] I.C. Decker, D.M. Falcão, and E. Kaszkurewicz, "Conjugate Gradient Methods for Power System Dynamic Simulation in Parallel Computers", submitted to the *1995 IEEE PES Summer Meeting*.

[34] M.H.M Vale, D.M. Falcão, and E. Kaszkurewicz, "Electrical Power Network Decomposition for Parallel Computations", *IEEE Symposium on Circuits*

and Systems, San Diego, CA, pp. 2761-2764, May 1992.

[35] M.Ilić-Spong, M.L. Crow, and M.A. Pai, "Transient Stability Simulation by Waveform Relaxation Methods", *IEEE Transactions on Power Systems*, vol. 2, no. 4, pp. 943-952, November 1987.

[36] M.L. Crow and M. Ilić, "The Parallel Implementation of the Waveform Relaxation Method for Transient Stability Simulations", *IEEE Transactions on Power Systems*, vol. 5, no. 3, pp. 922-932, August 1990.

[37] F. Alvarado, "Parallel Solution of Transient Problems by Trapezoidal Integration", *IEEE Transactions on Power Apparatus and Systems*, vol. 98, no. 3, pp. 1080-1090, May/June 1979.

[38] M. LaScala, A. Bose, D.J. Tylavsky, and J.S. Chai, "A Highly Parallel Method for Transient Stability Analysis", *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1439-1446, November 1990.

[39] M. LaScala, M. Brucoli, F. Torelli, and M. Trovato, "A Gauss-Jacobi-Block Newton Method for Parallel Transient Stability Analysis", *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1168-1177, November 1990.

[40] J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

[41] J.J. Dongarra, I.S. Duff, D.C. Sorensen, and H.A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, 1991.

[42] H.A. van der Vorst, "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems", *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 2, pp. 631-644, March 1992.

[43] D.M. Falcão, E. Kaszkurewicz and H.L.S. Almeida, "Application of Parallel Processing Techniques to the Simulation of Power System Electromagnetic Transients", *IEEE Transactions on Power Apparatus and Systems*, vol. 8, no. 1, pp. 90-96, February 1993. .

[44] N. Martins, "Efficient Eigenvalue and Frequency Response Methods Applied to Power System Small-Signal Stability Studies", *IEEE Transactions on Power Apparatus and Systems*, vol. 1, no. 1, pp. 217-226, February 1986.

[45] N. Martins, L.T.G. Lima, H.J.C.P. Pinto, and N.J. P. Macedo, "A State-of-the-Art Computer Program Package for the Analysis and Control of Small Signal Stability of Large AC/DC Power Systems", *Proceedings of the IERE Workshop on New Issues in Power System Simulation*, Caen, France, pp. 11-19, March 1992.

[46] J.M. Campagnolo, N. Martins, J.L.R. Pereira, L.T. G. Lima, H.J.C.P. Pinto, and D.M.Falcão, "Fast Small-Signal Stability Assessment Using Parallel Processing", *IEEE Transactions on Power Apparatus and Systems*, vol. 9, no. 2, May 1994.

[47] J.M. Campagnolo, N. Martins, and D.M. Falcão, "An Efficient and Robust Eigenvalue Method for Small-Signal Stability Assessment in Parallel Computers", presented at the *1994 IEEE PES Summer Meeting*, San Francisco, CA, July 1994.

[48] D.M. Falcão, F.F. Wu, and L. Murphy, "Parallel and Distributed State Estimation", presented at the *1994 IEEE PES Summer Meeting*, San Francisco, CA, July 1994.

[49] D.Q. Siqueira, *Power System Static Contingency Analysis Using Parallel Processing*, M.Sc. Thesis, COPPE, 1991, (In Portuguese).

[50] D.P. Pinto e J.L.R. Pereira, "Um Método Integrado para Análise de Contingências Usando Processamento Paralelo ", *Anais do 10 Congresso Brasileiro de Atomática*, Rio de Janeiro, RJ , Setembro 1994.

[51] C.L.T. Borges, *Performance Assessment of Power Flow Solution Methods for Parallel and Vector Processing*, M.Sc. Thesis, COPPE, 1991, (In Portuguese).

[52] Y.P. Dusonchet, S.N. Talukdar, H.E.Sinnot. and A.H. El-Abiad, "Load Flows Using a Combination of Point Jacobi and Newton's Method", *IEEE Transactions on Power Apparatus and Systems*, vol. 90, pp. 941-949, 1971.

[53] S.N. Talukdar, S.S. Pyo, and R. Mehrotra, "Design Algorithms and Assignments for Distributed Processing", *EPRI Report EL-3317*, 1983.

[54] R. Mehrota and S.N. Talukdar, "Task Scheduling on Multiprocessors for Power System Problems", *IEEE Transactions on Power Apparatus and Systems*, vol. 102, pp. 3590-3597, 1983.

[55] S.N. Talukdar, S.S. Pyo, and T.C. Giras, "Asynchronous Procedures for Parallel Processing", *IEEE Transactions on Power Apparatus and Systems*, vol. 102, pp. 3652-3659, 1983.

[56] B. Barán, *An Study of Asynchronous Team Algorithms*, D.Sc. Thesis, COPPE, 1993, (In Portuguese).

[57] B. Barán, E. Kaszkurewicz, and A. Bhaya, "Parallel Asynchronous Team Algorithms: Convergence and Performance Analysis", submitted to *Scientific Computing, SIAM*, 1994.

[58] B. Barán, E. Kaszkurewicz, D.M. Falcão, "Team Algorithms in Distributed Load Flow Computations", submitted to the *1995 IEEE PES Winter Meeting*.

[59] M.J. Teixeira, H.J.C. Pinto, M.V.F. Pereira, and M.F. McCoy, "Developing Concurrent Processing Applications to Power System Planning and Operations", *IEEE Transactions on Power Systems*, vol. 5, no. 2, pp. 659-664, May 1990.

[60] H.J.C. Pinto, M.V.F. Pereira, and M.J. Teixeira, "New Parallel Algorithms for the Security-Constrained Dispatch with Post-Contingency Corrective Actions", *Proceedings of the 10th Power Systems Computation Conference*, Graz, Austria , August 1990.

[61] M. Rodrigues, O.R. Saavedra, and A. Monticelli, "Asynchronous Programming Model for the Concurrent Solution of the Security Constrained Optimal Power Flow Problem", *IEEE/PES 1993 Summer Meeting*, Vancouver, Canada, July 1993.