

Parallel processing in dynamic simulation of power systems

ANJAN BOSE

Department of Electrical Engineering, Arizona State University, Tempe, Arizona 85287, USA

Present address: School of Electrical Engineering & Computer Science, Washington State University, Pullman, WA 99164–2752, USA

Abstract. The dynamic behaviour of a large interconnected electric power system is characterized by a simultaneous set of nonlinear algebraic and ordinary differential equations. The solution is obtained by numerical methods and the simulation of the transient behaviour for a few seconds after a fault is the standard analytical procedure used in planning and operational studies of the system. The need for on-line simulation in near real time for more efficient operation has encouraged the search for faster solution methods and the use of parallel computers for this purpose has attracted the attention of many researchers. The success of parallelization depends on three factors: the problem structure, the computer architecture, and the algorithm that takes maximum advantage of both. In this problem, the generator equations are only coupled through the electrical network providing some parallelization in (variable) space, and a solution is needed at each time step leading to some parallelization in time (waveform relaxation). However, since the problem formulation is not completely decoupled, parallel algorithms can only be developed by trading off any relaxation with a degradation in convergence. The fastest sequential algorithm used today is the combination of implicit trapezoidal integration with a dishonest Newton solution. The Newton algorithm is not parallel at all but has the fastest convergence while a Gauss–Jacobi algorithm is completely parallel but converges very slowly. A relaxation of the Newton algorithm appears to be a good compromise. As for the parallel hardware, the coupling seems to require significant communication between processors thus favouring a data-sharing architecture over a message-passing hypercube. Special architectures to match the problem structure have also been an area of investigation. This paper elaborates on the above issues and assesses the present state-of-the-art.

Keywords. Power systems; transient stability; parallel computation; parallel algorithms.

1. Introduction

The simulation of power system dynamics is a standard analytical procedure used in the planning and operation of power systems. The large number of differential-

algebraic equations and their nonlinearities that represent power system dynamics make time simulation the only viable alternative for such extensive engineering analysis. Production grade programs have been developed over the years to make this tool support various analytical requirements. Experience has also provided comparisons of various algorithms. Since these simulations are computationally expensive, and hence time consuming, the search for more efficient algorithms continues to this day. If such analysis can be done faster than real time, its use can be extended from purely analytical studies to on-line control applications.

The development of algorithms over the last three decades makes it less likely that major breakthroughs will dramatically change simulation speed. However, computer hardware improvements are making faster processors available at lower prices, thus speeding up the solution time for the same algorithm. This trend in the processor speed as well as the trend to develop multiprocessor computers are raising expectations of being able to conduct such simulation in the on-line environment. To take full advantage of the multiprocessor architectures the present algorithms may not be the most efficient because they were particularly optimized for the uniprocessor environment. Thus a need to reexamine the algorithms for parallelism arose as parallel computers became a reality.

New parallel algorithms for power system dynamics are needed to take advantage of the new parallel architectures. This task is made difficult by the situation that multiprocessor architecture design is still fluid. In fact, it is possible to design architectures to suit particular problem structures but such customized designs are usually not commercially viable alternatives. It is more likely that some generic architectures will evolve as the best alternatives and algorithms will have to be optimized for them.

In this paper, we examine the structure of the power system dynamics problem and the best known sequential algorithm, which uses an explicit integration and a modified Newton solution. Several types of parallel algorithms, Gauss–Jacobi and relaxed Newton, are then applied to the problem. These algorithms are implemented on two different types of multiprocessor machines that are commercially available today. Results on a realistic power system problem show that a mildly relaxed version of the present Newton method provides the most speedup.

It should be mentioned that the work presented here is not meant to be conclusive. Anyone with some experience in the application of algorithms to complex problems can verify that variations of these algorithms are almost infinite and some of them could prove to be more effective. Also, the variations in multiprocessor architectures and their compilers can have significant effect on the performance of a particular algorithm. In that respect this work has just scratched the surface of a very large area of investigation. We hope, however, that the combination of the theory and implementation in this paper will leave the reader with a clearer impression of the different levels of complexity of the problem, and also provide some insight on some methodology to investigate this problem.

2. Problem formulation

2.1 Model

In an interconnected power system, the complete description of the system models

consists of a set of nonlinear differential equations:

$$\dot{x} = f(X, V), \tag{1}$$

and a set of algebraic equations:

$$I - Y(X) * V = 0, \tag{2}$$

where X are the generator variables that describe the dynamics of the system and V are the node voltages of the network. There are two basic integration approaches, the explicit and the implicit. The latter is preferred because it provides better stability and can avoid the difficulty of stiff problems. Hence, (1) can be discretized by the trapezoidal rule and rearranged as:

$$R_G = X_t - X_{t-1} - (h/2)\{f(X_t, V_t) + f(X_{t-1}, V_{t-1})\} = 0, \tag{3}$$

$$R_N = I(X_t, V_t) - Y(X_t) V_t = 0, \tag{4}$$

where $t = 1, 2, \dots, T$, denotes time steps, while h represents the step length. The structures of (3) and (4) can be seen more clearly in their linearized forms:

$$\begin{bmatrix} R_G \\ R_N \end{bmatrix} = - \begin{bmatrix} A_G & B \\ C & Y + Y_{ld} \end{bmatrix} \begin{bmatrix} \Delta X \\ \Delta V \end{bmatrix} = -J \begin{bmatrix} \Delta X \\ \Delta V \end{bmatrix} \tag{5}$$

where

$$A_G = \partial R_G / \partial x, \quad B = \partial R_G / \partial V, \tag{6}$$

$$C = \partial R_N / \partial X, \quad J_N = Y + Y_{ld} = \partial R_N / \partial V.$$

A more detailed Jacobian can be shown as:

$$J = \begin{bmatrix} A_{G1} & & & B_1 \\ & \ddots & & \vdots \\ & & A_{Gi} & B_i \\ & & & \vdots \\ & & & A_{Gm} & B_m \\ \hline C_1 & \cdot & C_i & \cdot & C_m & | & J_N = Y + Y_{ld} \end{bmatrix} \tag{7}$$

A_{Gi} is a square matrix with dimension q ranging from 2 for the classical model to 20 for the detailed representation including exciter, governor, stabilizer and prime mover. Y is the network admittance matrix while Y_{ld} is a diagonal matrix obtained from the derivatives of nonlinear load currents with respect to nodal voltages. After Gaussian elimination we get:

$$\hat{R}_N = R_N - C A_G^{-1} R_G, \tag{8}$$

$$\hat{J}_N = J_N - C A_G^{-1} B, \tag{9}$$

$$-\hat{J}_N \Delta V = \hat{R}_N, \tag{10}$$

and then perform backward block substitution to obtain:

$$\Delta X = A_G^{-1} (R_G - B \Delta V). \tag{11}$$

Many production grade programs for sequential computers iteratively solve these

equations using the very dishonest Newton (VDHN) method which does not update the Jacobian unless the system undergoes significant change or the iteration number exceeds a pre-determined threshold value. Thus, computation time is saved by not only keeping the Jacobian constant over the iterations at the same time step but also over many time steps. The slowdown in the convergence is more than compensated by this time saving. To avoid divergence, the iteration count is tracked and the Jacobian updated if this exceeds a threshold. Since this method is the fastest and the most numerically stable for sequential computers, the improvement achieved by any parallelization must be measured against this sequential method.

2.2 Parallel algorithms

To devise parallel algorithms for this problem the most obvious approach is to apply the Gauss–Jacobi (G–J) method to (3) and (4). This will decouple all the equations, that is, all variables X and V , within one time step (parallel-in-space) (LaScala *et al* 1989):

$$X_t^{k+1} = X_{t-1}^k + (h/2)[f(X_t^{k-1}, V_t^k) + f(X_{t-1}^k, V_{t-1}^k)], \quad (12)$$

$$V_t^{k+1} = Y_D^{-1}(X_t^k)(-Y_{off} V_t^k + I^k(X_t^k, V_t^k)). \quad (13)$$

This is also similar to the Picard iteration. Y_D and Y_{off} are the diagonal and off-diagonal admittance matrix Y . During each iteration period, there is no data exchange until the end of the iteration where variables are updated. In addition, following the procedure for the waveform relaxation method (WRM), all the time-step solutions can be solved in parallel (parallel-in-time) (Ilic-Spong *et al* 1987). In practice, only a limited number of time steps can be solved simultaneously and this is referred to as a time window. Thus, all the equations for all the time steps in a time window can be solved in parallel. As is shown below, the convergence of this approach is very slow and, when compared to the sequential VDHN method, the improvements are not that significant. The main problem is that the G–J is more than 6 times slower than the VDHN on a single processor. When parallelized in time, this slowdown factor increases nearly linearly.

An alternative approach is to examine the structure of the VDHN formulation, (8)–(11). The structure of the Jacobian in (7) shows that the equation for each generator is coupled to the others only through the network. Thus if the matrices B and C are relaxed the generators can be decoupled from the network as well as from each other (LaScala *et al* 1990):

$$\Delta V_t^{k+1} = -\hat{J}_N^{(0)-1} \hat{R}_N(X_t^k, X_{t-1}^k, V_t^k, V_{t-1}^k), \quad (14)$$

$$\Delta X_t^{k+1} = A_{Gt}^{k-1}(R_{Gt}^k + B_t^k \Delta V_t^{k+1}). \quad (15)$$

Note the Jacobian \hat{J}_N computed by (9) is not updated (hence the very dishonest Newton) unless the number of iterations exceeds a threshold value or the system undergoes a topological change. The execution of LDU factorization and forward/backward substitutions is sequential. In fact, the biggest block of equations then is that of the network (14) and the effectiveness of the parallelization depends on the solution of this block (Enns *et al* 1990; Lau *et al* 1990). This parallelization in space can also be augmented with parallelization in time by simultaneously solving the time steps in a window. As is shown below, this parallelization is quite effective in speeding up the solution despite the bottleneck of the large network solution. This method is referred to as the parallel VDHN method.

This raises the obvious question as to whether the VDHN equations can be relaxed further than that of (14)–(15). Two such relaxations are tried here to obtain further parallelization. The successive over-relaxed Newton (SOR-Newton) method uses only the diagonal part of the Jacobian instead of the full Jacobian, while the Maclaurin–Newton method (MNM) takes advantage of the truncated Maclaurin series to approximate the inverse of the Jacobian.

To avoid the time-consuming computation of inverting the network Jacobian matrix, the one-step SOR-Newton algorithm uses an approximation for the whole Jacobian matrix containing only diagonal elements. The iterative equations can be stated as

$$\partial f_q(X_t^k, V_t^k)/\partial S_{p,t} = \begin{cases} \partial_q f_q(X_t^k, V_t^k), & p = q, \\ 0, & p \neq q. \end{cases} \quad (16)$$

The $(k + 1)$ th updated values of any variable in (X_t, V_t) are given by:

$$S_{q,t}^{k+1} = S_{q,t}^k - W_q [f_q(X_t^k, V_t^k)/\partial_q f_q(X_t^k, V_t^k)], \quad (17)$$

where S represents the total elements of X and V , and W_q is the relaxation factor for the q th element of S .

The parallel iterative MNM algorithm can be stated as:

$$\Delta V_t^{k+1} \approx \omega_v (I - \hat{J}_D^{-1} \hat{J}_{off}) \hat{J}_D^{-1} \hat{R}_N^k, \quad (18)$$

$$\Delta X_t^{k+1} \approx \omega_x A_{G_t}^{k-1} (\hat{R}_{G_t}^k + B_t^k \Delta V_t^{k+1}), \quad (19)$$

where ω_v and ω_x are acceleration factors. \hat{J}_D is the diagonal matrix of the Jacobian and J_{off} is the off-diagonal matrix which is the same as the off-diagonal matrix of the network admittance matrix Y . The multiplication of a matrix and a vector can be easily parallelized. A further improvement can be obtained if a secondary iteration technique is utilized.

As is shown below, both these relaxed methods run approximately as fast as the VDHN, but are benefitted by large parallelism. The methods do not require any bus ordering to minimize the fill-ins, which reduce the programming complexity. Their drawbacks are the large number of iterations required for convergence relative to the regular Newton algorithm and their sensitivity to the severity of the faults and the locations. Further investigation into this type of algorithms is needed for future industrial applications.

3. Implementation

The success of any iterative algorithm for solving large problems depends largely on implementation techniques. This is even more so when implementing parallel algorithms on multiprocessor computers. Implementation techniques are usually not amenable to theoretical analysis. The following implementation techniques and their variations have been tried by us and other researchers. Their advantages and limitations are discussed here in the light of our experience in utilizing these with various algorithms and architectures.

3.1 Partitioning

Partitioning a network or a group of generators into subgroups among processors is the common way to achieve parallelism. If the task scheduling is done by dividing

the total generators and buses evenly among processors, the overhead due to uneven computations can be considerable because the network buses have very different connectivity and the differential equations for each generator are very different both in size and complexity. The alternative way is to divide the load according to the total number of equations, but this increases programming difficulty and message exchange which in turn increases the overhead, thus lowering efficiency. The efficient method is to assign different numbers of generator blocks and buses to processors, taking into account the bus connectivity and complexity of the differential equations, so that each processor has roughly the same computational load. The penalty is the increased programming difficulty, especially for the general-purpose programs (Brasch *et al* 1978; Lee *et al* 1989).

A more sophisticated partitioning scheme is the dynamic partitioning which divides the whole network into different sized regions according to their electrical distance from the perturbation, and groups the generators according to their coherency. Different model representations for various regions can be used with changeable step lengths. This localized response significantly improves the execution speed, but with the lower parallelism (due to inter-model switch) and the interpolation of the variables at different level groups (due to multiple time-step sizes) require considerable message exchange and some sequential executions. For limited parallelism this dynamic multi-level partitioning has the advantage.

3.2 Windowing

It is now well established that parallelizing in time (as in WRM) helps speed up the computation. However, the convergence is sequential, i.e. later steps converge later because results from the previous steps are required. This means that processors for the earlier steps idle after convergence and thus introduce inefficiencies. The compromise is to use a few steps in parallel at a time and once the steps in this window have converged, the next window of steps is initialized and processed. Obviously, there is an optimal window size, which was shown, in a typical implementation, to be 8 time steps (step size = 0.02s) for a 662 bus system (Chai *et al* 1991)

3.3 Asynchronization

The asynchronization in parallel processing is generally considered a good means to reduce the massive communication overhead, especially for the relaxed iterative method. The asynchronous message passing and receiving in Gauss–Jacobi iterations can considerably increase the solution speed because little time is wasted in synchronization at the end of each iteration and the immediately available updates can be used. This method becomes a kind of quasi-Gauss–Seidel method. There are two bottlenecks associated with this asynchronization. The first is the difficulty of the convergence control. Processors may exit even before the real convergence is reached since all processors may not receive the message in time and thus think that the increments are zero. Another possibility is that some processors receive the updates in time, which are close to the old values, but some other processors which should have received significantly different updates may not get the message, and thus come to the wrong conclusion that convergence was reached.

The second bottleneck is the unpredictability of the solution speed and iteration number for the same case tested at different times. A tiny difference in computing

speed and execution sequence will change the order of message-passing and receiving, and thus affect the number of iterations required for convergence, which is particularly true in a multi-user environment.

For fast convergent algorithms such as Newton-type methods, the reduction of residuals or increments after one iteration is usually near an order of magnitude. If the iteration is not synchronized, the variables, those that do not receive and those that do receive messages in time, have a large difference. This incompatibility will cause serious convergence problems, and often a divergence will occur. The author and his colleagues have tested many cases with quadratic and quasi-quadratic convergence algorithms using asynchronization, and found that *none* of the cases have convergent results. The asynchronization approach may be unfortunately limited to the slow convergent parallel algorithms such as the Gauss–Jacobi.

3.4 Multi-grid approach

The multi-grid approach is usually used for waveform-relaxation type (or parallel in time) methods. A more general classification may include multi-rate network partitioning or space multi-grid methods. The multi-grid methods involve different models for coarse and fine grids. The coarse-grid solutions provide better estimates for the fine-grid initialization and thus the solution speed is improved. However, sometimes the multi-grid strategy may instead suffer in convergence because of the inter-grid change of models and the interpolation introducing new errors which cause the slowdown.

Again, due to the fast convergence of Newton-type algorithms, the multi-grid method has been ineffective in improving the execution speed. Different models and step sizes used in different grids and the inter-grid interpolation make the parallelization very difficult, especially when using parallelism both in space and in time. The advantage of multi-grid methods is more limited to the Gauss-type algorithms for the relatively simple models and non-serious stiff problems.

3.5 Toroidal implementation

The parallelism in space and in time with the windowing technique has been quite effective in speed-up improvement as compared with pure parallelism in space. Within the window, the processors which have completed the convergence for the initial time steps of a window may just sit idle or do redundant calculations while processors in later time steps are busy with computation. To reduce this processor idling, a travelling window technique or the toroidal method is introduced here. When one time step reaches convergence, that group of processors send a signal to the other processors that they are starting to simulate the next ($t + \text{window}$) time step, which is actually at the beginning of the new window. This new window is one time step forward, and hence the name 'travelling window'. All processors are busy until the last window of the study interval, where the CP units at different time steps exit in the order of their convergence and the last step CPU exits last.

The main bottleneck in this toroidal implementation is its complicated inter-processor coordination. Many signals need to be passed to the processors both prior to and after the current time step, creating costly overhead. Generally, the toroidal method has about 5% to 50% improvement over the windowing technique, depending on

how much parallelism in space and in time is used, the cases simulated and also the types of the algorithms used. Detailed simulation results are given in later sections.

4. Results

In this section, results of implementing the above methods using some of the implementation techniques mentioned are shown. The tests are conducted on a 662 bus power system that represents a portion of the mid-western United States. Realistic models of the generators, exciters and loads are used to ensure that the algorithms are exercised on all the usual nonlinearities and time constants. These results were obtained on two different types of multiprocessing computers. One type was a 32 processor iPSC hypercube machine that transfers data between processors by passing messages. The other type transfers data through a shared memory and was represented by an Alliant 8 processor computer and a Sequent 26 processor computer.

To measure the effectiveness of a parallel algorithm, let G be defined as parallel speedup of a particular algorithm:

$$G = \frac{\text{run time of sequential algorithm on 1 CPU}}{\text{run time of parallel algorithm on } N \text{ CPU}}$$

This measure only shows the efficiency of the parallelization in terms of the algorithm itself but not a comparison with the best sequential algorithm. Thus a more realistic measure is defined as the speedup of the parallel algorithm against the sequential VDHN method:

$$SP = \frac{\text{run time of serial VDHN on 1 CPU}}{\text{run time of parallel algorithm on } N \text{ CPU}}$$

Table 1 clearly shows the difference between these two measures when examining the numerical results for the Gauss–Jacobi as implemented on a 32 node iPSC hypercube. The speedup G increases monotonically and even reaches as high as 25 times with 2 processors computing parallel in space and 16 parallel in time. But this measure G , although used by many researchers, can be misleading because the real speedup SP against the sequential VDHN can be seen to be quite modest.

The reason for this discrepancy between G and SP can be seen in table 2. The G–J method is 6 times slower than the VDHN on one processor without parallelization in time and even more so as more time steps are solved in parallel. This is because the updating of trajectories propagates through time and larger time windows provide better parallelization efficiency at the cost of higher overall run time.

Table 1. Speedups of parallel G–J algorithm using 32 CP units.

Gain	Window				
	1	2	4	8	16
G	5.43	7.52	15.57	21.78	24.92
SP	0.88	1.03	1.79	2.27	2.19

Table 2. Serial G-J on 1 node (iPSC), study interval = 0.48s. Step size = 0.02s. US Midwestern 662 bus system.

Window (Step #)	1	2	4	8	16
Run time (s)	1237.6	1457.2	1736.5	1916.3	2275.1
Slowdown	6.2	7.3	8.7	9.6	11.4

For the G-J method, the more meaningful measure SP increases as window sizes increase, but peaks at window equal to 8 time steps with speedup of 2.3. The disadvantage of the G-J algorithm is clearly displayed even though it is 100% parallelizable. When large parallelism in space is used, the calculation time of the subtasks assigned to each CPU is much shortened and easily overwhelmed by the massive communication overhead. Any significant improvement in speed relies mainly on fast communication channels.

The results obtained when using the parallel VDHN method is much better than the G-J. As was said before, the bottleneck in this method is the size of the network calculations. For the 662-bus system, this sequential portion takes about 7% of the entire iteration time. Using Amdahl's (1967) rule the maximum theoretical speedup is given by:

$$G = 1/[f_s + (f_p/N) + f_{oh}] < 1/f_s,$$

which, in this case, is 14.3 (1/7). The f_s represents the percentage of the execution time for the sequential portion, while f_p is for parallelizable parts, and f_{oh} is overhead. The actual speedup gain G ($G = SP$ for this algorithm) is always less and for this case, shown in table 3, it is only about 5.6 when implemented on a 32-node hypercube if only parallelism in space is used (Chai *et al* 1991). Any large increase in CPU numbers will have little improvement in this saturated speedup. With the parallelism in space and in time technique, the speedup G is enhanced to 8.79 times using 4 in space and 8 in time. It is further improved by using the toroidal (travelling window) technique, as shown in table 3. The largest improvement in speedup is only around 3 (from 5.9 for windowing to 8.9 for toroidal) with 2-in-space-and-16-in-time implementation. The difficult part still lies in the sequential solution of the linear network equation (10).

The sequential execution speed of the SOR-Newton method is slightly slower than the VDHN method, and also has the disadvantage of being more sensitive to the severity and location of the faults. The method is parallel and thus speedup gain obtained on an Alliant shared memory machine is an impressive 7.41 times on 8 CPU units, as shown in table 4. The implementation results in an iPSC 32 node machine are much

Table 3. Gain of parallel VDHN using windowing and toroidal methods on a 32-node hypercube.

Method	Window				
	1	2	4	8	16
Windowing	5.6	6.8	8.2	8.8	5.9
Toroidal	5.6	7.1	9.2	10.2	8.9

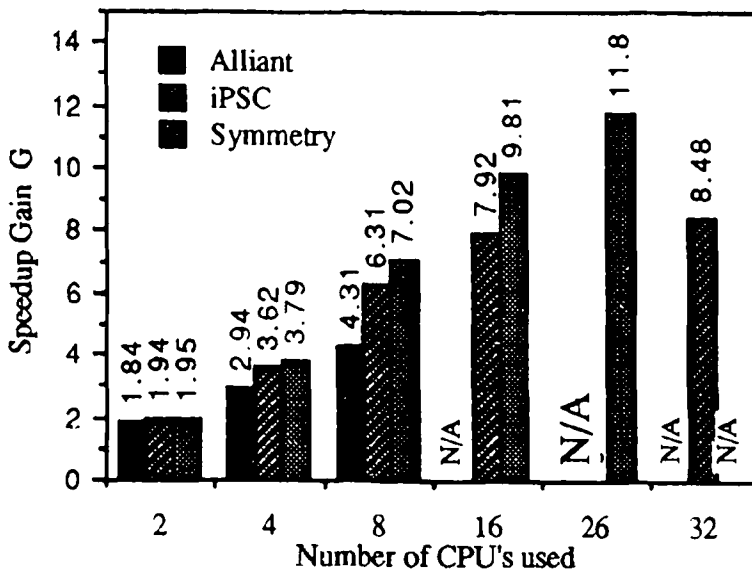
Table 4. Speedups of SOR-Newton against VDHN.

CPU #		1	2	4	8	16	32
ipsc	G	1	1.67	2.6	3.48	4.01	3.62
	SP	0.93	1.55	2.42	3.24	3.73	3.37
Alliant	G	1	1.99	3.91	7.41	N/A	N/A
	SP	0.87	1.73	3.40	6.45	N/A	N/A

lower than expected. The reason is that the SOR-Newton requires many iterations to converge, but the execution time of each iteration is very short. The massive data exchanges at the end of each iteration consume more time than the calculation itself and cause a significant degradation in speedups. Without doubt, if the toroidal method were used in implementation, the parallel efficiency would improve.

Although the SOR-Newton method is an improvement over the parallel VDHN, its major drawback, however, is not evident in the tabulated results. The severe relaxation of the Jacobian elements significantly weakens the convergence of the method. Although convergence was obtained in these tests, often some tuning of the relaxation factors was needed to do so. Since such case dependent tuning cannot be done during production use, this is a severe disadvantage of this method.

The Maclaurin-Newton method is very similar to the SOR-Newton in its characteristics. The speedup vs. the number of processors for MNM on three different multiprocessors is shown in figure 1. The results were obtained with parallelizing only the equations (i.e. in space). The gains obtained from the ipsc and symmetry machine are quite close when a small number of CP units are involved, while the gains from the Alliant are much lower. This is because the Alliant parallel compiler cannot "see" some part of the parallelizable code due to the sparsity coding technique, and hence "artificially" creates a sequential portion which drives down the efficiency. This invisibility from an unsophisticated software is a serious bottleneck in exploiting

**Figure 1.** Speedups of MNM vs. CPU number on three machines.

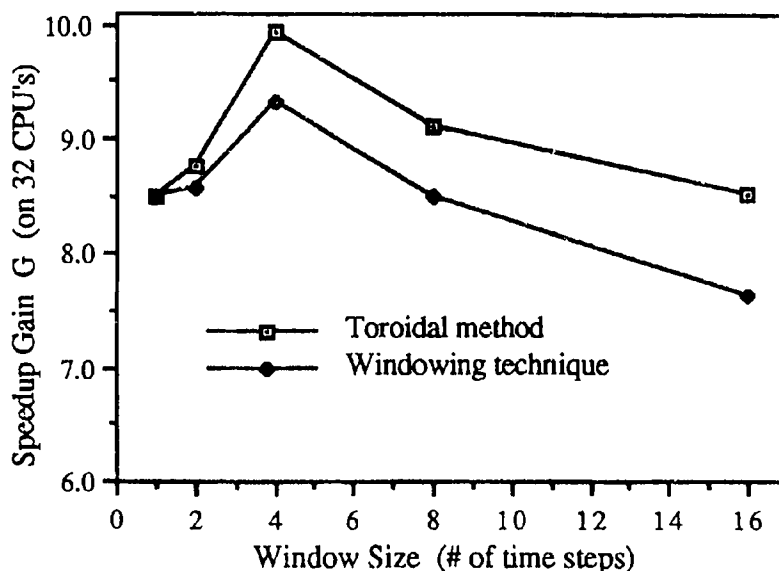


Figure 2. Comparisons of toroidal method and windowing technique for MNM with different window sizes on 32 CP units.

large parallelism, especially for the production grade programs. The gain of 11.8 out of 26 units on the symmetry is mainly attributed to enormous programming efforts by decomposing the tasks manually. On using auto-parallel directives provided by the manufacturer, the gain is much lower. The speedups on the iPSC can reach 8.48 for parallelism in space, which is much higher than 5.6 for the VDHN as shown in table 3, because the MNM is not only computationally intensive but also completely parallel. However, the gains do quickly get saturated when more than 8 units are used. The upperbound seems to be around an order of magnitude if parallel algorithms are implemented on a message-passing machine.

The parallelism in time plus in space provides better efficiency in spite of a large number of iterations required. As an example, the choice of window sizes vs. speedup G for the MNM using 32 nodes of the iPSC, is plotted in figure 2. But the increase of G for the MNM method is not very significant, as compared with the speedups for the VDHN (Chai *et al* 1991). The main reason is that the acceleration factors used in MNM cause a large swing in residuals when the current step process constantly receives new updates from the previous step. The slow adjustment of the residuals due to parallelism in time decreases the speedup gains. The curve of speedup G vs. window size is thus quite flat. With the toroidal technique, some improvement is achieved at large window sizes, but because of this sensitivity of the residuals to the acceleration factors and updates from the previous step, the gains are not enhanced on a large scale.

From the above results, it can be seen that the relaxed Newton methods are better than the Gauss-Jacobi, which is naturally parallel. However, the extent of the relaxation on the Newton algorithm does affect its convergence characteristics. Thus, the completely parallel SOR-Newton and the MNM, although faster than the parallel VDHN, are susceptible to divergence under many more conditions. The parallel VDHN, however, appears to be quite robust while providing speedups of more than a magnitude over the sequential VDHN.

5. Conclusions

The research in parallel computation for power system dynamics is relatively recent although the pressing need for on-line analysis has engendered a high level of interest. Most of the early work available is in the development of parallel algorithms themselves without actual implementation tests. Since the architecture of the multiprocessor computer has a major effect on the success of the algorithm, the matching of the possible algorithms to available architectures becomes a daunting task. This is further aggravated by the fact that parallel-computer architectures are still evolving rapidly and it is difficult to predict what commercially available multiprocessors will look like in the future. There is also the possibility of developing specialized architectures to fit the algorithms but it is generally believed that this may not be a commercially viable approach.

In this paper, an overview is presented from the perspective of using multiprocessors that are at present commercially available. Message-passing hypercube and memory-sharing architectures are used to test the viability of parallel algorithms. Comparison is made with the fastest known sequential algorithm, the very dishonest Newton method (VDHN), which is used in most of the production grade programs. Three types of parallel algorithms are compared. The completely parallel Gauss–Jacobi method has too slow a convergence to take much advantage of the simultaneous computation. The simple decoupling of the generator equations in the VDHN method provides computational speedup of about an order of magnitude. Higher speedups can be obtained by further relaxing the Jacobian matrix of the VDHN method using the SOR-Newton or the Maclaurin–Newton, but only at the cost of making the convergence less robust.

It is also shown that the parallelization of the equations can be augmented by parallelizing the computation at different time steps. Since the convergence is unidirectional in time, this parallelization in time does not pay if too many time steps are solved simultaneously. It appears that a time window of four to eight time steps gives the best results. Some indication of the suitability of architectures is also obtained. Hypercube type architectures can produce better results for some algorithms because the actual assignment of computation between the processors can be controlled more precisely, but only at the cost of more custom programming. Memory-sharing computers, on the other hand, are more flexible in handling different programs without much modification. This makes it the more commercially viable alternative, although much improvement on existing compilers is certainly needed.

All the results mentioned in this paper were obtained by Dr Ning Zhu and Dr Jason Chai, who were my research associates.

References

- Amdahl G 1967 Validity of the single-processor approach to achieving large-scale computer capabilities. *AFIPS Conference Proceedings* 30: 483–485
- Brasch F M, Van Ness J E, Kang S C 1978 Evaluation of multiprocessor algorithms for transient stability problems. EPRI-EL-947

- Chai J S, Zhu N, Bose A, Tylavsky D J 1991 Parallel Newton type methods for power system stability analysis using local and shared memory multiprocessors. *IEEE Trans. Power Syst.* 6: 1539-1545
- Enns M K, Tinney W F, Alvarado F L 1990 Sparse matrix inverse factors. *IEEE Trans. Power Syst.* 5: 466-473
- Illic-Spong M, Crow M L, Pai M A 1987 Transient stability simulation by waveform relaxation methods. *IEEE Trans. Power Syst.* 2: 943-952
- LaScala M, Bose A, Tylavsky D J, Chai J S 1989 A highly parallel method for transient stability analysis. *IEEE Power Industry Computer Applications Conference*
- LaScala M, Brucoli M, Torelli F, Trovato M 1990 A Gauss-Jacobi-block-Newton method for parallel transient stability analysis, *IEEE/PES Winter Meeting, Atlanta*
- Lau K, Tylavsky D J, Bose A 1990 Coarse grain scheduling in parallel triangular factorization and solution of power system matrices. *IEEE/PES Summer Meeting, Minneapolis*
- Lee S Y, Chiang H D, Lee K G 1989 Parallel power system transient stability analysis on hypercube multiprocessors. *IEEE Power Industry Computer Application Conference* (New York: IEEE)

The constants C_1, C_2, C_3, C_4 are determined by the initial conditions. The exponents λ_1 and λ_2 are the eigenvalues of the Jacobian matrix

$$J = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

and can be obtained by solving $|J - \lambda I| = 0$ (where a, b, c, d are the partial derivatives of f_1 and f_2 evaluated w.r.t. x and y at the equilibrium point).

$$\lambda_{1,2} = \frac{1}{2}[\text{Tr}(J) \pm D^{\frac{1}{2}}]$$

$$\text{Tr}(J) = a + d; \Delta = \text{discriminant} = \text{Tr}(J)^2 - 4 \det(J).$$

Except for three critical cases: (i) $\det(j) = 0$; (ii) $\text{Tr}(J) = 0$; $\det(J) > 0$; (iii) $\det(J) = 0$; $\text{Tr}(J) = 0$; the integral curves of the nonlinear system have the same behaviour as those of linearized systems in the neighbourhood of the equilibrium. These results are summarized with the values of the trace and determinant of the corresponding Jacobian matrix as shown in the phase diagram (figure 1). For linear systems in R^3 Reyn (1964) made sound classification and arrangement of phase portraits.

However, in the three critical cases mentioned before, the structure of orbits in the state space will change qualitatively. Such a qualitative change is called a bifurcation. This bifurcation may be due to variation of certain parameters in the system. The critical value of the parameter where the bifurcation occurs is the bifurcation value of the parameter.

The paper is organised as follows: Section 2 describes the general principles involved in the study of bifurcation behaviour of an n dimensional dynamical system. Sections 3, 4 and 5 discuss the recent advances in the numerical techniques that can be effectively used to identify various bifurcation points.

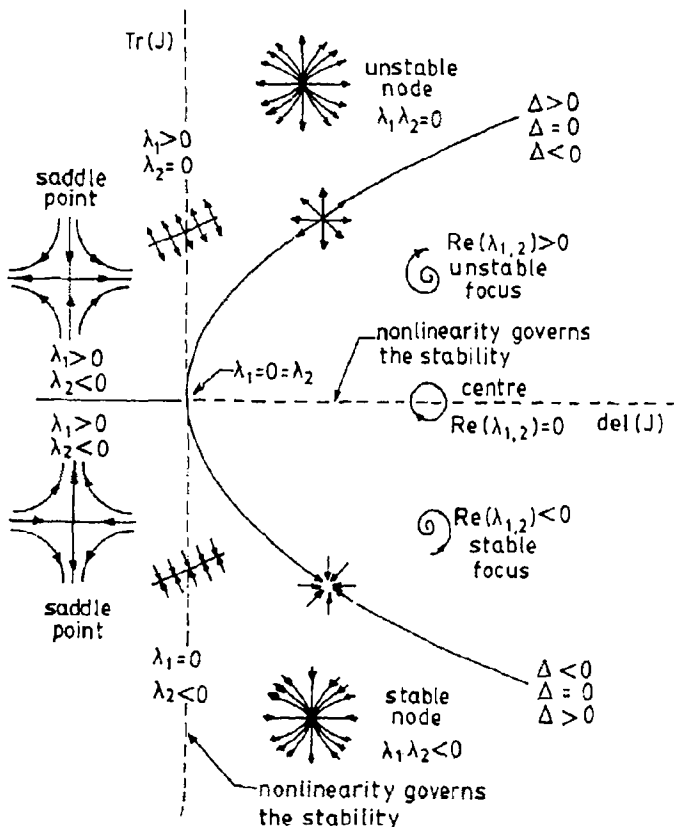


Figure 1. Phase diagram.

2. Bifurcation of dynamical systems

Consider a dynamical model of a system described by autonomous differential equations of the vector form in n -dimensional space

$$x' = F(x, \lambda), \quad x \in R^n; \lambda \in R^k. \quad (5)$$

Here x denotes the state variables. For power system models these are: generator angles, generator angular velocities, load voltage magnitudes, or angles etc. λ is a vector of time invariant scalar parameters. At an equilibrium point (x_0, λ_0) , the left hand term x' of equation becomes zero, i.e., the steady state solution of (5) satisfies the set of nonlinear algebraic equations $F(x_0, \lambda_0) = 0$. If the eigenvalues of the Jacobian $\partial F/\partial x$ become non-zero, then according to implicit function theorem the equilibria of (5) can be expressed as the smooth function of $x = x(\lambda)$. The function $x(\lambda)$ is called the branch of equilibria. However if the Jacobian has an eigenvalue with zero real part occurring at some λ , say λ_c , the system $x' = F(x_c, \lambda_c)$ is structurally unstable and several branches of $x = x(\lambda)$ can come together at (x_c, λ_c) in R^{n+k} . The parameter set λ_c where the system loses its stability is called a bifurcation set. The point (x_c, λ_c) is called bifurcation point. (In general, in engineering systems a one-parameter family with $k-1$ relations between the parameters $\mu_1, \mu_2, \mu_3, \dots$ can be represented as a curve, λ , in the k -dimensional parameter space.) Thus the principle of linear stability differentiates between two categories of equilibrium solutions. For the hyperbolic fixed points (where the eigenvalues have non-zero real parts), linear stability analysis suffices completely. For nonhyperbolic fixed points (the points where at least one eigenvalue has zero real part), a linear stability analysis is not applicable and a full nonlinear analysis has to be carried out. There are techniques available to simplify, without any significant loss of information, the representation of the flow in the nonlinear dynamical systems in the neighbourhood of nonhyperbolic points. One of these techniques is the centre manifold theory. This theory closes the gap left by Hartman–Grobman theorem (HGT). According to HGT, if the Jacobian $\partial F/\partial x$ has no eigenvalues with zero real part, then the family of trajectories near an equilibrium point (x_0, λ_0) of a nonlinear system $x' = F(x, \lambda)$, and those of the locally linearized system have the same topological structure, which means that in the neighbourhood of (x_0, λ_0) there exist homeomorphic mappings which map trajectories of the nonlinear system into trajectories of the linear system. Should, however, an eigenvalue with a zero real part exist, the open question arises how this effects the flow in the neighbourhood of the equilibrium point. It is this gap left open by HGT that is closed by the centre manifold theory.

2.1 Centre manifold

(Carr 1981) Let (x_0, λ_0) be the equilibrium point of $F(x, \lambda)$, and E^s , E^u and E^c the corresponding generalized eigenspaces of the Jacobian matrix $\partial F/\partial x|_{x_0}$, where the real part of the eigenvalues (μ) defines the eigenspaces,

$$Re(\mu) \begin{cases} < 0 - E^s \\ = 0 - E^c \\ > 0 - E^u \end{cases}$$

Then there exist stable W^s , unstable W^u and centre manifold W^c , which are tangential to E^s, E^u, E^c respectively at (x_0, λ_0) . If one is interested in the long term behaviour

(i.e., $t \rightarrow \infty$) the overall dynamics in the neighbourhood of an equilibrium point are reproduced by the flow on the centre manifold W^c . This reduction of the dynamics to those in the W^c subspace is the subject of centre manifold theory. In order to calculate the flow of the reduced dynamics on W^c , the nonlinear vector field can be transformed to the following form. We can assume that unstable manifold W^u is empty. This makes the presentation simple, without loss of generality.

$$x'_c = A_c x_c + f(x_c, x_s); \quad x_c \in R^{n_c}, \quad (6)$$

$$x'_s = A_s x_s + g(x_c, x_s); \quad x_s \in R^{n_s}. \quad (7)$$

The matrix $A_c(n_c, n_c)$ contains n_c eigenvalues with zero real parts. A_s matrix (n_s, n_s) contains n_s eigenvalues with negative real parts. The nonlinear functions f and g should be continuously differentiable at least twice and vanish together with their first derivatives at the equilibrium point. x_c correspond to centre manifold and are sometimes called active variables. x_s correspond to stable manifold and are called passive variables. Due to nonlinear couplings the influence of x_s in the equation for x_c cannot be ignored. Hence the correct way of analysis is to compute the centre manifold.

$$x_s = h(x_c), \quad (8)$$

by expressing the dependence of x_s on x_c from (7) and then to eliminate from (6) to obtain the bifurcation equation

$$x'_c = A_c x_c + f(x_c, h(x_c)). \quad (9)$$

Then the equivalence theorem (Guckenheimer & Holmes 1983) states that for $t \rightarrow \infty$, the dynamics of (9) in the neighbourhood of the equilibrium point is equivalent to the dynamics of the initial system $x' = F(x, \lambda)$ with λ fixed at the value λ_c . In order to solve (9), one has to know the function $h(x_c)$. This can be obtained as follows

$$dx_s/dt = dh(x_c)/dt = \partial h/\partial x_c \times dx_c/dt; \quad (10)$$

from (6) and (7), (10) can be written as

$$A_s h(x_c) + g(x_c, h(x_c)) = (\partial h/\partial x_c)[A_c x_c + f(x_c, h(x_c))]$$

or

$$(\partial h/\partial x_c)[A_c x_c + f(x_c, h(x_c))] - A_s h(x_c) - g(x_c, h(x_c)) = 0. \quad (11)$$

The functions h and $(\partial h/\partial x_c)$ are zero at the equilibrium point. Equation (11) is in general a partial differential equation which cannot be solved exactly in most cases. But its solution can sometimes be approximated by a series expansion near the equilibrium point. The aforementioned reduction technique of the centre manifold theory, which is clearly shaped by a geometrical interpretation of dynamics in the phase space, has its physical counterpart in the slaving principle associated with the synergetic approach founded by the physicist Herman Haken in the early seventies (Haken 1983). n also divides the dynamics into stability groups on the basis of a linear stability analysis in accordance with the classification of the eigenvalues.

In summary, if x is nonhyperbolic then there exist invariant centre manifolds tangential to the centre subspace and its dimension is equal to the number of eigenvalues of the Jacobian matrix having zero real parts. Then the practically interesting local stability behaviour is completely governed by the flow on the centre manifold.

Effect of small perturbations of the critical parameters around the bifurcation point can also be studied by unfolding the centre manifold. This can be achieved via the method of normal forms (Arnold 1972, 1983; Bruno 1989; Wiggins 1990). Normal forms play an essential role in bifurcation theory because they provide the simplest system of equations that describe the dynamics of the original system close to the bifurcation points. Even away from the bifurcation point Poincaré's theory of normal forms reduces the initial nonlinear equations into the simplest possible forms without distorting the dynamic behaviour in the neighbourhood of fixed points or periodic solutions. The transformations, which yield to a reduction to normal forms, can be generated by developing the deviations from a state of equilibrium or from periodic motion into power series. Symbolic manipulation packages like MACSYMA, and MAPLE, are helpful in the development of normal forms. Application of normal form theory to power system examples is given by Vittal *et al* (1992, pp. 2553–56) and examples of the application of centre-manifold theory to power systems are given by Rajgopalan *et al* (1989), Chiang *et al* (1989) and Ajarapu & Lee (1992).

The number of possible types of bifurcation increases rapidly with increasing dimension of the parameter space. The bifurcations are organised hierarchically with increasing co-dimension, where co-dimension is the lowest dimension of a parameter space which is necessary to observe a given bifurcation phenomenon. In this paper we discuss only the dynamical system with a single parameter variation. Changing this parameter may drive the system into a critical state at which (i) a real eigenvalue becomes zero or (ii) a pair of complex conjugate eigenvalues becomes imaginary. In case (i) new branches of stationary solutions usually arise and are called static bifurcations. (Typical static bifurcations are (i) saddle node or fold, (ii) trans-critical, and (iii) pitchfork.) Case (ii) may lead to the birth of a branch of periodic solutions called dynamic bifurcations. Typical dynamical bifurcation is Hopf.

In many practical engineering problems, identification of these bifurcations is important. For example, buckling load of elastic structures (Riks 1979) and steady state voltage collapse in power systems (Chiang *et al* 1989; Ajarapu & Lee 1991) is related to saddle-node bifurcations. Hopf bifurcation and bifurcation of periodic solutions are observed in chemical engineering (Halvacek 1986), mechanical engineering (Moon 1987; Thomson & Stewart 1986) and electrical engineering (Holmes 1980) to name a few. The next section concentrates on the numerical identification of these bifurcations.

3. Detection of bifurcation points

3.1 Static bifurcations

3.1a *Fold bifurcation*: In mathematical literature these are sometimes called turning points. To calculate these turning points several methods have been suggested. These methods based their analysis basically on two approaches.

- Direct methods,
- Indirect methods.

In direct methods, the original system of equations is suitably augmented by an extra set of equations in such a way that the turning point becomes the solution of this system. Indirect methods start around the neighbourhood of a turning point and

several different solutions of $F(x, \lambda) = 0$ are calculated by continuation. At the same time a certain test function is monitored along the solution path which gives information about the turning point. Both these approaches are described in the following sections.


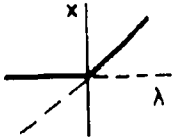
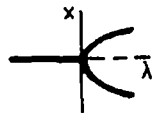
(i) *Direct methods* – In many physical problems it is necessary to solve a system of nonlinear equations of the form, $F(x, \lambda) = 0$. If λ is varied, the corresponding state vector x changes. As mentioned before, as long as F_x is nonsingular equilibrium point x can be obtained for a given value of the parameter λ . However, sometimes we are interested in the maximum allowable variation of λ . Unfortunately, in most of the physical problems the Jacobian F_x becomes singular, when the parameter approaches its maximum value. When F_x becomes singular, $F(x, \lambda) = 0$ cannot be solved by ordinary Newton–Raphson method. To avoid this singularity several methods have been published e.g. Abbot (1978), Seydel (1979), Moore & Spence (1980). They cleverly augmented the original system of equations in such a way that for this enlarged system, the turning point becomes regular. For example, if

$$Y_c^T = [x_c, \lambda_c, h] \text{ solves}$$

$$G(Y) = \begin{bmatrix} F(x, \lambda) \\ h_k - 1 \\ F_x(x, \lambda)h \end{bmatrix} = 0.$$

Then (x_c, λ_c) is a turning point of $F(x, \lambda) = 0$. This procedure basically augments the original equations of $F(x, \lambda) = 0$ by $F_x(x, \lambda)h = 0$, with $h_k = 1$. This augmentation makes the Jacobian G_x of enlarged system $G(Y)$ non-singular and guarantees a solution. The proof can be found in Seydel (1979). This approach has some drawbacks. The dimension of the nonlinear set of equations to be solved is twice that of the original number. The approach requires a good estimate for the vector h . However, convergence of the direct method is very fast if the initial operating point is close to the turning point. The enlarged system can be solved in such a way that it requires the

Table 1. Static bifurcation types.

Bifurcation type	Transversality condition	Prototype equation	Bifurcation diagram
Fold	$\frac{\partial F}{\partial \lambda} \neq 0; \frac{\partial^2 F}{\partial x^2} \neq 0$	$\lambda - x^2 = 0$	
Transcritical	$\frac{\partial^2 F}{\partial \lambda \partial x} \neq 0; \frac{\partial^2 F}{\partial x^2} \neq 0$	$\lambda x - x^2 = 0$	
Pitch fork	$\frac{\partial^2 F}{\partial \lambda \partial x} \neq 0; \frac{\partial^2 F}{\partial x^3} \neq 0$	$\lambda x - x^3 = 0$	

----- unstable mode; ——— stable mode

solution of $n \times n$ (n is the dimension of the Jacobian $F_x(x, \lambda)$) linear systems, each with the same matrix. This method needs only one LU decomposition. At this turning point, $\text{rank } F_x(x_c, \lambda_c) = n - 1$ and $F_\lambda(x_c, \lambda_c) \in \text{range } F_x(x_c, \lambda_c)$, that is $\text{rank } F_x(x_c, \lambda_c) / F_\lambda(x_c, \lambda_c) = n$. These are called transversality conditions. Depending on the type of transversality condition, different types of static bifurcations can occur. Fold or saddle node is generic or the most commonly occurring static bifurcation. Table 1 summarizes the type of static bifurcation and corresponding transversality condition for a one-dimensional scalar system. Details can be found in Wiggins' (1990) book. The application of this method to power system voltage stability is reported by Ajarapu (1991) and Alvarado & Jung (1989).

(ii) *Indirect methods* – In this section we consider as before the calculation of a solution branch of the problem

$$F(x, \lambda) = 0. \tag{11}$$

System (10) consists of n scalar equations defining a curve in $(n + 1)$ dimensional (x, λ) space. Continuation means tracing this curve. If $F_x^0 = F_x(x_0, \lambda_0)$ is non-singular then we can solve (10) for $\lambda = \lambda_0$ by Newton's method to obtain x_0 , and the iterations will converge quadratically with a good-enough initial guess. Further, the non-singularity of F_x at λ_0 guarantees, by the implicit function theorem, that for λ in a neighbourhood of λ_0 there is a unique local branch of solutions $x(\lambda)$ with $x(\lambda_0) = x_0$. Thus given the solution x_0 at λ_0 we can compute the solution x_1 at λ_1 by taking total derivative of (10) as follows:

$$(\partial F / \partial x) \times (dx/d\lambda) + (\partial F / \partial \lambda) = 0, \tag{12}$$

$$(dx/d\lambda) = -(\partial F / \partial x)^{-1} (\partial F / \partial \lambda), \tag{13}$$

by integrating this system, starting from the initial value (x_0, λ_0) , one obtains the branch parametrized by λ . This procedure, as it was proposed in Davidenko (1953), fails at turning points because one encounters a singularity of F_x . However an important special case resulting from (12) is the tangent to the branch. If we abbreviate this tangent with the symbol t . Then

$$t_i = dx_i (1 \leq i \leq n), \quad t_{i+1} = d\lambda.$$

Equation (12) represents n equations, whereas the tangent vector t contains $(n + 1)$ unknowns. A normalization has to be imposed to give t a non-zero length. One can write for example

$$e_k^T t = t_k = 1,$$

where e_k is the k th unit vector of the $(n + 1)$ dimensional space. With this the tangent t can be obtained as the solution of the linear system

$$\begin{bmatrix} F_x & F_\lambda \\ e_k & 1 \end{bmatrix} \times [t] = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{14}$$

It can be shown (Rheinboldt & Burkhardt 1983; Rheinboldt 1986) that the Jacobian of the resulting $(n + 1)$ by $(n + 1)$ system of (14) is non-singular. This tangent vector

t can be used to predict the next solution $(x_p^{k+1}, \lambda_p^{k+1})$ for a given initial solution (x^{k+1}, λ^{k+1}) as

$$(x_p^{k+1}, \lambda_p^{k+1}) = (x^k, \lambda^k) + \sigma t^k,$$

where σ is an appropriate step length. This tangent predictor can be considered as an Euler step for solving a differential equation that describes the branch. This predicted solution is utilized to correct the solution to satisfy $F(x^{k+1}, \lambda^{k+1}) = 0$. This process is explained in the next section with a power system example.

3.2 Application to power system example

If F is used to denote the whole set of steady state power flow equations, then to solve the problem, the continuation algorithm starts from a known solution and uses a predictor-corrector scheme to find subsequent solutions at different load levels. Here

$$x = [\delta, v, \lambda]^T,$$

δ = load voltage angles,

v = load voltage magnitudes,

λ = load parameter.

3.2a Predictor: Once a base solution has been found ($\lambda = 0$) a predictor of the next solution can be made by taking appropriately sized steps in a direction tangential to the solution path. If $t = [d\delta dv d\lambda]^T$ is used to denote the tangent vector, then

$$\begin{bmatrix} F_\delta F_v F_\lambda \\ e_k \end{bmatrix} [t] = \begin{bmatrix} 0 \\ \pm 1 \end{bmatrix}. \quad (15)$$

Once the tangent vector has been found by solving (15), the prediction can be made as follows

$$\begin{bmatrix} \delta^* \\ v^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} \delta \\ v \\ \lambda \end{bmatrix} + \sigma \begin{bmatrix} d\delta \\ dv \\ d\lambda \end{bmatrix},$$

where "*" denotes the predicted solution for a subsequent value of λ (loading) and σ is a scalar that designates the step size.

3.2b Parametrization and the corrector: Now that a prediction has been made, a method of correcting the approximate solution is needed. Actually, the best way to present this corrector is to expand on the parametrization, which is vital to the process. Each continuation technique has a particular method of identifying each solution along the path being traced. But the local parametrization proposed by Rheinboldt & Burkhardt (1983) looks promising. In local parametrization the local original set of equations is augmented by one equation that specifies the value of the parameter or the value of one of the state variables. In the case of the power system example this means specifying either a bus voltage magnitude, a bus voltage angle, or the load parameter λ . Since the λ is also included as a state variable, the new set

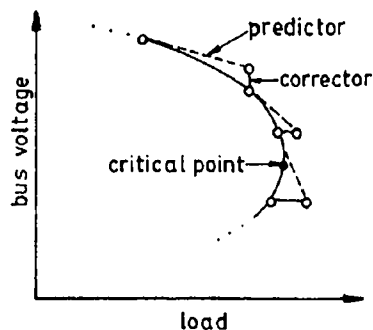


Figure 2. An illustration of the predictor–corrector scheme used in the continuation power flow.

of equations would be

$$\begin{bmatrix} F(x) \\ x_k - \eta \end{bmatrix} = [0]. \tag{16}$$

Selection of the continuation parameter corresponds to the state variable that has the largest tangent vector component. More simply put, this corresponds to the state variable that has the greatest rate of change near a given solution. Therefore x_k at a particular step is the maximum of $(|t_1|, |t_2|, |t_3|, |t_4| \dots |t_m|)$. Now, once a suitable index k and value of η are chosen, a slightly modified Newton–Raphson iterative process can be used to solve the above set of equations (16). The general form of the iterative corrector process at the j th step is

$$-\begin{bmatrix} F_x(x^j) \\ e_k \end{bmatrix} \times [\Delta x^j] = \begin{bmatrix} F(x^j) \\ 0 \end{bmatrix}.$$

The corrector Jacobian can be seen to have the same form as the predicted Jacobian. Actually the index k used in the corrector is the same as that used in the predictor and η will be equal to x_k^* , the predicted value of x_k . In the predictor it is made to have a non-zero differential change ($dx_k = t_k = \pm 1$) and in the corrector its value is specified so that the values of other state variables can be found. Figure 2 demonstrates the predictor–corrector process used in the continuation power flow. This methodology to investigate steady state voltage collapse is reported in Ajarapu & Christy (1992).

4. Hopf bifurcation

4.1 Existence of Hopf bifurcation point

(Marsden & McCracken 1976; Hassard *et al* 1981.) If (i) $F(x_c, \lambda_c) = 0$, (ii) the Jacobian matrix $(\partial F/\partial x)$ has a simple pair of purely imaginary eigenvalues, $\mu(\lambda_c) = \pm j\omega$, (iii) $d(\text{Re}(\mu(\lambda_c)))/d\lambda \neq 0$.

Then there is a birth or death of limit cycles at (x_c, λ_c) depending on the sign of the derivative in (iii). λ_c is the value of the parameter at which Hopf bifurcation occurs. Requirement (iii) guarantees there is a transversal crossing of the imaginary axis by the pair of complex conjugate eigenvalues. Numerical determination of the Hopf bifurcation point involves determination of the point (x_c, λ_c) . A costly way of

identifying the point is to evaluate all the eigenvalues of the Jacobian matrix. However, as in the static approach there are efficient ways of identifying the Hopf point by direct methods as well as by indirect methods.

4.1a Direct methods: Direct methods (Kubicek & Marek 1983) calculate the Hopf point by solving one single suitably chosen equation. At the Hopf point, one pair of complex eigenvalues crosses the imaginary axis. Let this pair be

$$\mu(\lambda) = \alpha(\lambda) \pm j\beta(\lambda),$$

with

$$\alpha(\lambda_c) = 0; \quad \beta(\lambda_c) \neq 0; \quad (d\alpha/d\lambda)\lambda_c \neq 0.$$

For an eigenvalue μ of the Jacobian matrix $F_x [= (\partial F/\partial x)]$, the following equation is valid

$$F_x W = \mu W, \tag{13}$$

where $W = u + jv$ is an eigenvector corresponding to the eigenvalue μ . Since $\alpha(X_c) = 0$, (13) can be written as

$$F_x(u + jv) = (+j\beta)(u + jv),$$

$$F_x u + jF_x v = -\beta v + j\beta u,$$

$$F_x u + \beta v = 0, \tag{14}$$

$$F_x v - \beta u = 0, \tag{15}$$

where u and v are vectors of dimension n . We have in fact $3n$ nonlinear algebraic equations (14), (15) and $F(x, \lambda) = 0$ with $3n + 2$ unknowns ($x_1, x_2, \dots, x_n, u_1, u_2, \dots, u_n, v_1, v_2, \dots, v_n, \lambda, \beta$). However the other two unknowns can be obtained by putting two normalizing conditions that force W to be non-zero. This means that practically we can choose two components of the vectors u and v arbitrarily. The Newton iterations method can be effectively used to solve this $3n$ by the $3n$ system to get the Hopf point. An efficient algorithm based on the direct approach is provided by Giewank & Reddien (1983). The application of the boundary value problem for direct computation of the Hopf points was proposed by Seydel (1981).

4.1b Indirect methods: The Hopf bifurcation point (x_c, λ_c) can also be located by an indirect approach. This can be achieved by obtaining the information collected during any continuation method described before, i.e. an iteration technique is used to solve the algebraic equation $Re(\mu(\lambda)) = 0$ by means of the secant method. A change of sign of the real part $\alpha(\lambda)$ indicates that λ_c has been passed. Therefore the check,

$$\alpha(\lambda_j)\alpha(\lambda_{j-1}) < 0,$$

should be performed after each continuation step $\lambda_{j-1} \rightarrow \lambda_j$.

A good comparison of various methods of computing Hopf bifurcating points is given by Roose (1985). Application of Hopf bifurcation to power system problems can be found in Rajgopalan *et al* (1989, 1991), Abed & Varaiya (1984) and Ajjarapu & Lee (1992).

5. Complex bifurcation

Further variation of the parameter beyond the Hopf point may lead to other complex phenomena; basically one has to trace the monodromy matrix of a periodic orbit for different values of the parameter. The stability of periodic solution is determined by Floquet multipliers which are the eigenvalues of the monodromy matrix. For a particular value of λ , the monodromy matrix has n -Floquet multipliers. The magnitude of one of them is always equal to unity. The other $n - 1$ Floquet multipliers determine (local) stability by the following rule (Arnold 1983; Seydel 1988).

$x(t)$ is stable if $|\mu_j| < 1$, for $j = 1, \dots, n - 1$,

$x(t)$ is unstable if $|\mu_j| > 1$, for some j .

On the stable periodic orbit, the $n - 1$ multipliers are always inside the unit circle. The multipliers are the functions of the parameter under consideration. When we vary the parameter, some of the multipliers may cross the unit circle. The multiplier crossing the unit circle is called the critical multiplier. Different types of branching occur depending on where a critical multiplier or pair of complex conjugate multipliers leave the unit circle. Three associated types of branching are (i) the critical multiplier goes outside the unit circle along the positive real axis, with $|\mu(p_c)| = 1$, (ii) the multiplier goes outside the unit circle along the negative real axis with $|\mu(p_c)| = -1$ and (iii) a pair of complex conjugate multipliers crosses the unit circle with a non-zero imaginary part. All these types refer to a loss of stability when λ passes through λ_c . (On the other hand, if a critical multiplier enters the unit circle, the system gains stability.) In the case (i) typically, turning points of the periodic orbit occur with a gain or loss of stability. Transcritical or pitchfork type bifurcations in periodic orbits are also possible for this case. In the case (ii), the system oscillates with period two. In the case (iii), the phenomenon of bifurcation into a torus occurs, which is also called secondary Hopf bifurcation, or generalized Hopf bifurcation. The period doubling bifurcation often occurs repeatedly which generally leads to chaos. Lyapunov exponents are generally used to identify the chaos (Hao Bai Lin 1990). This exponential serves as a measure for exponential divergence or contraction of nearby trajectories. Chaos is characterized by at least one positive Lyapunov exponent, which reflects a stretching into one or more directions. In general, chaos has the following ingredients (Hao Bai Lin 1990): (i) the underlying dynamics is deterministic; (ii) no external noise has been introduced; (iii) seemingly erratic behaviour of individual trajectories depends sensitively on small changes of initial conditions; (iv) in contrast to a single trajectory, some global characteristics are obtained by averaging over many trajectories or over a long time (e.g., a positive Lyapunov exponent) that do not depend on initial conditions; (v) when a parameter is tuned, the erratic state is reached via a sequence of events, including the appearance of one or more subharmonics. In the last few years, a great number of conferences and workshops devoted to chaotic dynamics have been organized. In most of them, papers by researchers from various branches of science and engineering have been presented. Research in chaos is well documented by Hao Bai Lin (1990). Numerical methods to identify chaos can be found in Parker & Chua (1989). Observations of chaos in power systems are reported in Chiang *et al* (1992), Ajarapu & Lee (1992) and Nayfeh *et al* (1990). Figure 3 gives the overall possible bifurcation scenario.

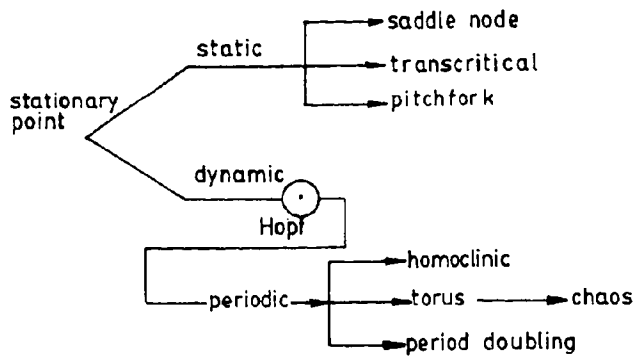


Figure 3. List of possible bifurcations.

6. Conclusions

The founding father of the bifurcation theory was the outstanding French mathematician, Henri Poincaré. Poincaré sketched a general bifurcation theory and qualitative dynamics. He was the first person to realize that simple deterministic systems can be unpredictable. Since that time, this theory has undergone tremendous development with infusion of new ideas and methods from dynamical system theory, singularity theory, group theory, and computer-assisted study of dynamics. If one varies the parameters of a dynamical system, the phase portrait of that system may deform slightly without altering its qualitative features, or sometimes the dynamics may be modified significantly, producing a qualitative change in the phase portrait. Bifurcation theory studies these qualitative changes in the phase portrait, e.g., the appearance, or disappearance of equilibria, periodic orbits, or more complicated features such as strange attractors. The methods and results of bifurcation theory are fundamental to an understanding of nonlinear dynamical systems, and the theory can be potentially applied to any area of nonlinear engineering systems. In power systems this theory received considerable attention especially with respect to understanding voltage collapse phenomena. Intensive research activity is going on with respect to theoretical as well as numerical aspects of bifurcation theory. The developments of this very interesting field will be helpful to natural scientists and engineers in their studies of nonlinear systems of both theoretical and practical interest. In this paper no attempt has been made to give a complete list of all the available references on the application of nonlinear dynamical system theory to engineering problems.

References

- Abed E H, Varaiya P P 1984 Nonlinear oscillations in power systems. *Int. J. Electr. Power Energy Syst.* 6: 37–43
- Abbot J P 1978 An efficient algorithm for the determination of certain bifurcation points. *J. Comput. Appl. Math.* 4: 19–27
- Ajarapu V 1991 Identification of steady state voltage stability in power systems. *Int. J. Energy Syst.* 11: 43–46
- Ajarapu V, Christy C 1992 The continuation power flow: A tool for steady state voltage stability analysis. *IEEE Trans. Power Syst.* 7: 416–423
- Ajarapu V, Lee B 1992 Bifurcation theory and its application to nonlinear dynamical phenomenon in an electric power system. *IEEE Trans. Power Syst.* 7: 424–431
- Alvarado F L, Jung T H 1989 Direct detection of voltage collapse conditions. *Proceedings. Bulk Power System Voltage Phenomenon – Voltage Stability and Security, EPRI EL-6183, Project 2473–21, Electric Power Research Institute*

- Arnold V I 1972 Lectures on bifurcation in versal families. *Russ. Math. Surv.* 27: 54–123
- Arnold V I 1983 *Geometric methods in the theory of ordinary differential equations* (New York: Springer)
- Blaquiere A 1966 *Nonlinear system analysis* (New York: Academic Press)
- Bruno A D 1989 *Local methods in nonlinear differential equations* (Berlin: Springer-Verlag)
- Carr J 1981 *Applications of center manifold theory. Applied Math. Sciences.* Vol. 35 (New York: Springer-Verlag)
- Chiang H D, Dobson I, Thomas R J, Thorp J S, Lazhar F A 1989 On voltage collapse in power systems. *Proceedings of the IEEE Power Industry Computer Application (PICA) Conference* (New York: IEEE)
- Chiang H D, Liu C W, Varaiya P P, Wu F F, Lauby M G 1992 Chaos in a simple power system. *IEEE/PES Winter Meeting, WM151-PWRS* (New York: IEEE)
- Giewank A, Reddien G 1983 The calculation of Hopf bifurcation by direct method. *Inst. Math. Appl. J. Numer. Anal.* 3: 295–303
- Guckenheimer J, Holmes P J 1983 *Nonlinear oscillations, dynamical systems, and bifurcation of vector fields* (New York: Springer-Verlag)
- Haken H 1983 *Synergetics, an introduction* 3rd edn (Berlin: Springer)
- Halvacek V (ed.) 1986 *Dynamics of nonlinear systems* (Gorden and Breach Science Publishers)
- Hao Bai Lin (ed.) 1990 *Chaos* (Singapore: World Scientific)
- Hassard B D, Kazirinnoff N D, Wan Y H 1981 *Theory and applications of Hopf bifurcation* (Cambridge: University Press)
- Holmes P J (ed.) 1980 *New approaches to nonlinear problems in dynamics* (Philadelphia: Siam)
- Jordan D W, Smith P 1977 *Nonlinear ordinary differential equations* (Oxford: University Press)
- Kubicek M, Marek M 1983 *Computational methods in bifurcation theory and dissipative structures* (New York: Springer-Verlag)
- Marsden J E, McCracken M 1976 *The Hopf bifurcation and its application* (New York: Springer)
- Moon F C 1987 *Chaotic vibration* (New York: Wiley)
- Moore G, Spence A 1980 The calculation of turning points of nonlinear equations. *SIAM J. Numer. Anal.* 17: 567–576
- Nayfeh M A, Hamdan A M A, Nayfeh A H 1990 Chaos and instability in a power system: primary resonant case. *Nonlinear Dynamics* 1: 313–339
- Parker T S, Chua L 1989 *Practical numerical algorithms for chaotic systems* (New York: Springer)
- Rajgopalan C, Lesieutre B, Sauer P W, Pai M A 1991 Dynamic aspects of voltage power characteristics. *IEEE/PES Summer Power Meeting, SM 419-2 PWRS* (New York: IEEE)
- Rajgopalan C, Sauer P W, Pai M A 1989 Analysis of voltage control system exhibiting Hopf bifurcation. *IEEE Proceedings of the 28th Conference on Decisions and Control* (New York: IEEE)
- Reyn J W 1964 Classification and description of the singular points of a system of three linear differential equations. *J. Appl. Math. Phys.* 15: 540–555
- Riks E 1979 An incremental approach to the solution of snapping and buckling problems. *Int. J. Solids Struct.* 15: 529–551
- Rheinboldt W C 1986 *Numerical analysis of parametrized nonlinear equations* (New York: John Wiley & Sons)
- Rheinboldt W C, Burkhardt J V 1983 A locally parametrized continuation process. *ACM Trans. Math. Software* 9: 215–235
- Roose D 1985 An algorithm for the computation of Hopf bifurcation points in comparison with other methods. *J. Comput. Appl. Math.* 12 & 13: 517–529
- Seydel R 1979 Numerical computation of branch points in nonlinear equations. *Numer. Math.* 33: 339–352
- Seydel R 1981 Numerical computation of periodic orbits that bifurcate from the stationary solution of ordinary differential equations. *Appl. Math. Comput.* 9: 257–271
- Seydel R 1988 *From equilibrium to chaos* (New York: Elsevier Science)
- Thomson J M T, Stewart H B 1986 *Nonlinear dynamics and chaos* (Chicester: Wiley)
- Wiggins S 1990 *Introduction to applied nonlinear dynamical systems and chaos* (New York: Springer Verlag)
- Vittal V, Kliemann W, Starrelt S K, Fouad A A 1992 Analysis of stressed power systems using normal forms. *Proceedings of IEEE International Symposium on Circuits and Systems* (New York: IEEE)