Evaluation of Multiprocessor Algorithms for Transient Stability Problems

EL-947
Technical Planning Study 77-718

Final Report, November 1978

Prepared by

NORTHWESTERN UNIVERSITY
Electrical Engineering Department
2145 North Sheridan Road
Evanston, Illinois 60201

Principal Investigators
F. M. Brasch, Jr.
J. E. Van Ness
Sang-Chul Kang

Prepared for

Electric Power Research Institute
3412 Hillview Avenue
Palo Alto, California 94304

EPRI Project Manager
John Lamont
Electrical Systems Division

EPRI PERSPECTIVE

PROJECT DESCRIPTION

One of the developing concepts in digital computer applications is "parallel processing." This term is used to describe a variety of computers and analytic methods that carefully organize computations for faster implementation. Some machines have parallel computing elements that can perform elementary operations (+,-,x,÷) simultaneously. Others use a pipeline approach wherein operations are queued for execution, resulting in improved overall efficiency.

This study had the goal of exploring the use of distinct parallel microcomputers to perform complex power system stability calculations. Basically, the idea is to distribute the computing load to a number of separate processors that work independently. In doing this, a number of questions arise regarding the problem decomposition, the preferred number of parallel microcomputers, and the communications among the computers.

The timing of this study was based on the recent availability of microcomputers with hardware multiply/divide capability. This feature was considered essential for the transient stability problem under investigation. The study's scope included an assessment of the ways in which these small computers might be used to solve significant problems, such as transient stability, in a way that will improve the time and cost of computation. The contractor used the transient stability program of the Bonneville Power Administration as a basis for the system model and computational methods. This computer code was examined to determine how the computational task could be performed in parallel. The design parameters of the parallel computation were also studied.

PROJECT OBJECTIVES

This study investigated the feasibility of constructing a special purpose parallel microcomputer network to solve the transient stability equations. Alternative computer architectures were evaluated to determine the best number and arrangement of microcomputers. Methods of controlling the computation to optimize the use of computing hardware were investigated.

A number of hardware, time, and cost trade-offs will require analysis. It is not too surprising to find that two computers are not quite twice as good as one, nor are ten twice as good as five. This saturation effect is an important limitation to be overcome or optimized. This problem will almost certainly require more extensive analysis, and could become the ultimate bottleneck for efficient parallel computation of the type described here.

Another problem that was only briefly addressed is that of optimizing the scheduling of computing tasks for each microcomputer. There are many ways to do this scheduling, and the best method may require considerable future effort.

CONCLUSIONS AND RECOMMENDATIONS

Based on the preliminary results obtained from this study, it appears that a parallel micromachine can be designed that will show significant improvement in the speed and cost of transient stability calculations. The hardware required is just becoming available and future technologies look bright. The problem will be in developing a design that will use a number of processors effectively.

This study is being continued under a new research project, RP1355, "Design of Multiprocessor Structure for Simulation of Power System Dynamics."

John Lamont, Project Manager
Electrical Systems Division

CONTENTS

ILLUSTRATIONS

TABLES

SUMMARY

This project has investigated the possibility of building a dedicated network of microprocessors to solve the step-by-step computations necessary in the dynamic simulation of a large power system. This multiprocessor network would function as a special purpose peripheral for use in conjunction with a general purpose digital computer. The host processor would handle the data preparation tasks as well as the presentation of the results at the end of the computation. Only the time consuming and expensive step-by-step computation would be performed by the multiprocessor network. In this way, the great effort that has been put into establishing data bases and other input/output facilities in the present stability programs would not have to be repeated, while the advantages of the modern technology that are available in the microprocessor chips could be used to reduce the cost and possibly the time required to do these dynamic simulation studies.

In order to evaluate the gains made by using a multiprocessor network, we have based this study on the Bonneville Power Administration's transient stability program using data from their actual studies. The largest of the sample problems that was used contained 1723 buses with 398 generators. Most of the machines were represented in considerable detail, so that this would be a time consuming system to simulate on a general purpose computer. The actual step-by-step computations were divided into separate functions so that they could be accurately timed and evaluated for parallel computation. On the largest sample system that was examined over 97% of the computations in the step-by-step portion of the solution can be done in parallel.

There are two major groups of computations that are necessary. The first involves the equations for the machines together with the associated voltage regulators and governors. The second is the solution of the network equations. Since the couplings between the individual generators are through the network equations, those computations involving the generating units can be easily scheduled to be done in parallel. The major problems in parallel processing occur with the solution of the network equations.

Two major methods of solving the network equations were studied. The first is called the sequential method and involves solving the equations for the nodes in sequence. It has the advantage of being universally applicable to any network, but it has the disadvantage of not taking full advantage of the sparsity that exists in the power system network matrix. The second method investigated is called the bordered block diagonal form, or BBDF, and is based on clustering adjacent nodes together for the solution. This method depends heavily on having a sparse matrix for the network equations.

Both of these methods show a saturation effect as far as the number of processors is concerned. Up to a certain level, increasing the number of processors available in the multiprocessor network brings about an increase in the amount of parallelism being exploited by the multiprocessor network. But, after a certain point, the additional processors are not fully utilized, and there is a loss of effectiveness for the network. While there still are significant gains to be realized by using a multiprocessor network, there is a ceiling on the number of processors that can be used effectively.

A third approach to this problem that we have just begun to investigate involves a scheduling algorithm to be used in assigning tasks to the processors in the network. The solution of the network equations can be started before the computations for all of the generating units have been completed. The same sort of overlaps can be utilized as the network solution is being completed. With the 1723 bus system, the solution of the network equations takes approximately one-third of the solution time. By interleaving these computations with the other two-thirds which do not interact and thus are easy to schedule in parallel, we hope to be able to overcome some of the limitations brought about by the saturation effect. An optimum solution to this scheduling problem is extremely difficult to obtain, but we believe some heuristic methods can be developed that give sufficiently good solutions.

Based on a thorough understanding of the algorithm to be used for this problem, we believe that a microprocessor network can be designed for this simulation which will show great cost-effectiveness advantages over the present methods using general purpose digital computers. The hardware needed for this is just now becoming available, with even better components expected in the near future.

Section 1

INTRODUCTION

The purpose of this research project has been to investigate the possibility of using a multiprocessor computer network in the dynamic simulation of power systems. The step-by-step computations that must be performed in solving the large set of differential and algebraic equations used to describe the characteristics of a power system, such as in a transient stability study, are very lengthy and expensive on even the largest and fastest of the general purpose digital computers. With the advent of relatively inexpensive and small digital processors, it may be possible to build dedicated networks of these processors which could do the step-by-step calculations in parallel. The hope is that these multiprocessor networks would be much more cost effective and possibly even faster than the general purpose computers used now.

In all of our work we have assumed that it is best to exploit the macro-parallelism that exists in the power system network in formulating the solutions to this problem. The alternative would be to use a micro-parallelism which might parallel computing actions at the instruction level. However, an actual power system will have many generating plants operating in parallel with different individual characteristics. Not only will the values of the parameters differ, but the form of the mathematical model used for their description will depend on the age and the type of the generating unit involved. At the level of parallelism that we are considering in this problem, these different models can be handled by individual processors in parallel even though the order and the form of the models may be quite different.

We envision a general purpose digital computer being used to maintain the large data base required for this problem and to set up specific cases to be integrated. Interfaced with this computer would be a network of processors, probably made from microprocessor chips, which would do the actual step-by-step integration of the equations. The results would be returned to the general purpose computer for preparation of the output in printed and graphical form as needed by the users. We thus are proposing a network of computer processors dedicated to power system stability calculations. We believe that the economies are such as to easily justify this approach if satisfactory algorithms and computer network configurations can be developed.

A parallel processing workshop (1) was held by EPRI in October of 1977 at which many aspects of and approaches to this problem were presented. In addition, the literature is rich with articles and books on the subject of the transient stability problem dating back over many, many years. We are not going to attempt to summarize the excellent work that has been done in reaching our present understanding of this problem. A recent paper by Stott (2) is an excellent review of the numerical methods used in the digital calculation of power system dynamic response. We refer you to this paper for a discussion of the methods currently used for solving this problem and the details of the problem itself. An earlier paper by Dommel and Sato (3) forms a basic reference for the implicit integration schemes discussed by Stott and used by us in this project. A great number of other excellent papers are referenced by Stott in his survey article on this subject.

Section 2

NUMERICAL METHOD OF SOLUTION

2.1 CHOICE OF THE INTEGRATION ALGORITHM

One of the first questions to be considered is the choice of the integration algorithm. Not only are there several widely used production programs for the transient stability problem based on differing algorithms, but these can be further classified as to how they might be executed in a parallel mode. At the beginning of this project, we tried a simulation of the parallel solution of the basic netowrk equations using an algorithm based on the near block diagonal form (NBDF) for the equations. It became obvious that this method had the same convergence problems as the Gauss-Seidel method of solving the load flow problem. But more important, it also was apparent that there would be many difficulties if we attempted to develop our own method of solving the transient stability problem in a parallel mode.

The first problem was that a tremendous programming effort would be needed to develop a program capable of solving problems of the size and complexity that normally occur on power systems. Since this would be duplicating the effort, often extending over many years, that has been put into the development of many of the present production codes, it did not seem to be a justifiable use of our research efforts.

A second consideration was that we desired to evaluate the advantages to be gained by using parallel processing on a multiprocessor network. To do this, we would like to remove all variables from the test which are a function of the integration algorithm being used. If a standard production program could be used for the basis of our work, together with a realistic test problem, we would be able to make direct comparisons between our proposals and the methods actually used, thus avoiding questions that might come up as to whether any improvements seen were a result of the parallel processing or due to some simplification of the problem or the method in the test cases.

We have chosen to base our work on the production code used by the Bonneville Power Administration. In addition to the advantages listed above for using a standard production code, there was a distinct additional advantage in this

choice for us in that both BPA and Northwestern University use the same type of computer: BPA has a CDC 6400, while Northwestern has a CDC 6600. We were able to mount their tapes and get their programs running on our computer with a minimum of effort. They also provided us with two large sample problems which will be described in more detail later in this report. These programs and the sample problems that came from BPA will form the basis for our study of the advantages of parallel processing for transient stability studies.

At some time in the future, we would like to examine some different type of algorithm for this problem to compare it with the BPA program as far as the advantages of parallel processing are concerned, but our present efforts have been concerned with this one approach.

2.2 DESCRIPTION OF THE BPA SOLUTION METHOD

The BPA program is based on the method described by Dommel and Sato (3). It uses the trapezoidal rule to change the differential equations into difference equations which are then solved iteratively with the network equations. The basic computation is shown in Figure 2-1. The first block represents the solution of the difference equations obtained from the differential equations describing the system. Included here are the equations for the inertia of the machine, for the excitation system and supplementary controls, and for the governor and turbine. These equations will be local for each generating station and easily can be handled in parallel. The second block represents the computation of the injection currents at each of the generators. These computations again are independent for each of the generators and can be done in parallel. Concurrent with this is the computation of injection currents for nonstandard loads and the injection currents representing dc transmission lines. Following this comes the solution of the network equations.

The network is represented by the standard node pair equations

$$I = YE \tag{2-1}$$

The I's have been computed as the injection currents in the previous steps. The Y matrix is factored into an L, D and U which do not change except when switching occurs on the network. Thus, during the normal step-by-step solution, except at switching times, all that is needed is to make a forward substitution through the L matrix, a division by the D terms, and a back substitution through the U matrix to solve for the voltages on the network. The ways in which this might be done in parallel will form a major part of the discussion later in this report. Following the solution of the network equations, the power at each of the nodes is computed. These four blocks form a computational loop.

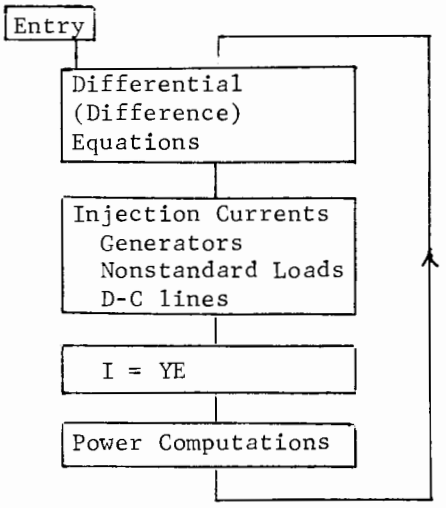


Figure 2-1. Basic loop for iterative solution of one time step

At the start of any time step, this loop is entered at the top box and the necessary state variables are extrapolated from previous values for the solution of the difference equations. Various test criteria are used throughout the loop and, when the solution has met these tests, the computations are terminated for that time step and the overall procedure moves on to the next time step.

From a parallel computation point of view, this computation is ideal. All of the major blocks in Figure 1, except the solution of I = YE, are independent for each of their nodes and can be easily paralleled. As will be discussed later in this report, there seems to be some possibility in scheduling parallel processors to even overlap some of these computations with those necessary for I = YE. With proper scheduling, this should even further increase the parallelism available for the solution of this problem.

Section 3

THE SAMPLE PROBLEMS

The larger of the two major sample problems provided to us by BPA is typical of the production runs that they make on their system. The electrical network has 1723 buses and there are 398 generators connected to the network. Of those, 329 generators have exciters and are represented with saliency. There are many types of exciters, many of the machines have governor action, have saturation represented and include supplementary control systems on the network. Eight of the transmission lines can have local relay action. There are nonlinear loads and dc transmission. We are very enthusiastic about this sample problem. It seems to include all of the features that can cause great difficulty in solving this type of problem and which are often left out in the sample systems that those of us in the academic world generate on our own. We intend to use this problem as a major benchmark in the testing of our methods.

The second sample problem that they sent us has 1199 buses with 230 machines connected to the network. However, only 17 of these machines have exciters and are represented in any detail. This problem was made up to test their program and, while it does contain most of the major features used in the program, it is not typical of their production runs. We are running it for comparison, but tend to place greater importance on the 1723 bus system.

3.1 EXECUTION TIMES

The BPA program is written as a series of overlays. At least seven overlays are used for preparation of the data and setting up the necessary tables. Several more are used for printing and plotting the output in the desired form. We are doing nothing with these sections of the program. We are concentrating on the overlay that does the step-by-step computation of the response. This is where the major time is spent in the solution and is the part that can be successfully executed on the parallel processor network. We envision using a general purpose computer such as our CDC 6600 to prepare the data for the solution and to take the results and prepare the desired output. The parallel processor network would only do the step-by-step solution. The BPA program in its overlay format is structured for exactly this type of execution.

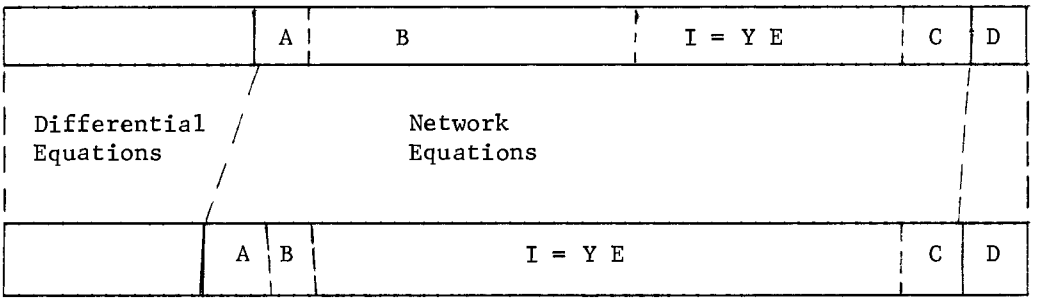
The results from a typical integration step are listed for each sample problem in Table I. The times given there were measured on our CDC 6600 using its internal clock. The results are also shown graphically in Figure 3-1.

TABLE 3-1

Problem description:				
Number of Buses	1723		1199	
Number of Machines	398		230	
Number of Exciters	329		17	
	Seconds	%	Seconds	%
Solution times per step:				
Differential Equations	5.7	29.7	0.90	18.8
Network Equations:				
Machine Currents	1.0	5.1	0.21	4.3
Load & dc Currents	4.6	24.1	0.22	4.6
I=YE, Forward & Back	6.3	32.8	2.99	62.6
Power Computation	1.1	5.5	0.20	4.3
Other Computations	0.5	2.8	0.25	5.2
Total Time	19.2	100.	4.77	100.
Time Definitely in Parallel	18.7	97.2	4.52	94.5

BPA SWING Program

1723 bus problem



1199 bus problem

A - Machine Currents

B - Load and dc Currents

C - Power Computation

D - Other Computations

All sections except D can be executed in parallel.

Figure 3-1

The biggest single computation shown in Table 3-1 is the solution of the network equations, I=YE. However, in the 1723 bus production program these computations take less than one-third of the total computation time. In the BPA program this time represents only the forward and back substitution through the LDU factors of the Y matrix. The Y matrix is refactored only at times when switching or some other discontinuity occurs on the system. The factoring time is not included in the results presented in Table 3-1 or in Figure 3-1.

All of the computation steps listed under differential equations and network equations in Table 3-1 can thus be done in parallel. The only part that cannot is listed under "Other Computations" in this table. Thus we see for the 1723 bus system, 97.2% of the computations can definitely be done in parallel. For the 1199 bus system, 94.5% of the computations can be done in parallel. Figure 3-1 shows the results of Table 3-1 in graphical form on a percentage basis. While the division of work between the various sections is quite different for the two types of problems, it is only the block labelled D at the right of each graph that cannot be done in parallel. This shows that we have a great opportunity to take advantage of parallel computation in this type of problem.

Section 4

SOLUTION OF NETWORK EQUATIONS

Many people have looked at the problem of solving linear equations with a multiprocessor. An excellent reference on the subject is the proceedings of the EPRI workshop (1) mentioned earlier. The choice of the BPA program as a basis for the step-by-step solution algorithm changes the emphasis of our investigation from that of most previous workers. The BPA approach is to structure the simulation equations so that the network equations need be factored only at switching times. Thus our work will discuss only the forward and back substitution process for a given triangularized matrix.

Following standard procedure, the admittance matrix Y is factored:

$$Y = LDU \tag{4-1}$$

where L is a unit lower triangular matrix , D a diagonal matrix, and U a unit upper triangular matrix. The solution process that we wish to accomplish in parallel is then achieved by solving three equations

$$LX = I \tag{4-2}$$

$$DZ = X \tag{4-3}$$

$$UE = Z \tag{4-4}$$

We would like to compare the total execution time, T, for different matrix structures, assuming unity execution time for each elementary accumulation operation, a multiplication followed by an addition. We will also ignore the time required to establish data communication between processors. We feel this is not a bad assumption for several reasons. First, we are interested in comparing several solution methods, not in obtaining absolute performance data at this stage. It appears that communication times will scale with operation count so that relative performance figures will remain accurate. Second, the algorithms that we will look at are so highly structured that we expect bus contention phenomena to be minimal or non-existent. Our earlier simulation of a non-structured solution technique, essentially a multi-processor version of Gauss-Seidel, revealed that contention was a negligible problem for equations as sparse as those of power systems, so we expect little performance degradation from this complication.

Now $T = T_L + T_D + T_U$ where

T_L is the time required to solve (4-2)

T_D is the time required to solve (4-3)

T_U is the time required to solve (4-4)

Since D is a diagonal matrix, the solution of (4-3) will be obtained by n divisions which can be carried out independently. Here n is the size of the admittance matrix. As such, with m processors the solution will be obtained in the smallest integer greater than or equal to n/m time units, where m is the number of processors used. As a notational convenience henceforth time units will be abbreviated as t.u., and the symbol $\lceil x \rceil$ will be used to mean the smallest integer greater than or equal to x.

Since the same method of analyzing the operations needed can be used for both the forward and back substitution processes, results obtainable from a detailed examination of the solution of equation (4-2) will be applicable to equation (4-4) also. Therefore, as a figure of merit we will concentrate on T_L.

In this section of the report we will examine in detail two solution schemes. The first is derived from the parallel row ordered elimination scheme of Wong (4) and will be referred to as the sequential method. This method can be applied to any triangular matrix. It will be shown to have a potential drawback in that it does not really take advantage of the sparsity that appears in power system admittance equations. The second method does try to take advantage of sparsity in a very straightforward way. The idea is to reorder the equations into a particular form, named the Bordered Block Diagonal Form, or BBDF for short. The BBDF ordering appears to greatly facilitate the parallel solution of linear equations and is described in detail in (5,6). Both algorithms will be assumed to execute on a structure like that of Figure 4-1 where communication between processors is over a single common bus of suitably high communication capacity. Our goal in this analysis will be to obtain the theoretical limits of performance of each of these methods and to quantify a saturation phenomena which they both exhibit.

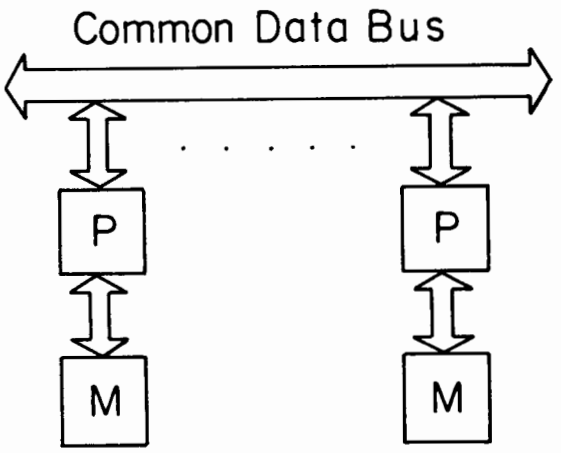


Figure 4-1. Multiprocessor Structure

4.1 SEQUENTIAL METHOD

Let us suppose that we have m processors interconnected by a common bus as in Figure 4-1. The sequential method then presupposes that the non-zero elements of any one row of the L, D, and U matrices as well as the ith row of I is stored in the local memory of a single one of the m processors. The forward substitution process can then be carried out as follows.

First, for each row of L for which a given processor is responsible, a temporary location in that processors' local memory must be initialized to the value of current associated with that row. The processor responsible for the first row can distribute the value of x_1, the first component of the solution vector x, because it is simply I_1. All processors now update their temporary locations, denoted t_i, as follows:

$$t_i = t_i - L_{i1}I_1 \qquad (4\text{-}5)$$

for i = 2, 3,..., n. The processor holding t_2 is now finished with the second row and can distribute $x_2 = t_2$ over the common bus. After receiving that element all processors update the remaining temporary locations and the procedure continues until all n components of x have been computed. A similar process clearly works for U.

If it is the case that m=n, the solution of (4-2) can be obtained in n-1 t. u., which is the minimum possible with the sequential method. Assuming that the rows of the L matrix are evenly distributed among all the processors so that the non-zero elements in each column of the L matrix are essentially equally distribu-