Instantaneous Relaxation-Based Real-Time Transient Stability Simulation

Vahid Jalili-Marandi, *Student Member, IEEE*, and Venkata Dinavahi, *Senior Member, IEEE*

***Abstract*—Real-time transient stability simulation is of paramount importance for system security assessment and to initiate preventive control actions before catastrophic events such as blackouts happen. Transient stability simulation of realistic power systems involves the solution of a large set of nonlinear differential-algebraic equations in the time-domain which requires significant computational resources. Exploitation of parallel processing techniques can provide an efficient and cost-effective solution to this problem. This paper proposes a fully parallel method known as instantaneous relaxation (IR) for real-time transient stability simulation. To validate the proposed method, two test systems have been implemented on an advanced PC-cluster-based real-time simulator. A comparison of the captured real-time results with those from the PSS/E software shows high accuracy.**

***Index Terms*—Iterative methods, parallel algorithms, power system transient stability, real-time systems, relaxation methods.**

I. Introduction

REAL-TIME digital simulation has become an integral part of the planning and design of power systems. It is also playing an important role in the operation of power systems by providing a true-to-life platform for online training of power system operators. The application of real-time simulators spans the entire spectrum of traditional power system studies ranging from steady-state to dynamic, and further to high-frequency electromagnetic transient studies. Computations that took significant time using traditional offline software programs can now be executed in real-time providing a realistic view of system behavior under changing conditions. In particular, when a new device such as a controller or relay needs to be tested and tuned in the hardware-in-the-loop (HIL) scenario for eventual implementation in the field, there is no alternative to a real-time simulator. Analog scaled-down simulator, also known as transient network analyzers (TNAs), were the predecessors of fully digital real-time simulators. However, realization of large-scale power systems with a high level of complexity, nonlinearity, and sophisticated dynamic elements using TNAs is practically impossible [1].

Manuscript received July 01, 2008; revised January 19, 2009. First published May 08, 2009; current version published July 22, 2009. This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC). Paper no. TPWRS-00517-2008.

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada (e-mail: v_jalili@ece.ualberta.ca; dinavahi@ece.ualberta.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TPWRS.2009.2021210

The objective of this paper is to revisit the application of real-time digital simulators to the transient stability problem. Transient stability simulation of realistic-size power systems involves computationally onerous time-domain solution of thousands of nonlinear differential algebraic equations (DAEs). Furthermore, from the point of view of dynamic security assessment which is required for safe system operation and control, several transient stability cases need to be run in a short period of time to initiate preventive control actions. Currently available commercial real-time simulators such as RTDS [2], RT-LAB from OPAL-RT Technologies, Inc. [3], and Hypersim [4] address these needs to a large extent by using multiple racks or clusters of multiprocessor architectures. The question that arises, however, is whether this approach is the most efficient and cost-effective, given the prevalent practice of using a real-time simulator, originally designed and built for electromagnetic transient studies, for transient stability simulations. This is done, of course, using simpler models and larger time-steps. For example, nominal-pi models are used instead of frequency-dependent models for transmission lines, and the simulator time-step is chosen to be in the range of milliseconds instead of microseconds. Nevertheless, there is underlying sequentiality in the electromagnetic transient simulation algorithm [5] that precludes an efficient utilization of the hardware resources for transient stability simulation. By exploiting parallelism inherent in the transient stability problem, a parallel solution algorithm can be devised to maximize the computational efficiency of the real-time simulator. This would reduce the cost of the required hardware for a given system size or increase the size of the simulated system for a fixed cost and hardware configuration.

In this paper we propose a fully parallel *instantaneous relaxation* (IR) method for real-time transient stability simulation. The idea of using relaxation-based solution of DAEs is certainly not new and has been explored before. The waveform relaxation (WR) method was first introduced in [6] for VLSI circuit simulation. Then in [7] this method was applied to the power systems area and used comprehensively for offline transient stability simulation [8]. The classical model of the synchronous machine was used in these simulations. Although this algorithm was implemented sequentially, it was predicted that it will accelerate the simulation by exploiting parallel processors [9]. Later in [10] the WR method was implemented on parallel computers.

As will be shown later, although the WR method is a parallel method successfully implemented for offline simulations, there are inefficiencies that surface when it is implemented in real-time. Therefore the IR method which overcomes these limitations is proposed for real-time implementation. To evaluate

0885-8950/$25.00 © 2009 IEEE

this method, two test cases are presented. PTI's PSS/E software package has been used to validate the real-time simulation results. The detailed model of the synchronous generator including AVR and PSS is used to demonstrate that the IR method converges even in the case of complex system models.

The paper is organized as follows: in Section II, there is a brief overview of the transient stability study in power systems with a discussion about methods for simulation of large-scale systems, and the WR technique is presented. The limitations of the WR method for real-time implementation are also discussed in this section. The algorithm of the proposed IR method is explained in Section III, and the approach for partitioning a power system for using the IR method is discussed in Section IV. Practical real-time simulation results and their comparative analysis with offline simulations using PSS/E are shown in Section V. Section VI presents the conclusion.

II. Background

A. Transient Stability Simulation

A widely used method for detailed modeling of synchronous generator for transient stability simulation is to use Park's equations with an individual dq reference frame fixed on the generator's field winding [11]. The network, including transmission lines and loads, is modeled using algebraic equations in a common DQ reference frame. Representation of AVR and PSS increases the number of differential equations and hence the complexity of the model. However, the validity of the dynamic response in a network with a lot of interconnections and in a time frame of few seconds highly depends on the accuracy of the generator model and other components which can have effects on the dynamics of the system. The general form of DAEs, which describe the dynamics of a multimachine power system assuming single-phase, positive sequence, and fundamental frequency behavior, is given as

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

$$\mathbf{0} = \mathbf{g}(\mathbf{x}, \mathbf{V}, t) \tag{2}$$

$$\mathbf{x}(t_0) = \mathbf{x_0} \tag{3}$$

where $\mathbf{x}$ is the vector of state variables, $\mathbf{x_0}$ is the initial values of state variables, and $\mathbf{V}$ is the vector of bus voltages. In this work the detailed model (the Appendix) of synchronous generator including AVR and PSS is used. Therefore, each generating unit consists of nine state variables. In a power network with m synchronous generators and n buses, $\mathbf{x}$ is a $9m \times 1$ vector and $\mathbf{V}$ is a $2n \times 1$ vector. The time frame of interest for the simulation of a transient stability event is in the order of 3–5 s following the disturbance. This time span may increase up to 10–20 s in the case of very large networks [12]. Typically a time-step in the range of milliseconds is chosen for simulation.

B. Solution of Large-Scale Systems

A common approach for time-domain simulation of a system, described by a set of nonlinear DAEs, consists of three steps: an integration method (e.g., trapezoidal rule) for discretizing the differential equations, an iterative method (e.g., Newton–Raphson) for solving the nonlinear algebraic equations, and a linear equation solver such as Gaussian elimination and back substitution. This approach is referred to as the *standard* or *direct* simulation approach [13]. Both the storage and cpu time required by the standard approach grow rapidly as the size of the system, measured in terms of its components (i.e., generators in the case of a power system), increases. Moreover, in a large system of DAEs, different variables change at various rates. In the standard approach, the integration method is forced to discretize all the differential equation with the same time-step which must be small enough to capture the fastest dynamics in the system. As such, simulating realistic-size large-scale systems using the standard approach became very time consuming. To address this problem, a family of techniques known as *domain decomposition* was developed. Domain decomposition refers to any technique that divides a system of equations into several subsets that can be solved individually using conventional numerical methods. Parallel processing can be exploited to find the solution of a large equation set. To solve a set of nonlinear DAEs, domain decomposition can be applied at any of the three levels of equations, i.e., differential equations, nonlinear algebraic equations, and linear algebraic equations.

Two different approaches were proposed in the literature to perform domain decomposition: *tearing* and *relaxation*. Tearing (introduced as the *diakoptics* method by Kron [14]) is the approach that takes advantage of the block structure of the system of equations. To describe the structure of a system, the notion of the *dependency matrix* (D) is used. For a system with n equations and n unknown variables, D is an $n \times n$ matrix whose elements are 1 or 0. If the ith equation involves the jth variable, then $D(i,j)$ is 1; otherwise, $D(i,j) = 0$. For a system of equations in which the dependency matrix is sparse, i.e., D has a small percentage of 1's, tearing can be used to achieve decomposition while maintaining the numerical properties of the method used to solve the system. The bordered blocked diagonal (BBD) form is one specific structure suitable for this approach. Tearing decomposition at the level of linear algebraic equations can be implemented as the *block LU factorization* method, and at the level of nonlinear algebraic equations as the *multilevel Newton–Raphson* method [15]. It should be noted that the computational efficiency of this approach over the standard approach depends critically on the structure of the system, and it does not increase when system dependency matrix is dense.

Relaxation [16] is an approach which is not restricted to a particular system structure. In this approach, the system is partitioned into a number of subsystems based on either the system equations or component connectivity. Solving these subsystems is always easier than solving the original system. Therefore, the complexity will be reduced regardless of the system sparsity. Two well-known types of relaxation decomposition are the Gauss–Seidel and Gauss–Jacobi methods [17]. The application of this approach for nonlinear algebraic equations can be found in [18] and [19]. Relaxation can be used at the level of differential equations as well. In this case, the system is broken into subsystems in a way that the components inside of each subsystem are strongly interdependent while the dependency between components in two different subsystems is weak enough to ignore their interconnection. In other words, the subsystems can be *relaxed*. Therefore, each part of the system is still a system of differential equations but with a smaller size that can be solved in the time-domain using the standard approach. The relaxation approach applied to the differential equations' level has been known as the waveform relaxation (WR) method. As will be shown in Section II-C, the efficiency of the WR method is significantly reduced when implemented in real-time. It is required to alter this technique to make it suitable for real-time simulation. To explain the reason for these changes, the WR method is briefly described using a simple example.

C. Waveform Relaxation

Consider a first-order two-dimensional system described by the following differential with known initial values:

$$\dot{x}_1 = f_1(x_1, x_2, t), \quad x_1(0) = x_{10} \tag{4}$$

$$\dot{x}_2 = f_2(x_1, x_2, t), \quad x_2(0) = x_{20} \tag{5}$$

where $x(t) \in \mathbb{R}^2$ and $t \in [0, T]$. To solve this set of equations by the WR method, first (4) is solved for $x_1(t)$ as a one-dimensional differential equation for all $t \in [0, T]$ with $x_2(t)$ fixed. Similarly, (5) is solved for $x_2(t)$ with $x_1(t)$ fixed for all $t \in [0, T]$. The next iteration begins with the substitution of $x_2(t)$ obtained from (5) in the previous iteration into (4), and $x_1(t)$ obtained from (4) in the previous iteration into (5). This procedure is repeated until differences between waveforms obtained from two consecutive iterations is less than a small preset number. Further details about this method and its convergency conditions can be found in [13].

Within each subsystem, the variables to be solved are called *internal* variables, e.g., $x_1(t)$ in (4) and $x_2(t)$ in (5), while the other variables which are fixed during each iteration are called *external* variables, e.g., $x_2(t)$ in (4) and $x_1(t)$ in (5). This algorithm is known as the Gauss–Jacobi waveform relaxation (GJWR), because in all subsystems, for each new iteration, external variables from the previous iteration are being used. If in each iteration, after finding the solution of each subsystem, the external variables are updated for other subsystems, the method is known as the Gauss–Seidel waveform relaxation (GSWR). It is clear that GJWR method is an inherently parallel method that can be implemented on parallel hardware. Unlike other methods which are parallel only in the sense of numerical task partitioning for solving the nonlinear or linear algebraic equations, the parallelization existing in the GJWR method appears in the whole body of the algorithm. Another advantage of this method is that it allows the simulator to exploit the stiffness of the system. Each subsystem can be discretized with an individual time-step determined based on the fastest dynamic in that subsystem [8]. Therefore the GJWR method is chosen in this paper for real-time simulation. The WR method, however, has some caveats for real-time implementation as discussed below.

D. Limitations of Waveform Relaxation for Real-Time Simulation

The outstanding difference between the WR and other classical decomposition methods for solving linear and nonlinear algebraic equations is that in this method, during each iteration, each subsystem is analyzed for the entire time interval, $[0, T]$. In other words, elements in this technique are waveforms of the variables rather than their instantaneous values. In each iteration of the WR method, each subsystem is solved by using the three basic steps of the standard approach for all $t \in [0, T]$.

It was discussed in [9] and [20] that the WR method works well for a certain interval, but it is inaccurate outside of this span. So, instead of applying the method in each iteration over the whole simulation time, i.e., $[0, T]$, it is more effective to divide the simulation time into k small intervals or windows, i.e., $[0, T_1], [T_1, T_2], \ldots, [T_k, T]$, and solve equations piece by piece within each interval. This technique, known as *windowing*, decreases the number of iterations required within each interval for achieving required accuracy [13]. Furthermore, windowing reduces the required memory space, because the iterative waveforms need to be stored only for small time intervals instead of the whole simulation time.

The waveform-based property of the WR method is one issue that needs to be changed for real-time simulation. There are two reasons. First, in real-time simulation and specifically in hardware-in-the-loop simulation, the instantaneous value of each variable at each time-step is required and not the complete waveforms. Second, if waveforms are going to be used as numerical elements, all waveforms of the variables such as bus voltages or generator angles must be computed for the entire simulation interval, say, 20 s, in the first time-step of the simulation, say, 1 ms. Clearly, this is not practical for a large-scale system with thousands of variables. To overcome this restriction, the windowing technique can be used. So, the whole simulation time is divided into small intervals, and each interval is computed in one time-step. The time intervals must be small enough so that the computation tasks can be performed during one time-step. Although windowing can help maintain the waveform property, however, working with waveforms in real-time is not as efficient as in offline simulation.

Suppose the simulation interval $[0, T]s$ is divided into k windows of length $m \times h$ milliseconds, h being the time-step. When the simulation starts, all waveforms for the interval of the first window must be computed during the first time-step. Then, there are two options. In the first option (Fig. 1), the real-time simulator is idle during the remaining length of the first window, i.e., for $(m-1) \times h$ milliseconds, when it just sends out instant values of variables at each time-step. After this period, simulator resumes computation for the interval of the second window, and again becomes idle. This process is repeated until the end of simulation time. In the second option, depicted in Fig. 2, the simulator continues the computation for each window in the subsequent time-steps while it also sends out the instantaneous values of variables at each time-step. Therefore, the computation finishes in k consecutive time-steps, and after that, the simulator becomes idle when it sends out instant values at each time-step. It can be concluded that in both options, the simulator performs

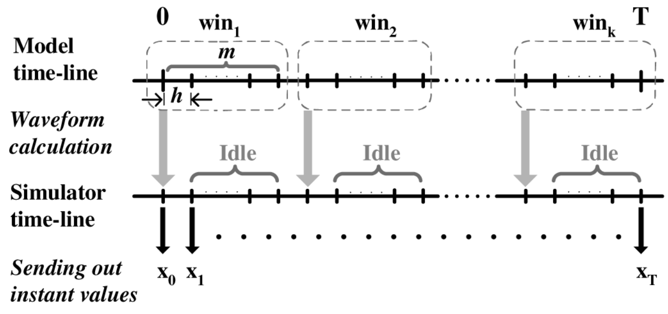

Fig. 1. Real-time implementation of the WR method: Option 1.

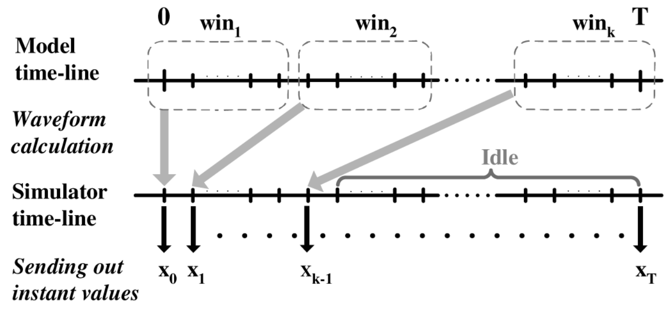

Fig. 2. Real-time implementation of the WR method: Option 2.

the entire computation in k time-steps and then remains idle for $(m-1)\times k$ time-steps. In other words, the computation load has not been distributed among the time-steps equally. Thus, real-time implementation of the native WR method can be inefficient from resource utilization point of view.

III. Instantaneous Relaxation for Real-Time Transient Stability Simulation

To overcome the limitations of the WR for real-time implementation, we propose the *point-wise* or IR technique. It is simply the WR method with a window length of one time-step, i.e., $m = 1$. It has been verified in the offline implementation of WR method that the smaller the length of window, the faster the convergence. On the other hand, if the window is made too small, the overall communication time among subsystems increases which causes a loss of the advantages of the WR method. However, the communication latency between computation nodes in currently available real-time simulators is in the order of a few microseconds. This latency is small compared to the time-step required for transient stability and can therefore be neglected. Based on the previous experience with the WR method and the arguments made in this paper, it can be concluded that the IR method not only inherits the advantages of the WR method but is also efficient from the real-time simulation point of view.

To apply relaxation methods at the level of differential equations, the preliminary step is clustering variables into groups which can be solved independently. This will be specifically discussed for the transient stability application in Section IV. After partitioning the system into n subsystems, the set of DAEs equations [i.e., (6) and (7)] are prepared to describe the dynamics of each subsystem as follows:

$$\dot{\mathbf{x}}^i = \mathbf{f}^i(\mathbf{x}^i, \mathbf{V}^i, t) \tag{6}$$

$$\mathbf{0} = \mathbf{g}^i(\mathbf{x}^i, \mathbf{V}^i, t) \tag{7}$$

$$\mathbf{x}^i(t_0) = \mathbf{x_0}^i \tag{8}$$

where $i = 1, 2, \ldots, n$ indicates the subsystem. Discretizing (6) results in a new set of nonlinear algebraic equations. In this work we used the trapezoidal rule as the implicit integration method to discretize the differential equations as follows:

$$0 = \mathbf{x}^i - \frac{h}{2}\left[\mathbf{f}^i(\mathbf{x}^i, \mathbf{V}^i, t) + f^i(\mathbf{x}^i, \mathbf{V}^i, t-h)\right] \tag{9}$$

where h is the integration time-step. Equations (7) and (9) can be linearized by the Newton–Raphson method (for the jth iteration) as

$$J\left(\mathbf{z}^i_{j-1}\right)\cdot\Delta\mathbf{z}^i = -\mathbf{F}^i\left(\mathbf{z}^i_{j-1}\right) \tag{10}$$

where J is the Jacobian matrix, $\mathbf{z}^i = [\mathbf{x}^i, \mathbf{V}^i]$, $\Delta\mathbf{z}^i = \mathbf{z}^i_j - \mathbf{z}^i_{j-1}$, and $\mathbf{F}^i$ is the vector of nonlinear function evaluations. Equation (10) is a set of linear algebraic equations that can be solved with Gaussian elimination and back substitution method. Benchmarking revealed that a majority of execution time in a transient stability simulation is spent for the nonlinear solution. By using the IR method, however, and by distributing the subsystems over several parallel processors, a large-scale system is divided into individual subsystems whose matrix sizes are smaller, resulting in faster computations.

To clarify the differences between the WR and IR methods, the flowcharts of both algorithms are shown in Figs. 3 and 4, respectively. Practically in the WR method, it does not seem efficient to perform several iterations of the Newton–Raphson. Let the exact solution of a waveform be $x(.)$, and the result of the kth iteration of the WR method be $x^k(.)$. Depending on the length of window, some iterations will be required for $x^k(.)$ to converge to $x(.)$; however, the starting iterations for $x^k(.)$ are poor approximations of $x(.)$. Thus, it is superfluous to perform Newton–Raphson iterations for computing a close approximation to $x^k(.)$ which itself is a poor approximation of $x(.)$. The convergence rate of the IR method is higher than that of WR, because its window length is minimum. Therefore, performing several iterations of Newton–Raphson, as shown in Fig. 4, increases the accuracy of IR.

Following the convergence of iterative solutions in all subsystems, the state and algebraic variables calculated from the last time-step are updated. The state variable description of generators used in this work is defined in the Appendix. These state variables and voltages of generator buses must be exchanged between all interconnected subsystems.

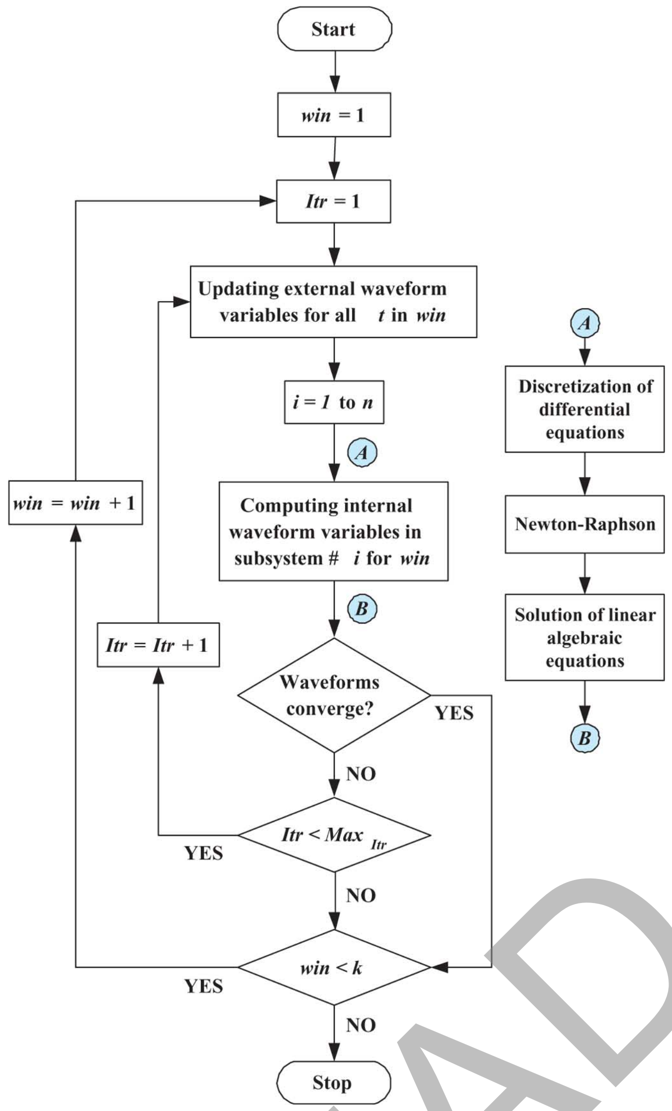
Start
win = 1
Itr = 1
Updating external waveform variables for all t in win
i = 1 to n
A
Computing internal waveform variables in subsystem # i for win
B
Waveforms converge?
YES
NO
Itr < Max Itr
YES
NO
Itr = Itr + 1
win = win + 1
win < k
YES
NO
Stop
A
Discretization of differential equations
Newton-Raphson
Solution of linear algebraic equations
B

Fig. 3. Flowchart of the WR method for the duration of $[0, T]$ which is divided into k window intervals. n: the total number of subsystems; *Itr*: counter of iterations in each window; Max_{Itr}: the maximum allowable number of iterations in each window; *win*: counter for windows.

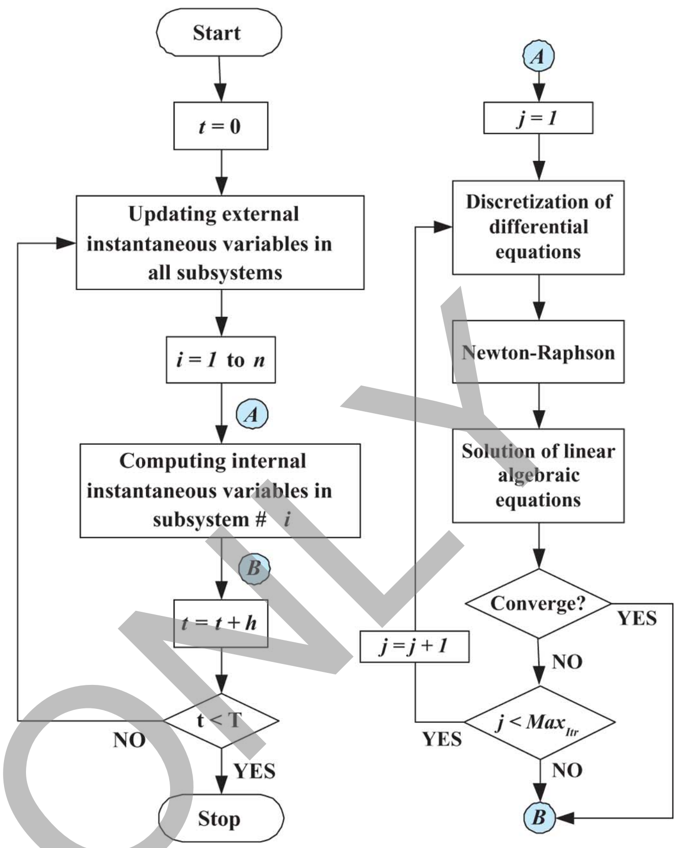
Start
t = 0
Updating external instantaneous variables in all subsystems
i = 1 to n
A
Computing internal instantaneous variables in subsystem # i
B
t = t + h
t < T
NO
YES
Stop
A
j = 1
Discretization of differential equations
Newton-Raphson
Solution of linear algebraic equations
Converge?
YES
NO
j = j + 1
j < Max Itr
YES
NO
B

Fig. 4. Flowchart of the proposed IR method for the duration of $[0, T]$ with a time-step of h. j: counter for iterations of the Newton–Raphson; Max_{Itr}: the maximum allowable number of iterations for the Newton–Raphson in each time-step.

IV. Coherency-Based System Partitioning for Instantaneous Relaxation

One way to partition a power system for parallel processing is to distribute equal numbers of generators and buses among the processors. However, this is not an efficient method because the network buses have different connectivity and the generator models vary in both size and complexity. This gives rise to the load balancing problem in a parallel multiprocessor simulator architecture. Another option is to split the computation burden among processors based on the total number of equations; however, this approach will increase both the programming and communication complexity. A more efficient method is to partition the system by considering the complexity of the generator models and the connectivity of the buses. In this case, different number of generators and buses are assigned to parallel processors, and the computation burden is roughly even; however, the drawback of this method is that it cannot be used for a general-purpose program, at least in the offline sense [21].

The primary requirement for successfully using the relaxation methods at the level of differential equations is to divide the system into subsystems in which tightly coupled variables are grouped together. In [13], it was shown that the WR method will converge for any chosen partitioning scheme; however, the rate of convergence is highly dependent on the method of partitioning [9]. In spite of all that, it is important to ask this question: is transient stability simulation of a large-scale network by either IR or WR methods restricted by this prerequisite? In other words, whether the partitioning scheme's dependence on system modeling or the characteristics of the disturbance such as its severity or location, will influence the convergence of the IR or WR methods.

Determination of tightly coupled variables or simply partitioning the system can find a physical meaning from the power system point of view. Following a large disturbance in the system, some generators lose their synchronism with the network. Thus, the system is naturally partitioned into several areas in which generators are in step together while there are oscillations among the different areas. Generators in each of these areas are said to be *coherent*. The coherency characteristic of the power system reflects the level of dependency between generators. Coherent generators can be grouped in the same subsystem which can be solved independently from other subsystems with the WR or IR methods. The partitioning achieved using the coherency property has two characteristics

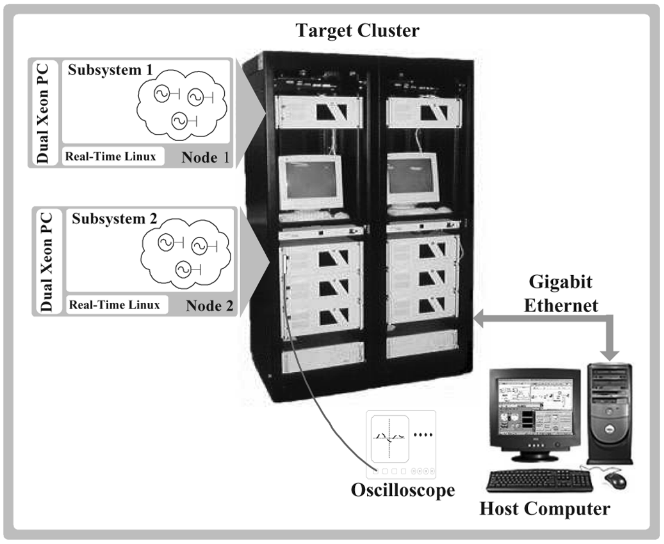

Fig. 5. Configuration of the real-time simulator.

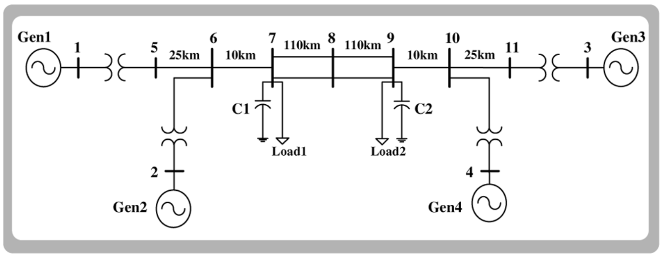

Fig. 6. One-line diagram for Case Study 1.

which make it appropriate for our study. The coherent groups of generators are independent of: 1) the size of disturbance and 2) the level of detail used in the generators. Therefore, the linearized model of the system and the simple classical model of generators can be used to determine coherency. Furthermore, slow coherency-based grouping is insensitive to the location of disturbance in the power system [22]. These features of slow coherency lead us to use this partitioning method in this paper.

V. Experimental Results

In this section, we will demonstrate results to verify the efficiency of the IR method for real-time simulation. To do so, we have chosen two case studies. One is the Kundur's four-machine and 11-bus system found in [23]. The other case study is the IEEE 39-bus New England test system [24]. The real-time results for these case studies have been validated using the PSS/E software program.

A. Real-Time Simulator Structure

A PC-cluster-based real-time simulator (Fig. 5) in the RTX-LAB at the University of Alberta is used for the implementation. The *Host* computer running on Windows XP is used as the console for result visualization and online parameter control. The IR method is coded in C and compiled using MATLAB's Real-Time Workshop and OPAL-RT's RT-LAB software [25]. The executable code is loaded onto the *Target Cluster* nodes to run in real-time. Each cluster node consists of a dual *Intel® Xeon™* shared-memory PC running at 3.0 GHz on a real-time Linux operating system. The inter-node communication is through InfiniBand with a 10 Gb/s data transferring rate. The target nodes are capable of eXtreme High Performance (XHP) mode execution, in which one CPU is dedicated entirely to program execution while the other CPU is running operating system tasks and managing the communication schedules with the host computer and other target nodes. The host computer and the target cluster are connected using Gigabit ethernet.

As shown in Fig. 5, a study system can be partitioned and distributed among the cluster nodes. In the case of using multiple nodes, one node is the Master and other nodes operate as Slaves. The real-time results can be recorded on an oscilloscope connected to the FPGA-based I/O card on the target node or they can be visualized on the host computer screen.

B. Case Study 1

Fig. 6 illustrates the test system used as the first case study. Each synchronous generator is equipped by an exciter and PSS. A set of six differential equations model mechanical rotation, field winding, and three damper windings of each synchronous generator. The complete system can be described by 36 nonlinear differential and eight algebraic equations. Since generators $\{1,2\}$ and $\{3,4\}$ are coherent, the system can be partitioned into two subsystems. This coherency relation can also be observed later in the simulation results. These two subsystems are distributed across two cluster nodes as seen in Fig. 7. The simulation time-step is chosen to be 1 ms. Using the IR method, once the steady-state has been reached, a three-phase fault at Bus 8 is imposed at $t = 5$ s and is cleared in 80 ms. The real-time simulation results are recorded on an external oscilloscope and saved. The relative machine angles are shown in Fig. 8 in which *Gen4*'s angle is selected as the reference. The real-time results are superimposed on the results found from PSS/E. As can be seen, the IR method is quasi-stable during the steady-state of the system, i.e., $t < 5$ sec. During the transient state and also after the fault is cleared, the real-time results closely follow the results from PSS/E. The maximum discrepancy between real-time simulation and PSS/E was found to be 0.93%, based on the following:

$$\varepsilon_\delta = \frac{\max|\delta_{PSS/E} - \delta_{IR}|}{\delta_{PSS/E}} \tag{11}$$

where $\delta_{PSS/E}$ and δ_{IR} were defined as the relative machine angles from PSS/E and IR method, respectively. It can be further observed from Fig. 8 that following the fault, the oscillations of *Gen4* are closer to the oscillations of *Gen3* rather than to those of *Gen1* or *Gen2*. If the reference generator is switched to *Gen1* or *Gen2* instead of *Gen4*, it was found that the oscillations of *Gen1* are closer to the oscillations of *Gen2* rather than to those of *Gen3* or *Gen4*. This observation practically demonstrates the coherency relation existing in this system.

To investigate the effect of the fault location and the partitioning scheme on the performance of the IR method, the following scenario was simulated. Suppose that the fault happens at Bus 5. Two different patterns of partitioning have been applied for this case. The first is based on the coherency property

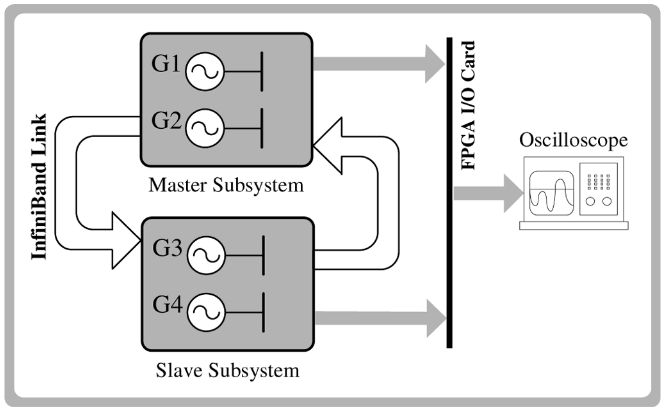

Fig. 7. Distribution of subsystems of Case Study 1 among cluster nodes of the real-time simulator.

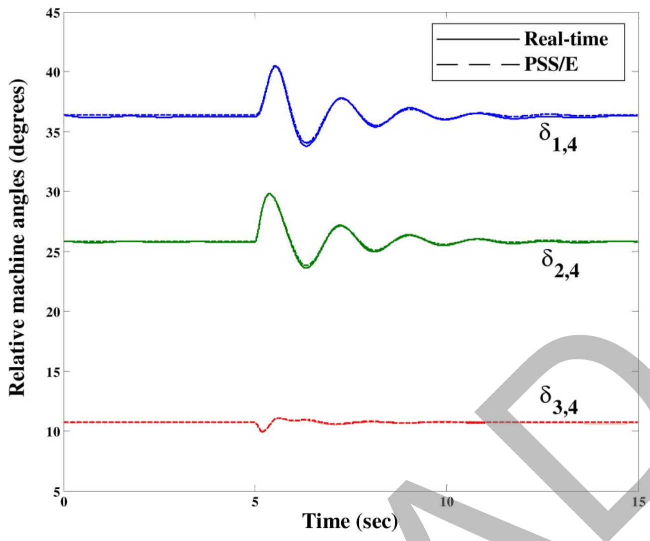

Fig. 8. Comparison of relative machine angles collected from real-time simulator and PSS/E simulation for Case Study 1: $\delta_{i,4} = \delta_i - \delta_4$; $i = 1, 2, 3$.

of the system, i.e., $\{1, 2\}$ and $\{3, 4\}$. The second pattern is based on the fact that since $G1$ is the closest generator to the fault location, it will accelerate faster than other generators in the system; therefore, the system is divided into two subsystems: $\{1\}$ and $\{2, 3, 4\}$. These two patterns have been simulated in real-time using the IR method, and then the results were compared with those of PSS/E. Results of both patterns are close to those from PSS/E, but the maximum error in the second pattern is larger than the maximum error when coherency-based partitioning was used. Several other combinations of fault location and partitioning patterns have also been examined in this system, and it was concluded that slow coherency partitioning-based IR method are the closest to the PSS/E's results.

Table I shows the timing performance of Master and Slave nodes of the real-time simulator running under the XHP execution mode during one time-step (1 ms). This table shows that the tasks of computation and communication are done in less than 60 μs. These execution times were sampled across many time-steps, and it was found that the idle times of the processors were uniform throughout those time-steps.

TABLE I
PERFORMANCE LOG FOR REAL-TIME SIMULATION OF CASE STUDY 1

	Duration (μs)	
Task	Master	Slave
Computation	47.91	47.20
Communication	11.55	8.12
Idle Time	939.45	940.07
Other	1.09	4.61
Total Step Size	1000	1000

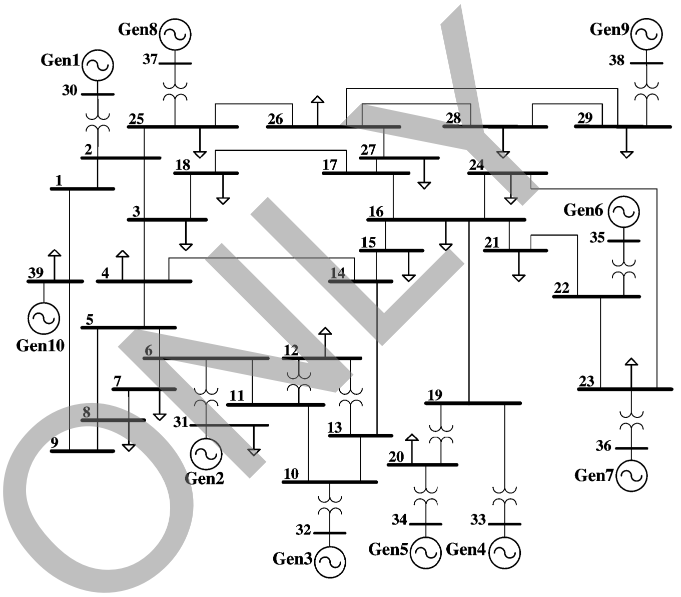

Fig. 9. One-line diagram for Case Study 2.

C. Case Study 2

The one-line diagram of IEEE's New England test system is shown in Fig. 9. As in the previous case study, all generator models are detailed and equipped with AVR and PSS. The complete system can be described by 87 nonlinear differential and 20 algebraic equations. Using the partitioning pattern mentioned in [24], the system has been divided into three subsystems: $\{1, 8, 9\}$, $\{2, 3, 4, 5, 6, 7\}$, and $\{10\}$. These three subsystems were distributed on three cluster nodes of the real-time simulator: one Master and two Slaves. A question which may arise here is about the uneven loading of CPUs. Although it is possible to add $\{10\}$ to subsystem 1 and to use only two computation nodes, our intent in this study was to demonstrate the implementation of IR in three parallel cluster nodes. Several fault locations have been tested and the results were compared with those of PSS/E; in all cases, results from the IR method match very well. In this section, a sample of these results are presented. A three-phase fault happens at Bus 21, at $t = 1$ s and it is cleared after 100 ms. *Gen10* is the reference generator and the relative machine angles are shown in Figs. 10 and 11. The maximum deviation of IR real-time simulation result from the PSS/E result based on (11) is 1.51%.

Table II shows the timing performance of Master and two Slave nodes in the real-time simulation during one time-step (1 ms). Again the sampled idle times were found to be uniform across several time-steps. In the PC-cluster architecture, the Master node is responsible for communicating with the host

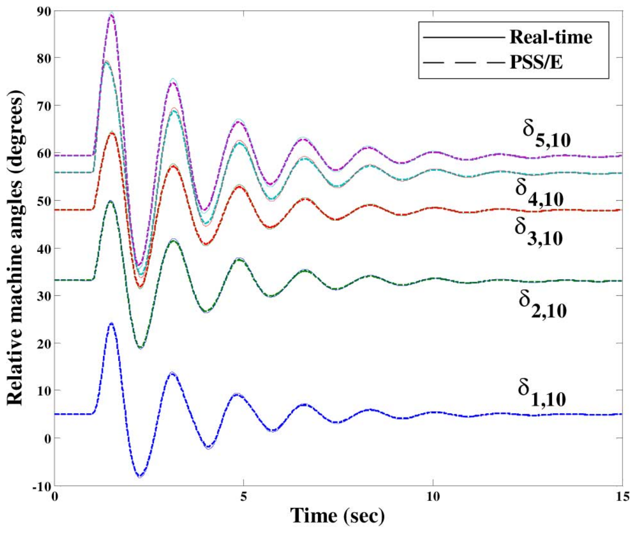

Fig. 10. Comparison of relative machine angles collected from real-time simulator and PSS/E simulation for Case Study 2: $\delta_{i,10} = \delta_i - \delta_{10}; i = 1 \ldots 5$.

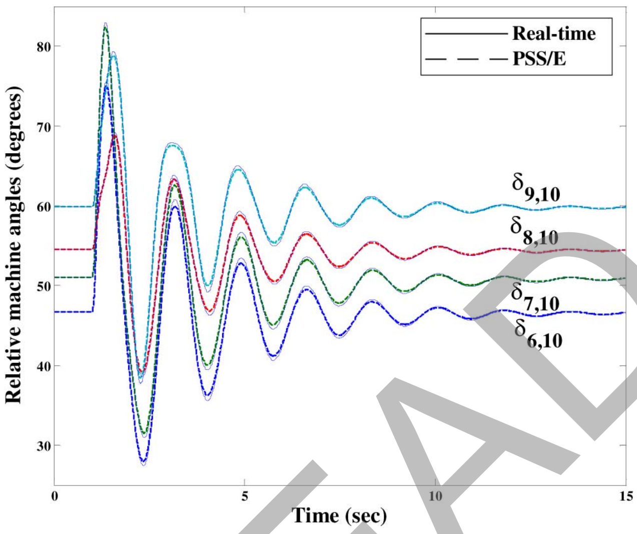

Fig. 11. Comparison of relative machine angles collected from real-time simulator and PSS/E simulation for Case Study 2: $\delta_{i,10} = \delta_i - \delta_{10}; i = 6 \ldots 9$.

computer and also for organizing the communication among the Slaves. This explains why the Master's communication time in Tables I and II is larger than Slaves' communication time. Moreover, it can be seen that the computation time in both case studies is not very high, since the computation load is distributed equally among all time-steps. From the idle time duration in both tables, it is concluded that larger subsystems can be implemented on each node, and that faster-than-real-time simulation is also possible.

The accuracy of the IR method is analyzed by varying the time-step and calculating the error in (11). The results are presented in Table III. As expected, when the time-step increases, the computation error increases as well. Nevertheless, it can be predicted that with larger time-steps, larger systems can be simulated on this hardware using the IR method. It can however be seen that the maximum error depends not only on the time-step but also on the size of the system. For instance, the time-step of

TABLE II
PERFORMANCE LOG FOR REAL-TIME SIMULATION OF CASE STUDY 2

	Duration (μs)		
Task	Master	Slave1	Slave2
Computation	212.77	348.13	17.27
Communication	13.44	7.30	4.51
Idle Time	770.93	631.73	974.12
Other	2.86	12.84	4.1
Total Step Size	1000	1000	1000

TABLE III
RELATION BETWEEN TIME-STEP AND ACCURACY OF THE IR METHOD

	Maximum error ε_δ%	
Time-Step (ms)	Case study 1	Case study 2
1	0.93	1.51
2	1.04	1.70
5	1.32	3.29
10	1.88	4.2

5 ms results in the maximum error of 1.32% and 3.29% in the four- and ten-generator systems, respectively. This dependence of computation error on the system size is counterintuitive and needs to be verified for larger test cases.

D. Speed-Up Comparison

In this section, we evaluate the speed-up achieved by the IR method in comparison with the standard method. To do so, a test program based on the standard approach discussed in Section II-B was created and its total simulation time was compared with that of the IR method. Since the standard method is sequential, to make a fair comparison, the IR method was also implemented sequentially. Fig. 12 lists steps of these two methods to be run on a 2.5-GHz quad-core AMD Phenom CPU supported by 4 GB of RAM. Except the hardware, the specifications of test case studies are the same as explained in the previous sections. The time-step of both methods is 1 ms. The standard and IR methods total time to simulate a duration of 15 s for the two case studies are summarized in Table IV. Speed-up is defined as the ratio between the computation time of the standard method to that of the IR method. The maximum error defined by (11) for both methods is similar. This comparison revealed two results: 1) on the same simulator hardware, the serial IR method is faster than the standard method for transient stability study, and 2) the speed-up of the serial IR method increases with the size of the system.

VI. CONCLUSION

This paper presents a parallel method known as IR for the real-time transient stability simulation of power systems. Although it is possible to utilize real-time simulators based on the electromagnetic transient simulation approach to perform transient stability analysis, the size and cost of the simulator is usually prohibitive, especially for simulating large-scale systems. The motivation behind this work is to test the real-time feasibility of a fully parallel method that could alleviate these limitations. The WR method was investigated in this paper for implementation in real-time. It was, however, found that WR method has some restrictions for real-time simulation due to the following reasons:

```
System Initialization ;
Start Timer ;
Do for each time step:
    Repeat until converged:
        Discretizing
        NR & Linear Solution;
        Updating;
    End of Repeat;
End of Do ;
Stop Timer ;
```

```
System Initialization ;
System Partitioning;
Start Timer ;
Do for each time step:
    Solve Subsystem 1;
    Solve Subsystem 2;
    ⋮
    Solve Subsystem n;
    Updating;
End of Do ;
Stop Timer ;
```

Fig. 12. Pseudo code for sequential implementation of (left) standard and (right) IR methods for transient stability computations.

TABLE IV
SPEED-UP COMPARISON

Case study	Total time (s) Standard method	IR method	Speed-up
1	8	3	2.67
2	85	25	3.4

- The WR method provides a set of values in the form of a complete waveform. However, real-time simulation, especially hardware-in-the-loop simulation, requires instantaneous values of variables.
- Implementation of the WR causes uneven computation loads among the time-steps. This results in execution time overrun in some time-steps and excessive idling time in the others. An overrun, which describes a situation when the simulator requires a larger time-step than the specified fixed time-step to finish its task, is not acceptable in hard real-time systems.

These problems are overcome by the proposed IR method. It inherits all the advantages of the WR method but is also efficient for real-time implementation. The two main differences between the IR and WR methods are as follows:

- In the IR method, the instantaneous values of the variables are being used and not their waveforms.
- To achieve the required accuracy, several iterations of the Newton–Raphson within each time-step are performed.

To demonstrate the performance of the IR method, two case studies have been implemented on a PC-cluster-based real-time simulator and the results are validated by the PSS/E software. Several comparisons verified the accuracy and efficiency of the IR method. To verify that the efficiency of the proposed method is not mainly due to the computing capacity of the PC-cluster, a sequential implementation of the IR method was compared with respect to the standard method. This evaluation revealed significant speed-up of the IR method for both case studies. In addition, the performance of the slow coherency method as the partitioning tool was analyzed, and it was concluded that for different fault locations in the system, results derived from this method had lower amounts of error.

APPENDIX
SYSTEM REPRESENTATION FOR THE TRANSIENT STABILITY ANALYSIS

The detailed model of a synchronous generator used in this paper is given here.

1) Equations of motion (swing equations or rotor mechanical equations): In transient stability studies, it is assumed that mechanical torque (T_m) is constant during the transient phenomena, and is the negative of the steady-state value of the electrical torque $(T_m = -T_e(0))$. Therefore, the turbine and governor systems are not modeled for the transient duration:

$$\begin{aligned}\dot{\delta}(t) &= \omega_R.\Delta\omega(t)\\ \dot{\Delta\omega}(t) &= \frac{1}{2H}[T_e(t) + T_m - D.\Delta\omega(t)].\end{aligned} \tag{12}$$

2) Rotor electrical circuit equations: This model includes two windings on the d axis (one excitation field and one damper) and two damper windings on the q axis:

$$\begin{aligned}\dot{\psi}_{fd}(t) &= e_{fd}(t) - R_{fd}i_{fd}(t)\\ \dot{\psi}_{1d}(t) &= -R_{1d}i_{1d}(t)\\ \dot{\psi}_{1q}(t) &= -R_{1q}i_{1q}(t)\\ \dot{\psi}_{2q}(t) &= -R_{2q}i_{2q}(t).\end{aligned} \tag{13}$$

3) Excitation system: Fig. 13 shows a bus-fed thyristor excitation system, classified as type *ST1A* in the IEEE standard [26]. This system includes an AVR and PSS:

$$\begin{aligned}\dot{v}_1(t) &= \frac{1}{T_R}[v_t(t) - v_1(t)]\\ \dot{v}_2(t) &= K_{stab}.\Delta\omega(t) - \frac{1}{T_w}v_2(t)\\ \dot{v}_3(t) &= \frac{1}{T_2}[T_1\dot{v}_2(t) + v_2(t) - v_3(t)].\end{aligned} \tag{14}$$

4) Stator voltage equations:

$$\begin{aligned}e_d(t) &= -R_a i_d(t) + L''_q i_q(t) - E''_d(t)\\ e_q(t) &= -R_a i_d(t) - L''_d i_d(t) - E''_q(t)\end{aligned} \tag{15}$$

where

$$\begin{aligned}E''_d &\equiv L_{aq}\left[\frac{\psi_{q1}}{L_{q1}} + \frac{\psi_{q2}}{L_{q2}}\right]\\ E''_q &\equiv L_{ad}\left[\frac{\psi_{fd}}{L_{fd}} + \frac{\psi_{d1}}{L_{d1}}\right].\end{aligned} \tag{16}$$

5) Electrical torque:

$$T_e = -(\psi_{ad}i_q - \psi_{aq}i_d) \tag{17}$$

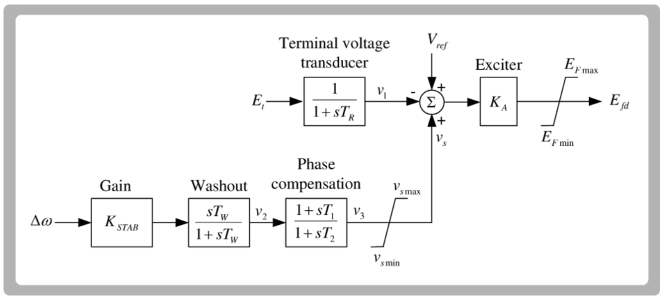

Fig. 13. Excitation system with AVR and PSS.

where

$$\psi_{ad} = L''_{ad}\left[-i_d + \frac{\psi_{fd}}{L_{fd}} + \frac{\psi_{d1}}{L_{d1}}\right]$$
$$\psi_{aq} = L''_{aq}\left[-i_q + \frac{\psi_{q1}}{L_{q1}} + \frac{\psi_{q2}}{L_{q2}}\right] \quad (18)$$

where $\omega_R, H, D, R_{fd}, R_{1d}, R_{1q}, R_{2q}, R_a, L_{fd}, L_{d1}, L_{q1}, L_{q2}, L''_d, L''_q, L_{ad}, L_{aq}, L''_{ad}, L''_{aq}, T_R, T_w, T_1, T_2$, and K_{stab} are constant system parameters whose definition can be found in [23].

According to this formulation the vector of state variables in (1) and (2) of the synchronous generator is

$$x = [\delta \quad \Delta\omega \quad \psi_{fd} \quad \psi_{1d} \quad \psi_{1q} \quad \psi_{2q} \quad v_1 \quad v_2 \quad v_3]^t. \quad (19)$$

References

[1] D. Jakominich, R. Krebs, D. Retzmann, and A. Kumar, "Real time digital power system simulator design considerations and relay performance evaluation," *IEEE Trans. Power Del.*, vol. 14, no. 3, pp. 773–781, Jul. 1999.

[2] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real time digital simulator for testing relays," *IEEE Trans. Power Del.*, vol. 7, no. 1, pp. 207–213, Jan. 1992.

[3] L.-F. Pak, M. O. Faruque, X. Nie, and V. Dinavahi, "A versatile cluster-based real-time digital simulator for power engineering research," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 455–465, May 2006.

[4] D. Paré, G. Turmel, J.-C. Soumagne, V. A. Do, S. Casoria, M. Bissonnette, B. Marcoux, and D. McNabb, "Validation tests of the hypersim digital real time simulator with a large AC-DC network," in *Proc. Int. Conf. Power System Transients*, New Orleans, LA, Sep. 2003, pp. 577–582.

[5] H. W. Dommel, "Digital computer solution of electromagnetic transients in single and multiphase networks," *IEEE Trans. Power App. Syst.*, vol. PAS-88, pp. 388–399, Apr. 1969.

[6] E. Lelarasmee, A. E. Ruehli, and A. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," *IEEE Trans. Computer-Aided Des. Integr. Circuits Syst.*, vol. 1, no. 3, pp. 131–145, Jul. 1982.

[7] M. Ilic-Spong, M. L. Crow, and M. A. Pai, "Transient stability simulation by waveform relaxation methods," *IEEE Trans. Power Syst.*, vol. 2, no. 4, pp. 943–949, Nov. 1987.

[8] M. L. Crow, "Waveform relaxation methods for the simulation of systems of differential/algebraic equations with application to electric power systems," Ph.D. dissertation, Univ. Illinois at Urbana-Champaign, Urbana, IL, 1990.

[9] M. L. Crow and M. Ilic, "The parallel implementation of the waveform relaxation method for transient stability simulations," *IEEE Trans. Power Syst.*, vol. 5, no. 3, pp. 922–932, Aug. 1990.

[10] L. Hou and A. Bose, "Implementation of the waveform relaxation algorithm on a shared memory computer for the transient stability problem," *IEEE Trans. Power Syst.*, vol. 12, no. 3, pp. 1053–1060, Aug. 1997.

[11] P. M. Anderson and A. A. Fouad, *Power System Control and Stability*. Ames, IA: Iowa State Univ. Press, 1977.

[12] "Definition and classification of power system stability," *IEEE Trans. Power Syst.*, vol. 19, no. 2, pp. 1387–1401, May 2004.

[13] J. K. White and A. L. Sangiovani-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*. Boston, MA: Kluwer, 1987.

[14] G. Kron, *Diakoptics—The Piecewise Solution of Large-Scale Systems*. London, U.K.: MacDonald, 1963.

[15] N. Rabbat, A. Sangiovanni-Vincentelli, and H. Y. Hsieh, "A multilevel newton algorithm with macromodeling and latency for the analysis of large-scale nonlinear circuits in the time domain," *IEEE Trans. Circuits Syst.*, vol. 26, no. 9, pp. 733–741, Sep. 1979.

[16] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic, 1970.

[17] A. R. Newton and A. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *IEEE Trans. Electron Devices*, vol. 30, pp. 1184–1207, Sep. 1983.

[18] M. L. Scala, M. Brucoli, F. Torelli, and M. Trovato, "A Gauss-Jacobi-block-Newton method for parallel transient stability analysis," *IEEE Trans. Power Syst.*, vol. 5, no. 2, pp. 1168–1177, May 1990.

[19] M. L. Scala, R. Sbrizzai, and F. Torelli, "A pipelined-in-time parallel algorithm for transient stability analysis," *IEEE Trans. Power Syst.*, vol. 6, no. 2, pp. 715–722, May 1991.

[20] J. Sun and H. Grotstollen, "Fast time-domain simulation by waveform relaxation methods," *IEEE Trans. Circuits Syst.*, vol. 44, no. 8, pp. 660–666, Aug. 1997.

[21] J. S. Chai and A. Bose, "Bottlenecks in parallel algorithms for power system stability analysis," *IEEE Trans. Power Syst.*, vol. 8, no. 3, pp. 9–15, Aug. 1993.

[22] H. You, V. Vittal, and X. Wang, "Slow coherency-based islanding," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 483–491, Feb. 2004.

[23] P. Kundur, *Power System Stability and Control*. New York: McGraw-Hill, 1994.

[24] X. Wang, V. Vittal, and G. T. Heydt, "Tracing generator coherency indices using the continuation method: A novel approach," *IEEE Trans. Power Syst.*, vol. 20, no. 3, pp. 1510–1518, Aug. 2005.

[25] Opal-RT Technologies, RT-LAB 8.0.2 [Online]. Available: http://www.opal-rt.com.

[26] *IEEE recommended practice for excitation system models for power system stability studies*, IEEE Std. 421.5-2005 2, IEEE Power Eng. Soc., Apr. 2006, pp. 1–85.

Vahid Jalili-Marandi (S'06) received the B.Sc. and M.Sc. degrees in power systems engineering from the University of Tehran, Tehran, Iran, in 2003 and 2005, respectively. Currently, he is pursuing the Ph.D. degree at the University of Alberta, Edmonton, AB, Canada.

His research interests include transient stability studies and digital real-time simulations in power systems.

Venkata Dinavahi (M'00–SM'08) received the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2000.

Presently, he is an Associate Professor at the University of Alberta, Edmonton, AB, Canada. His research interests include electromagnetic transient analysis, power electronics, and real-time simulation and control.