Power Systems Transient Stability - A Grand Computing Challenge

D. P. Koester, S. Ranka, and G. C. Fox
School of Computer and Information Science and
The Northeast Parallel Architectures Center (NPAC)
Syracuse University
Syracuse, NY 13244-4100
dpk@npac.syr.edu, ranka@top.cis.syr.edu, gcf@npac.syr.edu

NPAC Technical Report - SCCS 549

31 August 1992

Abstract

Real-time or faster-than-real-time power system transient stability simulations will have significant impact on the future design and operations of both individual electrical utility companies and large interconnected power systems. Sufficiently fast transient stability simulation implementations may significantly improve power system reliability which, in turn, will positively affect electrical utility company profits, environmental impact, and customer satisfaction. Past research into techniques to enhance the performance of transient stability simulations has included both concurrent processing and better algorithms, however, there are still considerable areas for research into this problem. The scope of real-time or faster-than-real-time transient stability analysis places this application in the category of being a grand computing challenge that could benefit from future teraflop (*trillion* floating-point operations per second) supercomputers. This paper describes various research areas that are of interest to the computational science academic community that offer promise to improve the quality and performance of power system transient stability simulations. Many of the research areas offer competitive alternatives for performance improvements; consequently, a parallel transient stability testbed at the Northeast Parallel Architectures Center (NPAC) at Syracuse University is proposed. We believe that it is possible to develop scalable transient stability algorithms using both concurrent computers and more efficient algorithms that could offer speedups of 100 on 32-processor distributed-memory multicomputers when compared to sequential codes such as the EPRI-Extended Transient/Mid-Term Stability Program. Even greater speedups would be possible on multicomputers with more processors.

1 Introduction

Transient stability analysis examines the dynamic behavior of a power system for as much as several seconds following a power flow disturbance. Transient stability analysis is concerned with the electrical distribution network, electrical loads, and the electro-mechanical equations of motion of the interconnected generators [1, 3, 46]. Traditionally, power system transient stability analysis has been performed off-line to understand the system's ability to withstand specific disturbances and the system's response characteristics, such as damping of generator oscillations, as a system returns to normal operations. Such contingency studies were limited to the system design/upgrade phase in order to ensure robust network design and limited to operator training exercises in order to assist with robust power system operations [3].

To date, the computational complexity of transient stability stability problems have kept them from being run in real-time to support decision making at the time of a disturbance. If a transient stability program could run in real-time or faster-than-real-time, then power system control-room operators could be provided with a detailed view of the scope of cascading failures. This view of the unfolding situation could assist an operator in understanding the magnitude of the problem and its ramifications so that proactive measures could be taken to limit the extent of the incident. Faster transient stability simulation implementations may significantly improve power system reliability which in turn will directly or indirectly affect:

1. electrical utility company profits
2. environmental impact
3. customer satisfaction

In addition to real-time analysis, there are other areas where transient stability analysis could become an integral part of daily power system operations:

1. system restoration analysis
2. economic/environmental dispatch
3. expansion planning

Real-time or faster-than-real-time transient stability analysis could also be a significant benefit to an operator when a power system is being restored after an outage. Incorrect decisions concerning the order to switch loads and generator capacity back on-line could cause recurrences of cascading system failures or even physical damage to generators, transformers, or power lines. In the near future, economic and environmental dispatch will combine with limitations on generation expansion and also combine with increasing load demand to force electrical power systems to operate closer to reliable operating margins; consequently, more frequent considerations of transient stability may be required [5].

It will be shown that computational requirements are a significant problem with transient stability simulations. The scope of real-time or faster-than-real-time transient stability analysis places this application in the category of being a grand computing challenge that could benefit from future teraflop (*trillion* floating-point operations per second) supercomputers. Past research into techniques to enhance the performance of transient stability simulations has included both concurrent processing and better algorithms, however, there are still considerable areas for research into this problem.

This paper provides background on the power system transient stability problems, and discusses various techniques available to improve the computational tractability of the transient stability problem. The goal is to develop efficient, accurate implementations of power system simulations that are sufficiently fast to offer inputs to interactive operator support routines. Techniques are examined to solve differential-algebraic equations — the computational heart of the transient stability problem. An examination of previous research reported in the literature has been used to form the basis of a discussion of relevant numerical research topics for transient stability analysis. The paper describes plans to develop a parallel transient stability analysis testbed at the Northeast Parallel Computing Center (NPAC) at Syracuse University to permit a consistent test environment to develop new algorithms and parallel computing applications as research is performed on this grand computing challenge. Lastly, estimates of the magnitude of potential speedup in power systems transient stability simulations are presented.

2 Power System Transient Stability Simulations

Transient stability analysis examines the dynamic behavior of power system electrical distribution networks, electrical loads, and the electro-mechanical equations of motion of the interconnected generators for as much as several seconds following a disturbance [1, 3, 46]. Under normal operating conditions, an electrical power system is near equilibrium, with only minor deviations from true steady-state conditions caused by small, nearly continuous, changes in the loads. When a short circuit occurs in the power distribution network, there are significant, nearly instantaneous, changes in the loads at some generators in the system.

Mechanical controls in the generators react slower than the electromagnetic loads, so there is the possibility that instead of the power system returning to a steady-state condition after the disturbance, one or more generators may encounter sufficient variations in rotational speed that they loose synchronization with the power network and must be taken off-line to avoid catastrophic problems. If a generator must be taken off-line because it has lost synchronization, there will be a decrease in available generator capacity and another source of disruption will be injected into the power system. Cascading system failures can cause wide spread power outages, reduce the interconnected power grid to islands of power service, and even cause physical damage to generating equipment [7].

There is a simple mechanical analog to the transient stability problem that will assist in understanding the nature of the problem. Consider a number of masses, that represent the electrical generators in a power system, suspended within a network of elastic strings, that represent the electrical transmission lines. A sudden loss of a transmission line can be modeled by cutting a string in the steady state network. The forces in the remaining strings will fluctuate, and the masses will experience coupled motion that is dependent on the network and the tensions in the strings. This disturbance may cause two effects:

1. The network will settle down into a new steady-state condition where the forces in the strings represent the electrical voltages in the power system.

2. One or more additional strings could break resulting in a chain reaction that reduces the network into small isolated groups of weights and strings.

A network is *transient stable for the fault in question* if the system is able to survive the disturbance and return to steady-state [27].

The reaction of a single generator to variations in its load can be modeled using ordinary differential equations (ODEs):

$$\dot{y} = f(y, z). \tag{1}$$

However, there are multiple generators supplying power to the network and the generators are coupled through the power network. The power network can be modeled by non-linear algebraic equations:

$$0 = g(y, z), \tag{2}$$

and this entire system of differential-algebraic equations (DAEs) must be solved simultaneously (or nearly so). Transient stability analysis is computationally intensive because the large systems of DAEs must be solved at small time increments to ensure that errors are minimized in the numerical integration of the potentially computationally-stiff ODEs [41]. The dynamics of each generator are individually modeled with as few as two and with as many as forty differential equations while the generators are coupled via the algebraic equations that describe the electrical network. This requires the solution of large systems of simultaneous sparse non-linear equations.

The number of equations required to solve the transient stability DAEs are extremely large and very sparse. For example, if there are twenty differential equations to describe each generator and two equations to describe the complex voltage/current at each network bus, then a state-wide or regional interconnected power system with 2000 buses and 300 generators could generate a sparse, irregular system of 10,000 non-linear algebraic equations that must be solved simultaneously. In this example, the matrix would be formed by 300 blocks of generator equations along the diagonal with additional blocks formed from the network equations. After reordering the matrix, all equations with variables outside of the blocks along the diagonal will be relocated into narrow borders along the bottom and right side of the matrix. Each generator equation may have as many as 20 variables, and each network equation would have only between two and ten variables due to the limited number of transmission lines at any bus.

An example of the block-bordered diagonal matrix for a smaller power system with only ten generators is depicted in figure 1. Generator equations occur in the blocks along the diagonal starting at the upper-left-hand side of the diagram, and are labeled G1 to G10. The network equations occur in the lower-right-hand side of the matrix, and in this representation, the network equations have been reordered into block-bordered-diagonal form according to the natural hierarchy encountered in electrical power distribution networks. The blocks along the diagonals, in this portion of the matrix, are independent portions of the network, while the *inner borders* contain network equations that depict the interconnections between independent portions of the electrical network. The *outer borders* contain equations that relate generator current to voltages at the buses connected to generators.

There are several levels of inherent hierarchy in a state or regional power network due to the clustering of loads within power systems, interconnections between load clusters for system reliability within an electrical utility, and the interconnections between utilities for added reliability and the inter-utility sale of electricity [22, 31]. A simple two-level hierarchical network and block-bordered diagonal matrix is illustrated in figure 2.

Solutions of simultaneous nonlinear equations are performed using iterative techniques such as Newton's method that require the solution of one or more linear systems of equations

$$Ax = b. \tag{3}$$

The Jacobian matrix used in the Newton method has the same sparsity pattern as the original matrix. Examples in the literature discuss using time-step sizes that would require between 12 and 100 time-steps per second of simulation time [10, 25, 33, 34, 35, 47]. Consequently, a real-time or faster-than-real time transient stability analysis program could require the solution of systems of 10,000 non-linear algebraic equations a hundred times per second, and each solution of the systems of nonlinear equations could require multiple solutions of a similar number of linear equations that require forward reductions and backward substitutions for a number of vectors equal to the number of linear equations [20].

At this point, it would be desirable to develop estimates of the number of floating-point operations required to solve these linear equations, in order to calculate a de-

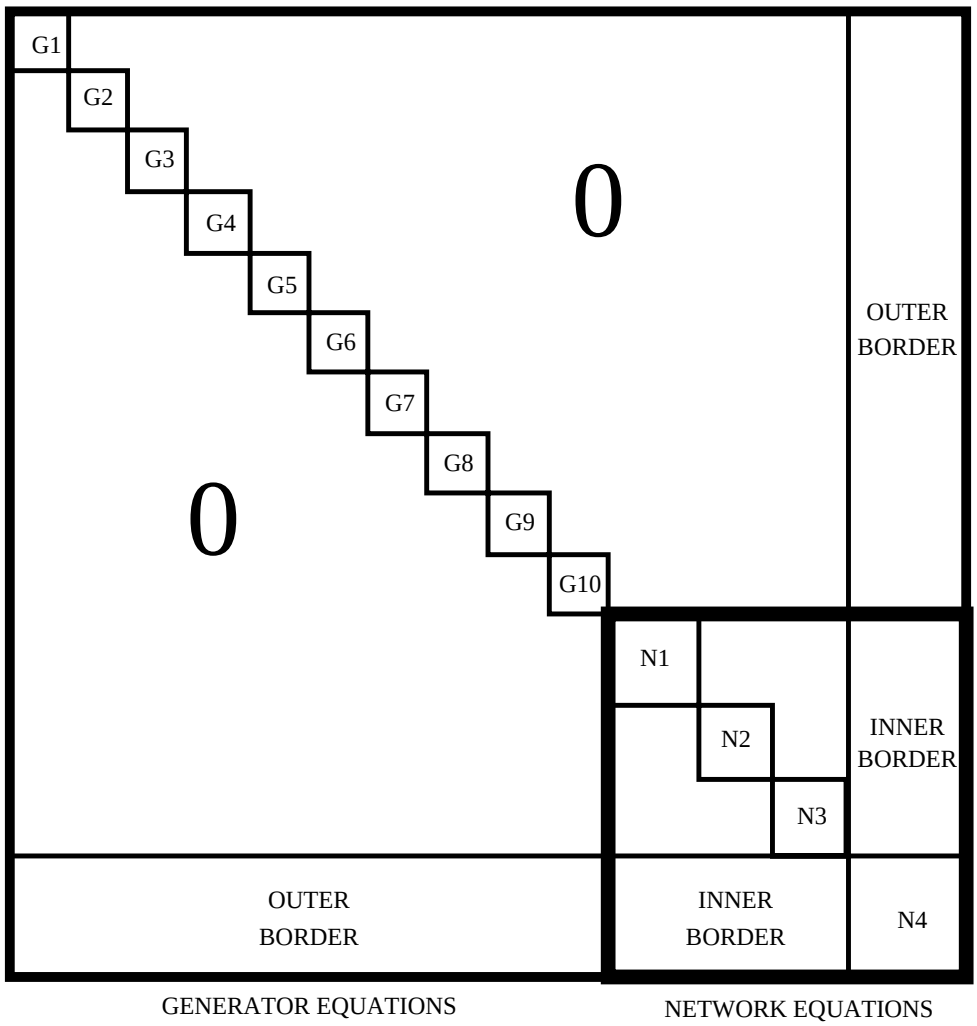

Figure 1: Block-Bordered Diagonal Matrix Form Derived from the Power System Transient Stability Differential-Algebraic Equations

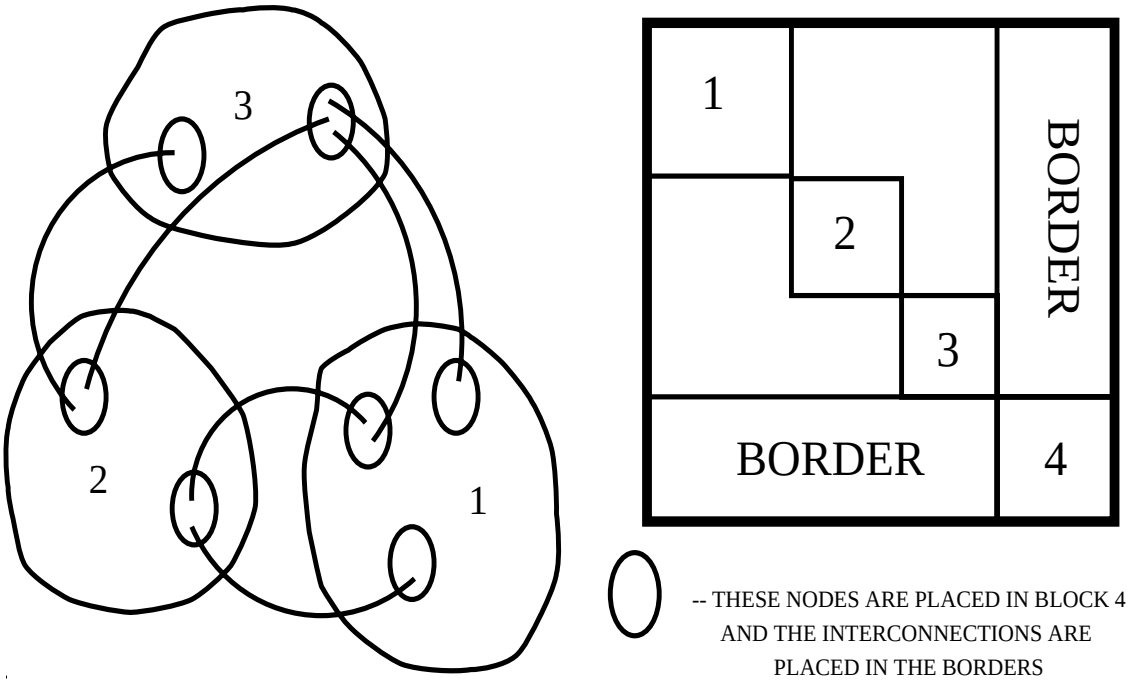

Figure 2: Hierarchical Power System Network Mapping to a Block-Bordered Diagonal Matrix Form

tailed analysis of the potential performance on a target parallel architecture, however, such an estimate would be highly dependent on the number of equations in the actual sparse matrix, and the sparsity structure in the matrix [14]. The sparsity structure can directly affect the number of calculations; although there are many techniques available to *reorder* a matrix in order to reduce the number of calculations and/or increase the amount of calculations that can be performed concurrently [23]. Estimates of the number of floating point operations in the calculations of the transient stability DAEs will be used in subsequent research as portions of objective functions to determine optimal parallel processing load balancing, however, without more details on a specific, large interconnected power system in this example, it is not possible to give more detailed estimates of the number of floating point operations. Nevertheless, it is possible to develop estimates of potential algorithmic and concurrent performance improvements using other approximate techniques based on the amount of overhead in a concurrent algorithm and the fraction of sequential operations in a concurrent implementation, using Amdahl's law [32, 37, 38]. Such performance estimates will be

discussed later in this paper.

3 Techniques to Speedup Transient Stability Simulations

At present, techniques available to the computational scientist to either improve the performance of an application or to make the implementation of a grand computing challenge problem even feasible fall into two categories:

1. faster hardware
2. more efficient algorithms

While emphasis is often primarily directed to the application of more computing power, the development of more efficient underlying algorithms can also be an effective means to reduce the wall-clock time for a problem as computationally intensive as transient stability simulations. Meanwhile, when attempting to enhance the performance of an application, a strong synergism exists between faster hardware and algorithms. In particular, the use of supercomputer hardware requires research into algorithms that make efficient use of vector processing or parallel processing architectures, especially when significant portions of applications are not *embarassingly parallel*.

Supercomputer technologies based on scalar or vector processing architectures are rapidly approaching the physical limitations of circuit size and logic switching speeds; consequently, concurrent or parallel processing has become the focus for addressing computational grand challenges [39]. Without a doubt, concurrent processing architectures will be the basis for the forthcoming teraflop (*trillion* floating-point operations per second) supercomputers. Detailed, accurate, faster-than-real-time transient stability analysis for state or regional electrical power governing authorities will require these impressive computing capabilities to be effectively harnessed using efficient algorithms.

Basic definitions of speedup for sequential and concurrent algorithms are required in order that preliminary estimates of algorithm performance can be developed with-

out extensive modeling of algorithms, computer architectures, and particular power systems.

Definition 1 (Speedup) *Given a single problem with two sequential algorithms that exhibit execution times of T_1 and T_2 with $T_2 < T_1$, speedup is defined as*

$$S_A \equiv \frac{T_1}{T_2}. \tag{4}$$

This simple, intuitive definition will be expanded in order to compare the performance of sequential and concurrent algorithms.

Definition 2 (Relative Speedup) *Given a single problem with a sequential algorithm running on one processor and a concurrent algorithm running on p independent processors, relative speedup is defined as*

$$S_p \equiv fracT_1T_p, \tag{5}$$

where T_1 is the time to run the sequential algorithm as a single process and T_p is the time to run the concurrent algorithm on p processors.

Definition 3 (Fair Speedup) *Given a single problem with a sequential algorithm and concurrent algorithm running on p independent processors, fair speedup is defined as*

$$\hat{S}_p \equiv \frac{T_{seq}}{T_p}, \tag{6}$$

where T_{seq} is the time to run the most efficient sequential algorithm as a single process and T_p is the time to run the concurrent algorithm on p processors.

Definition 4 (Amdahl's Law) *Given T_1, the time to solve a problem on a single processor, then T_p can be parameterized in $\alpha \in [0, 1]$ by*

$$T_p \equiv \alpha T_1 + (1 - \alpha)\frac{T_1}{p}, \tag{7}$$

where α is the inherently sequential fraction of computations. The aforementioned estimate of T_p can be used when estimating the relative speedup S_p [37] by

$$S_p = \frac{p}{1 + (p - 1)\alpha} = \frac{1}{\alpha + (1 - \alpha)/p} \leq S_\infty \equiv \alpha^{-1}. \tag{8}$$

Amdahl's Law can be used to estimate the maximum potential relative speedup by taking the inverse of the sequential portion of the parallel problem. According to

Amdahl's law, a task with 1% sequential operations could obtain no more than a speedup of 100, regardless of the number of processors applied to the problem.

Definition 5 (Overhead-Based Estimates of Speedup) *Amdahl's Law gives one preliminary estimate of the potential speedup in a concurrent algorithm, however, for some concurrent algorithms, overhead associated with the concurrent algorithm appears more critical than the inherent percentage of sequential operations in a concurrent algorithm. In these instances, the time for a parallel algorithm can be defined as*

$$T_p \equiv \frac{T_{seq}}{p}(1 + f_t). \tag{9}$$

Consequently, an estimate of fair speedup can be obtained by

$$\hat{S}_p = \frac{p}{1 + f_t}. \tag{10}$$

In this formula f_t is the total amount of overhead from numerous sources [16]*, that include:*

1. *nonoptimal algorithm or algorithmic overhead — additional calculations in the concurrent algorithm that are not present in a sequential algorithm*
2. *load balancing — speedup is limited by the processing time of the slowest node*
3. *software overhead — additional calculations that must be replicated at each processor, such as additional index calculations*
4. *communications overhead — idle time for processors as they wait for interprocessor communications that has not been overlapped with calculations*

It is not always easy to predict the amount of overhead in a parallel algorithm without extensive, detailed simulation. This measure of concurrent algorithm performance, along with Amdahl's law, provide preliminary estimates of potential performance improvement when comparing algorithms with differing communications properties or when comparing substantially different concurrent and sequential algorithms.

The two techniques available to the computational scientist to address this problem can thus be restated as:

1. execute as many instructions as possible concurrently

2. reduce the number of instructions to solve the problem

Both of these techniques have been addressed in previous publications on research into transient stability simulations, nevertheless, significant areas of application dependent research still exist. Numerous papers exist that address the use of vector or parallel processing architectures for the transient stability problem [2, 8, 10, 12, 13, 15, 25, 33, 34, 35, 40, 41, 44, 47]. Many papers in this field addressed theoretical applications of parallelism within the transient stability problem but offered no implementations [2, 8, 13, 15, 33, 34, 35]. Many recent papers address specific numerical analysis techniques and only a few papers have benchmark information from actual parallel processing hardware, and that hardware is often presently outdated [10, 25, 47]. Frequently those authors who have developed parallel processing implementations have relied on optimizing or automatic parallelizing compilers that were used with existing sequential transient stability software [10, 25, 47].

Meanwhile, algorithmic speedup research has been frequently reported in the IEEE Transaction on Power Systems, with only a single recent paper addressing the combination of algorithmic speedup in conjunction with parallelism [47]. Some parallel processing papers have addressed the synergism of algorithms with the computer architecture, however, the focus has been generally limited to techniques that permit parallelism by modifying the precedence when solving the DAEs at the normally sequential time-steps [2, 10, 33, 34, 35]. This *parallel time-domain* technique [2] technique has been widely used because it may have more calculations that can be performed in parallel than would a concurrent version of a sequential transient stability simulation and this *parallel time-domain* technique has permitted implementations that avoid addressing parallel implementations of such algorithms as the direct solution of linear systems of equations [10]. Nevertheless, the *parallel time-domain* technique requires significant additional calculations to be performed to enhance parallelism.

The example in [2], that illustrates the *parallel time-domain* technique, requires nearly 35% more calculations than solving the DAEs for sequential time-steps. These additional calculations are a result of reordering the lower bi-diagonal matrix that represents the application of the trapezoidal rule on the entire set of time-steps. Reordering the matrix modifies the precedence between calculations and causes fillin

that is not present in the purely sequential algorithm. Such additional calculations contribute to *algorithmic overhead* — or additional calculations that a parallel algorithm must perform that a sequential program does not perform. Any overhead specific to a concurrent implementation of an algorithm decreases the potential efficiency of that algorithm. For example, when 33% more calculations are required in a parallel algorithm, according to the overhead based estimate of speedup, the maximum speedup will be less than $(75\% * P)$, where P is the number of processors. *In spite of having to perform significant additional calculations, techniques like this are one method to utilize more parallel processors at a time.* Techniques that require additional calculations are an inefficient use of resources if other techniques exist that can be exploited for parallelism without the requirement for additional calculations.

Additionally, the *parallel time-domain* technique is inherently limited to using a low order numerical integration technique, the trapezoidal rule, that may require shorter time steps and consequently more calculations due to accuracy limitations [20]. Because the power system transient stability problem primarily involves the solution of DAEs, additional discussion on DAE solvers will be included in the next section.

One of the interesting areas for algorithmic speedup in transient stability simulations is the partitioning of the electrical network under test into study and external areas that permit larger integration step sizes in external areas. Variable timesteps can be used because the generator equations are better behaved the further the electromagnetic distance from the initial fault. Such techniques are reported in [6, 28, 30, 47], with a discussion of a parallel implementation on an Alliant FX/8 shared-memory multiprocessor being reported in [47]. That article concludes that an algorithmic speedup of approximately five is possible by simply limiting this source of unnecessary calculations. Significant improvements in both reducing the number of computer instructions and executing as many instructions in parallel on distributed-memory multiprocessors for the transient stability problem will require research into many areas not previously addressed in the state-of-the-art parallel power system transient stability analysis research described in the literature.

4 Differential-Algebraic Equation Solvers for Power System Transient Stability Simulations

A transient stability simulation is composed of three major software components:

1. the user interface
2. the DAE solver, and
3. analytical software to classify network stability.

with the vast majority of machine cycles being utilized by the DAE solver. Numerous general numerical analysis techniques have been addressed in the power systems literature that can be applied to the solution of the DAEs [13]. This list of techniques include:

1. iterated timing analysis
2. time-domain parallelism techniques
3. waveform Newton techniques
4. waveform relaxation techniques, and
5. network analysis software similar to that used in"SPICE".

Each of these techniques requires additional numerical analysis techniques such as:

1. numerical integration techniques using the trapezoidal rule
2. solutions of non-linear equations by Newton's methods
3. solutions of non-linear equations by relaxation methods
4. solutions of non-linear equations by Picard iteration
5. solutions of linear equations by direct methods, and
6. solutions of linear equations by iterative methods.

There have been several competing recurring general techniques used to solve the DAEs. [3] describes two techniques for solving the transient stability DAEs that are indicative of the aforementioned list:

1. the partitioned approach — where at each time step, the generator differential equations are solved independently to obtain estimates of generator and load currents, then the non-linear algebraic equations are solved to get an improved update for an estimate of the generator voltages at the sample time.

2. the linearization approach — where the generator equations and network equations are linearized, according to Newton's method, at a recent point on the solution trajectory by taking numerical differences and the linear algebraic equations are then solved by appropriate techniques.

The iterated timing analysis is an application of the partitioned approach, while waveform Newton techniques are an instance of a linearization approach.

These numerical DAE solution techniques have been applied to the transient stability problem, although, there has been no explicit mention in the power systems transient stability analysis literature of applications using public domain DAE solvers. These numerical analysis techniques are based on the following concept: *to avoid numerical differentiations, instead perform analytical differentiations of the given equations until they can be represented as a system of explicit differential equations* [21]. The reason for avoiding numerical differentiation is to reduce errors when taking numerical differences, and the reduction of errors in the numerical techniques is a critical factor that yields compound dividends. It is possible that the reduction in computational errors may permit trade-offs by permitting a reduction in the number of required state variables or generator equations, and trade-offs may be possible that attempt to keep numerical error constant while permitting an increase in time-step interval.

DAE solvers from the numerical analysis community often use higher order implicit Runga-Kutta integration techniques that yield more accurate solutions or permit larger time-steps. DAE solvers exist that have been explicitly developed to handle discontinuities in functions — a condition encountered in transient stability calculations. Much research is possible to determine the applicability of various DAE solution techniques to both algorithmic speedup and parallel processing speedup.

Only recently has the numerical analysis community provided various public domain software packages that utilize specialized techniques for the numerical solution of initial-value DAE problems [9, 21, 49]. There is the distinct possibility that public domain DAE solvers could significantly speedup the concurrent solution of the transient stability differential-algebraic equations in a manner similar to a petrochemical engineering application [37]. There are numerous competing techniques reported in the literature to solve DAEs:

1. multistep backward differentiation formulas (BDF) techniques
2. extrapolation techniques
3. Runga-Kutta techniques, and
4. Rosenbrock techniques.

This list is not intended to be all inclusive, nevertheless, it does illustrate a rich, untapped source of numerical analysis techniques that could be utilized to improve the performance of transient stability simulations on parallel architectures.

Various DAE solvers are available from the numerical analysis community, for example:

1. DASSL — a multistep, backward differentiation formulas (BDF) technique [9]
2. LIMEX — an extrapolation technique [9]
3. LSODI — a BDF technique [9]
4. RADAU5 — a multistep, implicit Runga-Kutta (IRK) based technique [21]
5. RODAS — a Rosenbrock technique [21], and
6. SEULEX — an extrapolation technique [21].

These DAE solvers offer the promise of many benefits that can be utilized by parallel implementations of transient stability analysis. There are numerous versions of DASSL that have been designed for specific purposes, including a concurrent version, CDASSL [37], which utilizes a concurrent, direct, non-symmetric sparse matrix solver. There is also a variant of DASSL, DASRT, that has root finding capabilities to locate

discontinuities when they are sufficiently large that DASSL cannot integrate through without intervention [24]. Such a capability could be important for transient stability simulations because some of the physical phenomena involved may be discontinuous [3]. IRK techniques may offer potential advantages to the transient stability analysis of power systems, because the generator ODEs may be stiff equations, with eigenvalues lying close to the imaginary axis [41]. This phenomena requires high order A-stable or nearly A-stable integration formulas and may benefit from codes such as RADAU5 [21].

Rosenbrock methods have advantages over IRK-based methods because they completely avoid non-linear systems of equations while providing the advantages of accurate solutions with stiff differential equations. However, to use Rosenbrock methods, the DAEs must be of index one and expressible in semi-explicit form [21]. This technique is implemented in the software RODAS. Extrapolation techniques utilize linearly implicit Euler methods and can be excellent choices when strict tolerances are required for implicit index one DAEs. LIMEX and SEULEX are implementations of this technique [9, 21]. All software discussed above is available through the Internet [21, 49].

5 Relevant Transient Stability Computational Science Research Areas

As stated earlier, there is the distinct possibility that dedicated DAE solvers could significantly speedup the solution of the equations that model the generators and power network in a transient stability analysis. The numerous techniques reported in the literature, illustrate a rich, untapped source of potential research that could be utilized to improve the performance of transient stability simulations on parallel architectures. While CDASSL, a concurrent implementation of DASSL has shown potential in a petrochemical engineering application [37], there have been no reports in the power systems literature of existing DAE solvers being applied to transient stability analysis. Research into a combination of algorithmic and parallel processing speedup based on these existing programs offers significant opportunities to improve the performance of transient stability software. Each of the aforementioned existing

programs has some feature that offers potential performance improvement. These performance improvements will be highly correlated to the particular generator and control equations.

Inherent in any of the aforementioned techniques is the solution of simultaneous systems of sparse linear equations. There is extensive research to illustrate that it is feasible to obtain reasonable parallelism and speedup when solving general sparse matrices on state-of-the-art distributed-memory multiprocessors by using direct techniques [17, 18, 19, 23, 26, 37, 43, 48], as well as by using iterative techniques [4, 11, 29, 36]. Meanwhile, there appears to be significant structure in sparse matrices encountered in power system transient stability simulations; that structure can be exploited for additional parallelism in both the portions of the matrices that represent the generator equations as well as in the portion of the matrix that represents the power network. Matrix structure can be used to improve the parallel performance of either direct or iterative methods. For parallel direct methods, matrix structure can provide additional parallelism, can reduce fillin, can minimize communications if partial pivoting is required, and can assist in developing an efficient balance that evenly distributes processing requirements and reduces interprocessor communications to a minimum while permitting the remaining communications to occur in a regular pattern. For parallel iterative methods, block-bordered diagonal matrix structure can speedup preconditioning techniques and the search for an efficient load balance with regular communications.

The transient stability DAEs are defined by a system of simultaneous equations that have special form — blocks of generator equations along the diagonals in addition to the algebraic equations representing the network admittance matrix (figure 1). After reordering the matrix, other variables are located in relatively narrow borders of the matrix [15, 41]. The generator equations naturally fall into distinct blocks on the diagonal, and the admittance matrix equations can also be placed in block-bordered diagonal form because power networks are hierarchical by nature. Matrices must be reordered for maximum parallelism, which includes considerations to minimize fillin in the sparse matrices — in order to minimize the amount of computations — while requiring optimum load balancing for the numerous parallel processors.

Significant parallelism is possible in the direct solution of the linear equations

encountered in the transient stability DAEs, because of their bordered-block diagonal form. Portions of the matrix can be reordered into a hierarchical block-bordered diagonal form and further benefit from the reduced fillin and reduced interprocessor communications for parallel implementations. For direct solutions of block-bordered diagonal matrices, significant parallelism is available in each of the three stages:

1. factorization
2. forward reduction, and
3. backward substitution.

However, a significant portion of the available parallelism in this problem is only addressable after the network is reordered into bordered-block diagonal form using either network or graph partitioning techniques. Diagonal blocks can be factorized independently — with separate blocks being factorized in parallel on separate processors or on small groups of processors. The borders are the only portion of the matrix that will require interprocessor communications, a potential source of communications overhead. Meanwhile, parallel forward reduction and parallel backward substitution can be efficient only for matrices with block-bordered diagonal form. Independent processors can work on separate portions of the matrix, with communications required only for elements in the borders. Forward reduction and backward substitution are generally considered to be sequential processes [17, 18, 19], however, the significant parallelism in these stages for transient stability simulations are a direct result of exploiting the matrix structure.

There are other research areas available in the parallel direct solution of systems of linear equations. After a matrix is reordered, there still is research possible to determine the most efficient mapping of data to particular processors and to determine whether *fan-in* or *fan-out* algorithms for the direct solution of systems of linear equations would be the most efficient for the parallel architecture of interest. Meanwhile, the balance of both calculations and communications is of continual concern.

In addition to the numerous research opportunities available with direct linear equation solvers, iterative methods exist for the solution of the linear equations. Implementation characteristics and numerical properties of iterative systems differ significantly from direct techniques to solve systems of linear equations. Conjugate

gradient methods and waveform relaxation are potential choices for iterative techniques. Iterative methods do not generate fillin and are not limited by synchronization in parallel implementations due to precedence in the calculations, however, there is no upper bound on the number of iterations required to converge to a desired accuracy. Convergence is primarily determined by the numerical behavior of the particular problem and the initial values — a set of *guesses* to start the calculations. Preconditioning techniques are often required to redefine ill-behaved matrices into more tractable ones that require significantly less iterations; previous solutions may make good initial guesses because of the small time-steps required to solve the stiff differential equations. Nevertheless, the stiff equations often have a rapid transient phase that could minimize the effectiveness of previous solutions as initial guesses in iterative techniques.

At present, we believe that direct methods offer the best possible choice for the solution of the non-linear equations inherent in the DAEs. Direct methods are robust and have well-defined processing times, which is important in real-time or faster-than-real-time software. Due to the fundamental differences in parallel implementations and the numerical properties of direct and iterative techniques to solve systems of linear equations, there are numerous research opportunities available to determine the most applicable technique to solve the systems of linear equations for an implementation of the power system transient stability analysis software on a particular parallel architecture. Also, there exists the possibility of developing hybrid techniques that utilize direct techniques to solve portions of the matrix, then use iterative techniques to update the solutions. This technique could be especially effective for symmetric portions of the matrix that could utilize parallel Choleski factorization to reduce the number of calculations.

Another research area is to combine load balancing with variable time step techniques when generators are partitioned into *near* and *far* groups that reside *inside* the study area where significant disturbances will be encountered and *outside* the study area where generators will encounter only minor disturbances. There has been some discussion of *dynamic partitioning algorithms* for parallel transient stability analysis in [47], where this technique has been implemented on a shared-memory multiprocessor, without detailed discussion of the required load balancing algorithm required to

consider the potentially unbalanced communications that would be encountered on distributed-memory multiprocessors.

Real-time or faster-than-real-time power system transient stability simulations will require significant research that may require optimizing algorithms using some or all of the aforementioned techniques to produce a scalable, architecture-dependent implementation. Scalable implementations permit an increase in speedup, and a corresponding decrease in wall-clock time, as the parallel algorithms are run on similar parallel computer architectures with greater numbers of processors. Meanwhile, power system transient stability simulations may be dependent on architecture particulars — for example, distributed-memory multiprocessors with efficient broadcast communications may benefit from sparse matrix solvers that use *fan-out* techniques, while other distributed-memory multiprocessors may benefit more from *fan-in* sparse matrix techniques.

Some state-of-the-art distributed-memory multiprocessors can utilize *Active Messages* [45] that:

1. minimize communications overhead and

2. allow communication to overlap computation.

Active Messages are implemented on the nCube 2 and CM-5 using a split-phase shared-memory extension to C, *Split-C*. [45] claims that order of magnitude improvements in message send-overhead is possible for short messages. This low overhead rate makes small messages very attractive, which is not usually the case with normal inter-processor communications models with high message start-up times. [45] also claims that matrix multiplication using *Active Messages* achieves 95% of peak performance on large nCube 2 configurations. Similar performance should be obtainable on the CM-5. In addition to the decrease in communications time, *Active Messages* foster the overlapping of communication and computation. For fan-out sparse matrix factorization, this is achieved by sending results as soon as they are calculated. Likewise for fan-in factorization, data for the next row is obtained while updating an entry. *Active Messages* have the potential to significantly improve the performance of parallel sparse matrix solvers, which in turn, has the potential to significantly improve the performance of power systems transient stability simulations. Consequently, *Ac-*

tive Messages is a significant transient stability research area that must be examined further.

There is also research potential in examining the benefits of using standards-based, reusable parallel-processing software such as Toolbox [37, 38]. Toolbox contains both numerical analysis software and high-level communications software that abstract away architecture dependencies. Numerical analysis algorithms are often a significant portion of applications software, and Toolbox provides a library of optimized versions of numerical analysis software for rapid development of efficient concurrent applications. CDASSL is an example of concurrent numerical analysis software presently available in Toolbox. CDASSL uses direct methods for solving the sparse matrices, although, it has not been optimized for the hierarchical network structures inherent in power system problems. In addition, CDASSL may have difficulties with the large discontinuities in power system problems, so other DAE solvers must be researched for this application. Nevertheless, the communications software in Toolbox can simplify implementation of new numerical analysis software, that in turn can be added to future releases of Toolbox.

The communications package in Toolbox is called Zipcode, and it has high-level commands for communications operations that are optimized for particular architectures. All Toolbox user-application software is written using the Zipcode communications routines that have similar, but optimally implemented, communications commands to simplify the migration of parallel algorithms and concurrent computer code between dissimilar architectures. Toolbox is presently implemented on the Thinking Machines Corporation CM-5, BBN TC2000, and Intel hypercubes. It will soon be modified to run on the nCube 2.

There is academic interest at NPAC in performing computational science research on the power system transient stability problem because of its stature as a computational grand challenge. Thus, there is a need to have a parallel computing testbed at NPAC that can be used to compare the performance of parallel transient stability algorithm implementations and determine whether or not the algorithms maintain sufficient accuracy to give comparable answers to known test systems [42]. We are not interested in developing new transient stability models, but in developing faster algorithms using a broad range of computational science techniques and faster parallel

implementations using algorithms that obtain the most from parallel architectures.

We are currently developing a Power System Transient Stability Testbed at NPAC, based on available software such as the EPRI-Extended Transient/Mid-Term Stability Program (ETMSP) or the PTI-Power System Simulation/E Program (PSS/E). The first target architectures for this testbed are the 32 node Thinking Machines Corporation CM5 and the 32 node nCube 2 presently available at NPAC. While the generator equations from these software packages are of primary interest to the NPAC testbed, the implementation of software that interprets the results will permit comparisons of the efficacy of new algorithms with benchmark results available in [42].

6 Estimates of Potential Speedup

The numerical analysis techniques proposed in this paper have the potential to be the basis for power system transient stability simulations that exhibit substantial speedup when compared to both present sequential programs and those parallel implementations developed for distributed-memory multiprocessors. In order to estimate potential speedup for power system transient stability simulations, estimates of the percentage of sequential calculations and concurrent overhead will be used as input respectively to Amdahl's law and overhead based estimates of speedup. The definitions for these estimators of speedup are in definition four and five.

The source of most sequential operations in the parallel algorithm will be a result of precedence in the direct solution of linear equations when solving the DAEs. If the matrix is reordered into block-bordered diagonal form, there will be significant reduction in the precedence in operations and a significant increase in the amount of available parallelism in the algorithm. We estimate that it should be possible to limit the amount of solely sequential operations to less than 5%, so, according to Amdahl's Law a maximum relative speedup of 20 could be possible for a concurrent transient stability algorithm. This estimate does not include speedup due to better transient stability algorithms. Even if the performance of *dynamic partitioning algorithms* are less than the speedup of five encountered in [47], there is the possibility of substantial speedup when combining better algorithms and concurrent processing. This is a very coarse estimate of potential performance that only considers sequential portions of

the algorithm.

Fair speedup is a often a more realistic estimator of potential performance improvement than *relative speedup* because *fair speedup* accounts for more factors that are involved with the development of parallel algorithms. Thus, estimates of potential speedup based on examining overhead in the algorithm can produce a much more realistic estimate of potential performance than does Amdahl's law. The research opportunities discussed in this paper could experience any of the four types of overhead listed in Definition 5. Often research into parallel algorithms is an attempt to examine the trade-offs between various sources of overhead, in order to minimize the aggregate overhead. Examples of overhead in the proposed research areas are presented below.

The *parallel time-domain* method [2] is an example of algorithmic overhead, or additional calculations that are performed in a parallel implementation that are not required in a sequential implementation. These additional calculations can significantly reduce potential speedup. The methods to solve the power system transient stability DAEs proposed in this paper require **no** additional calculations for the parallel versus sequential algorithms. This removes a significant source of potential overhead that erodes potential performance. Meanwhile, the proposal to examine *dynamic partitioning techniques* does introduce a source of algorithmic overhead. In a real-time parallel implementation, it would be required to examine load-balancing in real-time to minimize load balance overhead. Such calculations are not required in sequential algorithms. Meanwhile, the power systems under study are generally quite constant, so some work on load balancing for pre-chosen fault locations can be performed in advance to minimize the impact on real-time calculations.

The block-bordered diagonal form of the sparse matrix can simplify the distribution of data to processors in such a manner as to minimize communications overhead for either direct or iterative techniques to solve the sparse linear systems, even if partial pivoting is required in the direct solutions or in the preconditioning step for iterative solutions. There should be limited interprocessor communications required in the factorization of the block-bordered matrix until the narrow borders are factored. Likewise, some communications would be required for the forward reduction and backward substitution phases for the variables in the borders, however, there should be no communications required for these operations in other portions of block-bordered di-

agonal matrices. Consequently, the amount of interprocessor communications should be minimal, while required communications should be performed in a regular manner to minimize bottlenecks and should be *overlapped* with calculations when ever possible. *Active Messages* is presently available for parallel software developers to overlap communications with calculations using *Split-C* [45].

The remaining form of overhead, parallel software overhead, should be minimal as long as the granularity of calculations per processor is reasonably large. Consequently, the total amount of overhead in a concurrent transient stability simulation should be minimal. While there is no way to estimate the amount of overhead exactly without detailed simulations of the various algorithms or by actually implementing the algorithms, preliminary estimates of speedup can be developed using parametric values of overhead. We believe that aggregate overhead should be less than 20% for well-constructed concurrent algorithms. This will yield potential speedups of $(p/1.2)$, or greater than 26 for 32 processors. [47] reported a speedup of five for a transient stability *dynamic partitioning algorithm* on a shared-memory multiprocessor. Assuming that there will be 20% greater overhead for a distributed-memory multiprocessor implementation of the *dynamic partitioning algorithm*, it may be possible to obtain total speedups of as much as 100 for 32 processors when combining algorithmic and concurrent speedup. For larger multicomputers with as many as 1024 processors, it may be possible to get nearly thousand-fold speedup for power system simulations as long as the system modeled is sufficiently large. In general, only state or regional controlling authorities would have sufficiently large power system networks to efficiently use such large processors. If significant numbers of processors are utilized for each independent block in the block-bordered diagonal matrix, communications overhead could increase sharply and affect these estimates.

In concluding this discussion of potential speedup for advanced implementations of transient stability simulations, there are areas where speedup may be obtained by simply utilizing the advanced features of distributed-memory multiprocessors. Most notably, a sequential power system transient stability simulation is so large that only small portions of the data can be stored in cache or other very fast memory at any time. Partitioning the problem into smaller pieces on multiple processors many permit significantly more, and possibly all, data to be available in cache or fast memory. Due

to the size of the data structures in a sequential implementation, there will be many *cache misses*, which slow processing as data in the cache is swapped out for new data that is required in the present stage of processing. By placing the data onto multiple processors, *cache misses* can be minimized, offering significant unseen benefits to speedup the calculations.

7 Conclusions

A recent IEEE committee report [41] by a task force of the Computer and Analytical Methods Subcommittee of the Power Systems Engineering Committee states that

> *Except for those analytical procedures that require repeat solutions, like contingency analysis, there are no obvious parallelism inherent in the mathematical structure of power systems problems,*

We believe that this view of parallelism in power systems problems illustrates the need for closer coordination with the computational science research community. There are problems in other disciplines that illustrate the existence of parallelism in the solutions of differential-algebraic equations, the central component of the transient stability problem. The critical point is the granularity where parallelism exists and the level of sophisticated techniques that are required to extract that parallelism for particular parallel processing hardware.

Various computational science research topics for the transient stability problem have been proposed in this paper. The goal of this paper has been to describe various research areas where the computational science academic community can interact with the power systems engineering community to improve the quality and performance of power systems transient stability analysis simulations. Speedups of over 1000 appear possible for large multiprocessors simulating the transient stability of large interconnected power systems, while speedups of over 100 appear reasonable for individual medium-sized power utility companies. Real-time or faster-than-real-time transient stability simulations will require both highly parallel computers and better overall algorithms to get the computational speedup required for this grand computing challenge, but the benefits of improved system reliability should yield substantial

payoffs for electrical utility company profits, environmental impact, and customer satisfaction,

Acknowledgment: We thank Ernst Hairer, Alan Hindmarsh, Alvin Leung, Nancy McCracken, Linda Petzold, and Tony Skjellum for their assistance in preparing this paper.

References

[1] M. M. Adibi, P. M. Hirsch, and J. A. Jordan, Jr. Solution methods for transient and dynamic stability. *Proceedings of the IEEE*, 62(7):951–958, July 1974.

[2] F. L. Alvarado. Parallel solution of transient problems by trapezoidal integration. *IEEE Transactions on Power Apparatus and Systems*, PAS-98(3):1080–1090, May/June 1979.

[3] P. M. Anderson and B. Dembart. Computational aspects of transient stability analysis. In A. M. Erisman, K. W. Neves, and M. H. Dwarakanath, editors, *Electrical Power Problems: The Mathematical Challenge*, pages 159–189. SIAM, Philadelphia, 1980.

[4] C. Aykanat, F. Özgüner, F. Ercal, and P. Sadayappan. Iterative algorithms for solution of large sparse systems of linear equations on hypercubes. *IEEE Transactions on Computers*, 37(12):1554–1568, December 1988.

[5] A. Bass, S. Steinberg, R. Tabors, and P. DeGenring. A comprehensive study of supercomputing and its applicability to the utility industry. Technical report, TASC, Reading, MA, August 1990. Draft Final Report ESEERCO Project EP-90-10.

[6] R. Belhomme and M. Pavella. A composite electromechanical distance approach to transient stability. *IEEE Transactions on Power Systems*, 6(2):622–631, May 1991.

[7] A. R. Bergen. *Power Systems Analysis*. Prentice Hall, 1986.

[8] F. M. Brasch. Jr., J. E. Van Ness, and S. C. Kang. Simulation of a multiprocessor network for power system problems. *IEEE Transactions on Power Apparatus and Systems*, PAS-101(2):295–301, February 1982.

[9] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Elevier Science, 1989.

[10] 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*, 6(4):1539–1545, November 1991.

[11] A. T. Chronopoulos. Towards efficient parallel implementation of the cg method applied to a class of block tridiagonal linear systems. In *Supercomputing '91*, 1991.

[12] P. E. Crouch, E. Brady, and D. J. Tylavsky. Frequency domain transient stability simulation of power systems: Implementation by supercomputer. *IEEE Transactions on Power Systems*, 6(1):51–58, February 1991.

[13] M. L. Crow and M. Ilic. The parallel implementation of the waveform relaxation methods for transient stability simulations. *IEEE Transactions on Power Systems*, 5(3):922–932, August 1990.

[14] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford University Press, Oxford, 1990.

[15] J. Fong and C. Pottle. Parallel processing of power system analysis problems via simple parallel microcomputer structures. *IEEE Transactions on Power Apparatus and Systems*, PAS-97(5):1834–1841, September/October 1978.

[16] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker. *Solving Problems on Concurrent Processors.* Prentice Hall, 1988.

[17] A. George, M. T. Heath, J. Liu, and E. Ng. Solution of sparse positive definite systems on a shared-memory multiprocessor. *International Journal of Parallel Programming*, 15(4):309–328, August 1986.

[18] A. George, M. T. Heath, J. Liu, and E. Ng. Sparse cholesky factorization on a local-memory multiprocessor. *SIAM journal on Scientific and Statistical Computing*, 9(2):327–340, March 1988.

[19] A. George, M. T. Heath, J. Liu, and E. Ng. Solution of sparse positive definite systems on a hypercube. *Journal of Computational and Applied Mathematics*, 27:129–156, 1989.

[20] G. H. Golub and J. M. Ortega. *Scientific Computing and Differential Equations.* Academic Press, San Diego, CA, 1992.

[21] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II — Stiff and Differential-Algebraic Problems.* Springer-Verlag, New York, 1991.

[22] H. H. Happ. Diakoptics - the solution of system problems by tearing. *Proceedings of the IEEE*, 62(7):930–940, July 1974.

[23] M. T. Heath, E. Ng, and B. W. Peyton. Parallel algorithms for sparse linear systems. In *Parallel Algorithms for Matrix Computations*, pages 83–124. SIAM, Philadelphia, 1991.

[24] A. Hindmarsch, April 1992. personal correspondence with A. Hindmarsch.

[25] S. Y. Lee, H. D. Chiang, K. G. Lee, and B. Y. Ku. Parallel power system transient stability analysis on hypercube multiprocessors. *IEEE Transactions on Power Systems*, 6(3):1337–1343, August 1991.

[26] R. F. Lucas, T. Blank, and J. J. Tiemann. A parallel solution method for large sparse systems of equations. *IEEE Transactions on Computer-Aided Design*, CAD-6(6):981–991, November 1987.

[27] I. M. Mack. *Block Implicit One-Step Methods for Solving Smooth and Discontinuous Systems of Differential/Algebraic Equations.* PhD thesis, Harvard University, The Graduate School of Arts and Sciences, 1986.

[28] N. Müller and V. H. Quintana. A sparse eigenvalue-based approach for partitioning power networks. *IEEE Transactions on Power Systems*, 7(2):520–527, May 1992.

[29] C. Siva Ram Murthy. Parallel,iterative solution of large, sparse linear systems on hypercubes. In *IEEE ISCAS '89*, 1989.

[30] R. Nath, S. S. Lamba, and K. S. Prakasa Rao. Coherency-based system decomposition into study and external areas using weak coupling. *IEEE Transactions on Power Apparatus and Systems*, PAS-104(6):1443–1449, June 1985.

[31] E. C. Ogbuobiri, W. F. Tinney, and J. W. Walker. Sparsity-directed decomposition for gaussian elimination on matrices. *IEEE Transactions on Power Apparatus and Systems*, PAS-89(1):141–150, January 1970.

[32] P. C. Patton. Performance limits for parallel processors. In G. F. Carey, editor, *Parallel Supercomputing: Methods, Algorithms and Applications*, chapter 1. John Wiley & Sons, New York, 1989.

[33] M. La Scala, A. Bose, D. J. Tylavsky, and J. S Chai;. A highly parallel method for transient stability analysis. *IEEE Transactions on Power Systems*, 5(4):1439–1446, November 1990.

[34] M. La Scala, M. Brucoli, F. Torelli, and M. Trovato;. A gauss-jacobi-block-newton method for parallel transient stability analysis. *IEEE Transactions on Power Systems*, 5(4):1168–1177, November 1990.

[35] M. La Scala, R. Sbrizzai, and F. Torelli. A pipelined-in-time parallel algorithm for transient stability analysis. *IEEE Transactions on Power Systems*, 6(2):715–722, May 1991.

[36] J. N. Shadid and R. S. Tuminaro. Sparse iterative algorithm software for large-scale mimd machines: an initial discussion and implementation. Technical report, Sandia National Laboratories, 1991.

[37] A. Skjellum. *Concurrent Dynamic Simulation: Multicomputer Algorithms Research Applied to Ordinary Differential-Algebraic Process Systems in Chemical Engineering.* PhD thesis, California Institute of Technology, Division of Chemistry and Chemical Engineering, Pasadena, CA, 1990.

[38] A. Skjellum and C. Baldwin. The multicomputer toolbox: Scalable parallel libraries for large-scale concurrent applications. Technical report, Numerical mathematics Group, Lawrence Livermore National Laboratory, 1991.

[39] H. S. Stone. *High-Performance Computer Architecture.* Addison Wesley, Reading, MA, second edition edition, 1990.

[40] 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*, 7(1):1–7, February 1992.

[41] D. J. Tylavsky and A. Bose. Parallel processing in power systems computation. *IEEE Transactions on Power Systems*, 7(2):629–638, May 1992.

[42] Chairman V. Vittal. Transient stability test systems for direct stability methods. *IEEE Transactions on Power Systems*, 7(1):7–43, February 1992.

[43] S. Venugopal and V. K. Naik. Effects of partitioning and scheduling sparse matrix factorization on communications and load balance. NASA Contractor Report 189563 ICASE Report No. 91-80, NASA, Langley Research Center, October 1991.

[44] V. Vittal, G. M. Prabhu, and S. L. Lim. A parallel computer implementation of power system transient stability assessment using the transient energy function method. *IEEE Transactions on Power Systems*, 6(1):167–173, February 1991.

[45] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser. Active messages: a mechanism for integrated communication and computation. Technical report, Computer Science Division — EECS, University of California, Berkeley, CA, March 1992. Report No. UCB/CSD 92/#675.

[46] Y. Wallach. *Calculations and Programs For Power System Networks.* Prentice-Hall, 1986.

[47] N. Zhu and A. Bose. A dynamic partitioning scheme for parallel transient stability analysis. *IEEE Transactions on Power Systems*, 7(2):940–946, May 1992.

[48] M. Zubair and M. Ghose. A performance study of sparse cholesky factorization on intel ipsc/860. NASA Contractor Report 189634 ICASE Report No. 92-13, NASA, Langley Research Center, March 1992.

[49] D. Zwillinger. *Handbook of Differential Equations.* Academic Press, Boston, second edition edition, 1992.