AN IMPLEMENTABLE DISTRIBUTED STATE ESTIMATOR AND DISTRIBUTED BAD DATA PROCESSING SCHEMES FOR ELECTRIC POWER SYSTEMS

S.-Y. Lin C.-H. Lin

Department of Control Engineering
National Chiao Tung University
Hsinchu, TAIWAN, ROC

Abstract - In this paper, we present an implementable distributed state estimator and distributed bad data detection and identification schemes for electric power systems. The system under consideration is partitioned into k areas, each of which is governed by a local control center that has at least one computer system. These decentralized computer systems are linked by a communication network. The distributed state estimator and distributed bad data processing schemes are designed to be carried out in this computer network. The proposed distributed state estimator possesses the following properties, which ensure its implementability: 1) it uses only the computers available in the decentralized local control centers, 2) it is executed in a small-scale computer communication network, 3) it can converge even if errors occasionally occur in the communication links, and 4) the decentralized computers need not be synchronous. Furthermore, our distributed bad data processing schemes can process the bad data in each area independently using the corresponding local computer system.

I. Introduction

To render the burden of massive data processing and computations in the control center of large power system and to improve the efficiency of control and management, distributed processing is a trend. In recent years, distributed processing systems have become more practical to implement because advances in computer network technology and the drastic reduction in the cost of computers has made sophisticated computer communications environments and computers available at lower cost. However, before advanced computer network technology can be employed in power system control and management, methods that are suitable for distributed processing need to be developed. In this paper, we focus on state estimation, a fundamental problem in the control and management of power systems, and present an implementable distributed state estimator and distributed bad data detection and identification schemes. The concept of distributed state estimation was proposed by Lin in [1]. In that work, a distributed state estimator based on the RQPD method was considered. However, that distributed state estimator is hard to implement from a practical viewpoint, because 1) a data communication network that is topologically the same and physically in parallel with the power network is needed, 2) the data communication network should have no communication error, 3) too many processors are needed, since each node in the data communication network needs a processor, e.g., the number of processors equals the number of buses, and 4) a local synchronization scheme in each processor is needed to govern the synchronization of the iterative RQPD method. To circumvent those four restrictions on the data communication network, this paper uses a different approach. First of all, we assume that the power system is partitioned into several areas and that each area has a local control center to govern that area's operations. This assumption is reasonable because for large system decentralized control seems to be a promising approach. We further assume that the computer systems in the local control centers of adjacent areas are connected by communication links and thus form a small-scale computer network. Then, we present in this paper a distributed state estimator based on the partially asynchronous block Jacobi method, which will operate on the small-scale computer network. In contrast to the four drawbacks of the distributed state estimator based on the RQPD method, the distributed state estimator proposed here possesses the following properties, which ensure its implementability: 1) it uses only the computers available in the decentralized local control centers, 2) it is executed in a small-scale computer communication network, 3) it can converge even if errors occasionally occur in the communication links, and 4) the decentralized computers need not be synchronous. Furthermore, we will also present distributed bad data detection and identification schemes for use with the distributed state estimator; these schemes are based on a reduced model method proposed by Korres and Contaxis [2].

93 SM 531-4 PWRS A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1993 Summer Meeting, Vancouver, B.C., Canada, July 18-22, 1993. Manuscript submitted Jan. 4, 1993; made available for printing May 26, 1993.

PRINTED IN USA

II. Background of the Computer Network and Measurement Requirements

The power system under consideration is partitioned into k areas, and each area is governed by a local control center. The measurement data in each area will be collected in each individual local control center. Two areas are called *adjacent areas* if they are connected to each other by tie-lines. Fig. 1 shows such a structure. We assume that each local control center has at least one computer system for data acquisition, data processing, and computation. The computer systems of adjacent areas are connected by communication links, so these decentralized computer systems form a computer network. Fig. 2 shows a computer network corresponding to the system shown in Fig. 1, where each black square denotes a computer system.

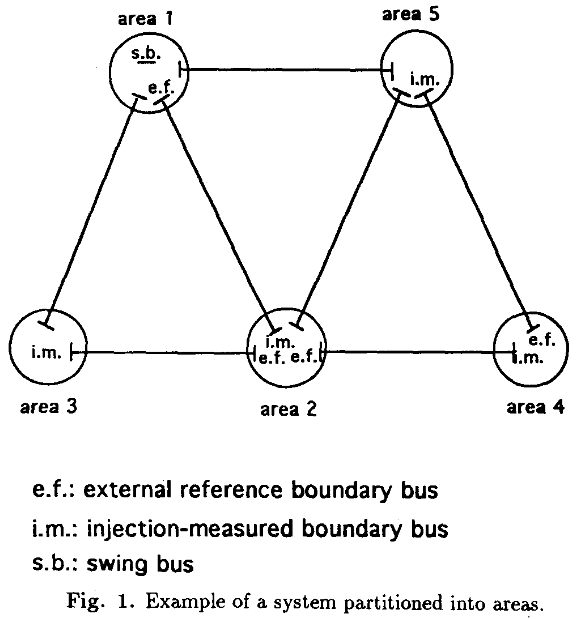


Fig. 1. Example of a system partitioned into areas.

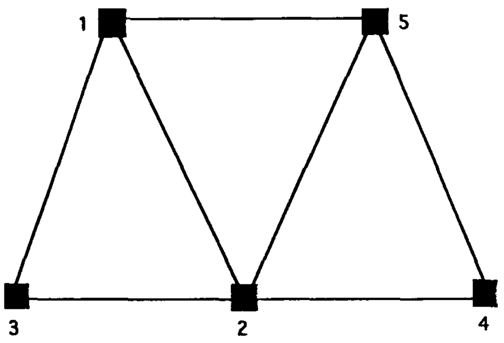


Fig. 2. Example of a computer network for the system shown in Fig. 1.

Let the boundary bus with real and reactive power injection measurements of an area be called an *injection measured boundary bus.* Let the tie-lines with power flow measurements be called *flow-measured tie-lines.* Then, a boundary bus of area i connected to some adjacent area, say area j, at an injection-measured boundary bus of area j or at a boundary bus of area j through a flow-measured tie-line is called an *external reference boundary bus* of area j. For example, in Fig. 1, the external reference boundary bus in area 1 connected to an injection-measured boundary bus in area 2 is an external reference boundary bus of area 2.

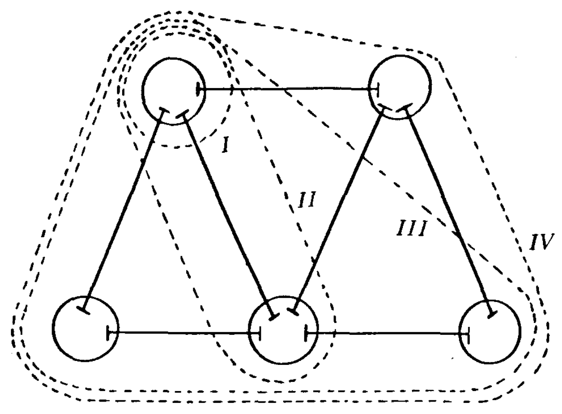
Then, for the purpose of distributed bad data detection and identification, the placement of the measurement meters in the system should meet the following requirements.

1. Each of the k areas is a maximal noncritical area as defined in [2] and is also an observable subnetwork as defined in [3].
2. Each area must contain at least one external reference boundary bus.

A property that follows directly from requirement 1 is that the injection-measured boundary buses of an area or the flow-measured tie lines must be critical measurements. We call these critical measurements *boundary critical measurements.* More details on this type of measurement setup will be presented in the section below on bad data detection and identification. Here we will simply note that the measurement setup of the test examples presented in [2] meets the two requirements above.

To account for the effect of the swing bus in the distributed state estimation, we use a *distributed tier propagation method,* which will be presented later, based on a *tier structure.* The tier-structure for the areas of the power system (or the computer network) is formed by first designating the area containing the swing bus to be area 1, and naming that area tier 1. Thus, tier 1 contains only one area. We then set $j = 1$ and label as the areas in tier $j+1$ all the as yet unlabelled areas that have external reference boundary buses in the areas of tier j. We repeat this labelling process until all areas are labelled. Let T denote the total number of tiers formed, and let j_n denote the total number of areas in tier j. Fig. 3 shows an example of the tier-structure for the power system shown in Fig. 1.

In the computer network, we need to assign one computer system to initiate the process of state estimation and to collect and broadcast simple status and command signals from and to all other computer systems. Accordingly, *communication paths* are needed, and we will also designate these communication paths based on the tier structure described above. Viewing the decentralized computer systems as *nodes* in the computer network, we designate the node corresponding to area 1 to be the *root node.* Then, based on the tier structure, we may form an outgoing directed radial tree as follows. All the nodes in tier 2 are directly connected to the root node by a single communication link. Thus, all these communication links will be set up as tree branches with an outward direction. We then set $j = 2$, let the set of nodes in tier $j+1$ be denoted by $\mathcal{N}^{j+1}$, and let the set $\bar{\mathcal{N}}^{j+1} = \mathcal{N}^{j+1}$. Starting from the first node of tier j, we assign the communication links that connect the first node of tier j to



I: tier 1, II: tier 2, III: tier 3, IV: tier 4.

Fig. 3. Example of the tier-structure for the system shown in Fig. 1.

nodes in $\bar{\mathcal{N}}^{j+1}$ as tree branches with an outward direction and update $\bar{\mathcal{N}}^{j+1}$ by deleting the nodes connected to the first node of tier j. We repeat this process from the second node until the j_nth node of tier j and from the second tier to $(T-1)$th tier, so that an outgoing directed radial tree is eventually formed. Fig. 4 shows such an outgoing directed radial tree for the computer network shown in Fig. 2. Thus, we let the root node be responsible for initiating the state estimation and collecting and broadcasting all the necessary signals. All other nodes will communicate with the root node through a backward direction in the radial tree. Furthermore, we define a node i in the radial tree as a predecessor of node j if there exists a directed tree branch from node i to node j. Consequently, node j will be a follower of node i if node i is a predecessor of node j. For example, in Fig. 5, node 2 is the predecessor of nodes 3 and 4, and nodes 3 and 4 are followers of node 2. Any node that has no followers is called a leaf node.

III. Statement of the Distributed State Estimation Problem

For the ith area ($i = 1, 2, \cdots, k$), let us adopt the following notation:

N_i: the number of buses

$\mathbf{z}_i$: the local measurement vector, which may include the tie-line flow measurements measured from the end bus in the ith area

$\mathbf{x}_i$: the local state vector, composed of N_i voltage magnitudes and N_i phase angles

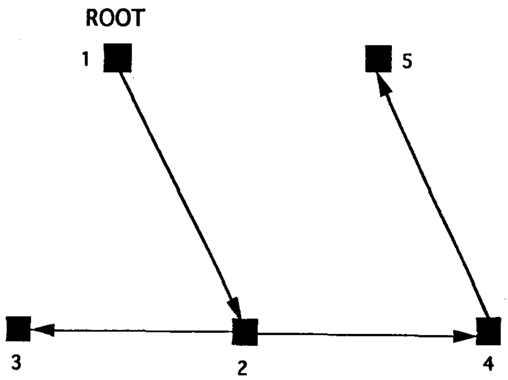


Fig. 4. Example of the directed radial tree for the computer network shown in Fig. 2.

$\mathbf{x}_{i,e}$: the state vector of the external reference boundary buses of area i

$\mathbf{h}_i$: the nonlinear measurement vector function corresponding to $\mathbf{z}_i$

η_i: the Gaussian random measurement error vector corresponding to $\mathbf{z}_i$

R_i: the diagonal covariance matrix of the measurement error vector η_i

The measurements in each area i can be expressed as

$$\mathbf{z}_i = \mathbf{h}(\mathbf{x}_i, \mathbf{x}_{i,e}) + \eta_i. \tag{1}$$

Let the state vector, measurement vector, nonlinear measurement vector function, the measurement error vector and the corresponding covariance matrix of the whole system be denoted by $\mathbf{x}$, $\mathbf{z}$, $\mathbf{h}$, η, and R, respectively. Then $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k), \mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_k), \mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_k), \eta = (\eta_1, \eta_2, \cdots, \eta_k)$ and $R = \begin{bmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & R_k \end{bmatrix}$. Furthermore, $N(= N_1 + N_2 + \cdots + N_k)$ denotes the total number of buses of the system. Based on (1), the measurements of the whole system can be expressed by

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \eta. \tag{2}$$

Then, the problem of our distributed state estimation is to use the computer network associated with the measurement data collected in each local control center to solve the following weighted least square (WLS) problem in a distributed way:

$$\min_{\mathbf{x}} J(\mathbf{x})(= \frac{1}{2}[\mathbf{z} - \mathbf{h}(\mathbf{x})]^T R^{-1}[\mathbf{z} - \mathbf{h}(\mathbf{x})]), \tag{3}$$

where the phase angle of the swing bus will be kept fixed as a reference.

IV. The Distributed State Estimator Based on the Partially Asynchronous Block Jacobi Method

From the definitions of $\mathbf{z}$, $\mathbf{x}$, $\mathbf{h}$, and η and the form of (1), the measurement equation (2) can be rewritten as

$$\begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_k \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1(\mathbf{x}_1, \mathbf{x}_{1,e}) \\ \vdots \\ \mathbf{h}_k(\mathbf{x}_k, \mathbf{x}_{k,e}) \end{bmatrix} + \begin{bmatrix} \eta_1 \\ \vdots \\ \eta_k \end{bmatrix} \tag{4}$$

and the WLS problem can be rewritten as

$$\min J(\mathbf{x})(= \sum_{i=1}^{k}[\mathbf{z}_i - \mathbf{h}_i(\mathbf{x}_i, \mathbf{x}_{i,e})]^T R_i^{-1}[\mathbf{z}_i - \mathbf{h}_i(\mathbf{x}_i, \mathbf{x}_{i,e})]). \tag{5}$$

Obviously, a distributed computation performed in the computer network environment should be different from a computation carried out in a central computer. First of all, in the centralized computation, the reference effect of the phase angle of the swing bus will affect the states of all other buses during the progress of the computing process. However, in the distributed computation, since the swing bus lies only in area 1, then initially, other areas cannot utilize the effect of the reference phase angle of the swing bus without certain initial value processing procedures. To overcome this drawback of a distributed computation environment, we use a *distributed tier propagation method* to deliver the reference effect of the reference phase angle of the swing bus to all other areas $i = 2, \cdots, k$.

4.1 The initial value processing procedures

The distributed tier propagation method, which contains the distributed forward calculation and backward notification procedures, is based on the tier structure as described below. Without loss of generality, we will order the sequence of the indices of the areas according to the tier structure as $1_1(= 1), 2_1, \cdots, 2_n, \cdots, T_1, \cdots, T_n(= k)$. Note that area 1_1 is the only area in tier 1 that contains the swing bus.

Distributed forward tier propagation for calculating initial values of the states: The procedures will be described in two steps.

(1) Once the root node initiates the process of state estimation, the computer system of area 1 will perform the following. Let $\bar{\mathbf{z}}_{1_1}$ denote $\mathbf{z}_{1_1}$ but exclude the boundary critical measurements, and let $\bar{\mathbf{h}}_{1_1}$ and $\bar{\eta}_{1_1}$ denote the nonlinear measurement vector function and measurement error vector corresponding to $\bar{\mathbf{z}}_{1_1}$, respectively. Since each area is an observable subnetwork by requirement 1 of the measurement setup, we will solve the following isolated area state estimation problem:

$$\min_{\mathbf{x}_{1_1}}[\bar{\mathbf{z}}_{1_1} - \bar{\mathbf{h}}_{1_1}(\mathbf{x}_{1_1})]^T \bar{R}_{1_1}^{-1}[\bar{\mathbf{z}}_{1_1} - \bar{\mathbf{h}}_{1_1}(\mathbf{x}_{1_1})]$$

in the computer system of area 1_1, where $\bar{R}_{1_1}$ is the covariance matrix of $\bar{\eta}_{1_1}$. Then, the computer system of area 1 will send the values of the states of the external reference boundary buses to the adjacent areas in tier 2.

(2) Once the computer system in area i of tier $j(> 1)$ receives the values of the state vector $\mathbf{x}_{j_i,e}^{j-1}$ of all the external reference boundary buses from the adjacent areas in tier $j-1$, it will perform the following. Let $\bar{\mathbf{z}}_{j_i}$ denote $\mathbf{z}_{j_i}$ but exclude all the boundary critical measurements except for those which are involved with the received states of the external reference boundary buses in the areas of tier $j-1$. If these included boundary critical measurements are involved also with the external reference boundary buses in the areas of the same tier j, we denote the states of the external reference boundary buses as $\mathbf{x}_{j_i,e}^{j}$. Let $\bar{\mathbf{h}}_{j_i}$ and $\bar{\eta}_{j_i}$ denote the nonlinear measurement vector function and measurement error vector corresponding to $\bar{\mathbf{z}}_{j_i}$, respectively. We will solve the following isolated area state estimation problem:

$$\min_{\mathbf{x}_{j_i}}[\bar{\mathbf{z}}_{j_i} - \bar{\mathbf{h}}_{j_i}(\mathbf{x}_{j_i}, \bar{\mathbf{x}}_{j_i,e})]^T \bar{R}_{j_i}^{-1}[\bar{\mathbf{z}}_{j_i} - \bar{\mathbf{h}}_{j_i}(\mathbf{x}_{j_i}, \bar{\mathbf{x}}_{j_i,e})],$$

where $\bar{R}_{j_i}$ is the covariance matrix of $\bar{\eta}_{j_i}$, $\bar{\mathbf{x}}_{j_i,e}$ contains $\mathbf{x}_{j_i,e}^{j-1}$, and $\mathbf{x}_{j_i,e}^{j}$, and the values of $\mathbf{x}_{j_i,e}^{j}$ are set as constant with 1 p.u. voltage magnitude and 0° phase angle in the above minimization problem.

Backward propagation for notifying root node of the completion of initial value calculation:

Let a flag $\hat{s}_i$ denote the completion of the initial value calculation of the ith area. $\hat{s}_i = 1(0)$ represents completion (incompletion). The following describes an accumulated way to inform the root node the completion of the initial value processing. Once any ith individual computer system in tier T completes the calculation of initial values, it will send a flag $\hat{s}_{T_i} = 1$ to its predecessor in the radial tree. For any node j in the radial tree, let $j_1, \cdots, j_f$ denote all the followers of node j. If node j receives the flag $\hat{s}_i = 1$, $i = j_1, \cdots, j_f$ from all its followers, then it will send a flag $\hat{s}_j = 1$ to its predecessor.

Stopping criteria for initial value processing: Let $r_1, \cdots, r_f$ denote the indices of the followers of the root node r. If the root node receives the flag $\hat{s}_i = 1$, $i = r_1, \cdots, r_f$ from all its followers, then the root node will initiate the partially asynchronous block Jacobi method based distributed state estimation process.

4.2 The distributed state estimation based on the partially asynchronous block Jacobi method.

First of all, we shall describe the block Jacobi method for the WLS problem.

4.2.1 The block Jacobi method. Let $C_i(\mathbf{x}_i) = H_i(\mathbf{x}_i)^T R_i^{-1} H_i(\mathbf{x}_i)$, where $H_i(\mathbf{x}_i) = \frac{\partial \mathbf{h}_i(\mathbf{x}_i, \mathbf{x}_{i,e})}{\partial \mathbf{x}_i}$. Then the block Jacobi method for solving (5) uses the following iterations:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \gamma \Delta \mathbf{x}(t) \tag{6}$$

where γ is a positive stepsize, t denotes the iteration index, and $\Delta \mathbf{x}(t)$ is the solution of

$$\hat{C}(\mathbf{x}(t))\Delta \mathbf{x}(t) = -\nabla J(\mathbf{x}(t)), \tag{7}$$

where

$$\hat{C}(\mathbf{x}(t)) = \begin{bmatrix} C_1(\mathbf{x}_1(t)) & 0 & \cdots & 0 \\ 0 & C_2(\mathbf{x}_2(t)) & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & C_k(\mathbf{x}_k(t)) \end{bmatrix},$$

$\nabla J(\mathbf{x}) = [\mathbf{z} - \mathbf{h}(\mathbf{x})]^T R^{-1} H(\mathbf{x})$, and $H(\mathbf{x}) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}$. If we consider the whole system as an area, then the above block Jacobi method is the Newton-like method. Since $\Delta\mathbf{x} = (\Delta\mathbf{x}_1, \Delta\mathbf{x}_2, \cdots, \Delta\mathbf{x}_k)$, $\nabla J(\mathbf{x}) = (\nabla J_1(\mathbf{x}_1), \cdots, \nabla J_k(\mathbf{x}_k))$, where $\nabla J_i(\mathbf{x}_i) = (\mathbf{z}_i - \mathbf{h}_i\,(\mathbf{x}_i, \mathbf{x}_{i,e}))^T\ R_i^{-1} H_i(\mathbf{x}_i)$, then from the structure of $\hat{C}(\mathbf{x}(t))$, we see that (7) can be decomposed into k independent sets of linear equations as shown below:

$$C_i(\mathbf{x}_i(t))\Delta\mathbf{x}_i(t) = -\nabla J_i(\mathbf{x}_i(t)), \quad i = 1, 2, \cdots, k. \tag{8}$$

Note that $C_i(\mathbf{x}_i(t))$ and $\nabla J_i(\mathbf{x}_i(t))$ are functions of $\mathbf{x}_i(t)$ and $\mathbf{x}_{i,e}(t)$. Therefore, if each local computer system receives the updated values of $\mathbf{x}_{i,e}(t)$ through the communication links from the adjacent computer systems, it can perform (8) and update

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \gamma\Delta\mathbf{x}_i(t). \tag{9}$$

4.2.2 The partially asynchronous algorithmic model for distributed state estimation. The execution of (8) and (9) for all $i = 1, \cdots, k$ in a centralized computer is carried out in a synchronous mode. However, in a distributed computing environment, it is difficult to execute an iterative method using decentralized computer systems and still maintain synchronization. Thus, if it is to be implemented in a distributed computing environment, it is almost unavoidable that the iterative method will be carried out somewhat asynchronously. However, in this case, a totally asynchronous mode is not required by contemporary sophisticated computer network technologies; rather, a partially asynchronous mode more adequately reflects the communication capability of computer networks. To precisely describe the asynchronism between the decentralized computer systems, we should consider the iteration index t in the previous subsection as a true time. Thus, $t+1$ represents one time unit ahead of t but not the next iteration count as shown in (9). Note that the time unit considered here is not a second or a minute; its value depends on the processing speed of computer, the communication rate of the communication links, and the size (number of buses) of the local area. The timing for the ith computer system to carry out the calculation of (8) can be asynchronous with other computer systems. Moreover, different frequencies of computing (8) can occur in different computer systems due to the different size of areas. Therefore, let T^i denote the set of times at which the local state vector is updated. Then, (9) under the partially asynchronous model becomes $\mathbf{x}_i(t+1) = \mathbf{x}_i(t)$ if $t \notin T^i$, and $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \gamma\Delta\mathbf{x}_i(t)$ if $t \in T^i$. Moreover, because the value of $\mathbf{x}_{i,e}$ stored in the memory of the ith computer system was sent from adjacent computer systems, $\mathbf{x}_{i,e}$ used in the ith computer system for solving (8) may not be the current $\mathbf{x}_{i,e}$ due to a delay in or loss of communication between computer systems. Let $\mathbf{x}_{i,e_j}(\tau_j^i(t))$ denote the value of $\mathbf{x}_{i,e_j}$, the state of the external reference boundary bus e_j, at time $\tau_{e_j}^i(t)$. Then $\mathbf{x}_{i,e_j}(\tau_{e_j}^i(t))$ denotes the value of $\mathbf{x}_{i,e_j}$ stored in the ith computer used in solving (8), where $\tau_{e_j}^i(t) \leq t$. Then, the partially asynchronous algorithmic model is based on an asynchronous measure B, which is a positive time unit, and makes the following partially asynchronous assumptions:

(a) For every ith computer system and every $t \geq 0$, at least one of the elements of the set $\{t, t+1, \cdots, t+B-1\}$ belongs to T^i.

(b) For all i and e_j, and all $t \geq 0$ belonging to T^i,

$$t - B < \tau_{e_j}^i(t) \leq t.$$

These assumptions imply that (a) each computer system performs an update at least once during any time interval of length B, and (b) the external information used by any computer is outdated by at most B time units.

4.2.3 The partially asynchronous distributed state estimator. It may happen that the ith computer system receives the updated $\mathbf{x}_{i,e}$ while solving (8). For such a case, we will not change the value of $\mathbf{x}_{i,e}$ during computation. However, to avoid data confusion, we will use two sets of memory locations to store the value of $\mathbf{x}_{i,e}$. One set of memory locations is called the *updating memory*, which stores the updated value of $\mathbf{x}_{i,e}$ received from adjacent computer systems. The other set of memory locations is called the *executing memory*, which will fetch the value of $\mathbf{x}_{i,e}$ from the updating memory when the computer starts solving (8). The stored value in the executing memory is kept unchanged before the computer finishes solving (8). Then, based on the above partially asynchronous assumptions, we see that each computer system can perform its designated computation without considering the situation of the other computer systems. Thus, the partially asynchronous block Jacobi method based distributed state estimator can be stated as follows:

Each local computer system continue performing the following three steps after the initial values of $\mathbf{x}_i$ are calculated.

Step 1: Solve (8) to obtain $\Delta\mathbf{x}_i(t)$ based on the current value of $\mathbf{x}_i(t)$ and the value of $\mathbf{x}_{i,e}$ in the executing memory, which consists of $\mathbf{x}_{i,e_j}(\tau_{i,e_j}^i(t))$ for all external reference boundary buses e_j.

Step 2: Once Step 1 is completed, update $\mathbf{x}_i(t)$ by $\mathbf{x}_i(t) + \gamma\Delta\mathbf{x}_i(t)$ and send the values of the states of the external reference boundary buses of the adjacent areas to the corresponding computer systems through the communication links.

Step 3: If $\mid \Delta\mathbf{x}_i(t) \mid_\infty$ (the largest magnitude of the components in $\Delta\mathbf{x}_i(t)) < \varepsilon$ (a preselected accuracy), set a flag $\nu_i = 1$. If node i is a leaf node, set $\hat{\nu}_i = \nu_i = 1$. If node i is not a leaf node, determine $\hat{\nu}_i = \nu_i \wedge \hat{\nu}_{i_1} \wedge \cdots \wedge \hat{\nu}_{i_f}$ where $i_1, i_2, \cdots, i_f$ are the indices of the followers of node i. Then, if the value $\hat{\nu}_i = 1$, this value will be sent to the predecessor of node i.

Stopping criteria: If $\hat{\nu}_r = 1$, where the subscript r denotes the root node, then the distributed state estimation based on the partially asynchronous block Jacobi method is completed. Consequently, the root node will broadcast to all other nodes a command to begin the process of distributed bad data detection and identification along the directed radial tree.

4.2.4 Discussion concerning the convergence of the partially asynchronous block Jacobi method based distributed state estimation and the stepsize γ. The general version of the partially asynchronous block Jacobi method has been applied to solve numerical problems using parallel processors. A theorem of convergence for the general version of this method has been shown in [4]. A version for our case is stated in the Appendix.

The main concern of this theorem is to determine a range $(0, \gamma_0)$ and show that if the updating stepsize $\gamma \in (0, \gamma_0)$, then the partially asynchronous block Jacobi method based distributed state estimator will converge. As can be seen from the formula for determining the value of γ_0 given in the Remarks concerning the theorem in the Appendix, the value of γ_0 is inversely proportional to the asynchronous measure B and the product of B and the size of the system, $N'(= 2N - 1)$. This conclusion is reasonable because (1) if the degree of asynchronism is larger, i.e., a larger B, the convergence is more difficult, thus the stepsize should be smaller; (2) if the asynchronism is present, the convergence is more difficult for larger system, hence the stepsize should be smaller. However, the actual value of B can be obtained only from testing the actual computer network and related power system, and this value may vary for the different computer networks. This implies that the value γ_0 will vary for different computer networks and related power systems due to different values of B and different size of the power systems. Therefore, we may not have a typical value for γ_0. However, γ can be any value between 0 and γ_0. Larger γ makes the distributed state estimation converge fast but unsafe if the value γ_0 is overestimated. Smaller γ makes the distributed state estimation converge slow but safe. Thus, the stepsize γ is best determined based on numerous simulations on the actual computer network and related power system. Anyhow, the theorem of convergence ensures the existence of the stepsize γ.

4.2.5 Some features of the partially asynchronous block Jacobi method based distributed state estimation. Based on the partially asynchronous assumptions and the above convergence theorem, this partially asynchronous block Jacobi method based distributed state estimator clearly has the following features: (i) each local computer system performs its computations independent of the other local computer systems, (ii) even if communication errors occur in the communication links between adjacent computer systems, as long as such errors happen only occasionally, they will not affect the convergence of the method, (iii) various communication delays in different communication links and different frequencies of computation in different computer systems are tolerable, and (iv) during the solution process, if topological changes occur in certain area and the local control center of this area is informed of these changes and update the data base; then without stopping the ongoing solution process of all areas, the final convergent solution is in accordance with the most updated topology.

The above four features make the proposed distributed state estimator implementable.

Remark 4.1: About the fourth feature, when the computer of an area correct its data base on account of the topological changes, the values of the states in the ongoing solution process can be viewed as initial values of the states for the system with updated topology. Thus, there is no need to inform other areas about this topological change. Furthermore, it can be observed that our block Jacobi method based distributed state estimator only need to know the boundary connections to adjacent areas but not the complete topology of the other areas.

V. Bad Data Detection and Identification

In some cases, the measurement data may be contaminated, thereby making the estimated state invalid. Therefore, to detect and identify all the possible bad data, eliminate them and then re-estimate the states from a necessary procedure to obtain a reliable set of estimated states. There are several good methods available [6]-[9] for implementing bad data detection and identification schemes in a central computer. However, for a distributed computing environment, we should use distributed bad data detection and identification schemes. The method we employ here is based on the reduced model method proposed by Korres and Contaxis [2].

5.1 Preliminaries

A measurement is called *critical* if its suppression from the measurement set makes the system unobservable. The connected portion of the subnetwork consisting of branches incident to *non-critical* measurements is called the *maximal non-critical area* [2]. Klements, et al., have developed in [3] a topological algorithm to determine the *region of measurement error residual spread.* Korres and Contaxis have shown in [2] that the *error residual spread area* equals the *maximal non-critical area.* Thus, the measurement errors on non-critical measurements of different maximal non-critical areas do not contaminate each other. This implies that bad data processing for disjoint maximal non-critical areas can be carried out independently. Furthermore, Krumpholz, et al., have shown in [5] that the criticality of measurements depends essentially on the type and location of the measurements. Therefore, if we place the measurement meters properly, so that each of the k areas is a maximal non-critical area, then each local computer system can process the bad data of the corresponding area independently. From the topological algorithm proposed by Klements, et al., in [5], we see that if we set up the measurements so that (i) each area is an observable subnetwork consisting of at least one redundant measurement in the topological sense [5], (ii) there are no flow measurements on all tie-lines, (iii) each area has at least one injection-measured boundary bus and at least one external reference boundary bus, and (iv) for any injection-measured boundary bus, there are no power injection measurements on the corresponding external reference boundary buses. Then, each area will essentially be a maximal non-critical area. Furthermore, the above setup meets the two requirements posed in Section II. However, it is important to double check the above measurement setup to ensure that the injection measurements on injection-measured boundary buses are really numerically critical for a case. Then, under this measurement setup, we can perform distributed bad data processing in the computer network.

5.2. Bad data detection and identification of each area

In each maximal non-critical area, let $\hat{\mathbf{x}}_i$ be the estimated states for area i determined previously. Korres and Contaxis showed in [2] that the performance index of area i, $J_i(\hat{\mathbf{x}}_i)$, has a chi-square distribution with d_i degrees of freedom, where $d_i = m_i - N_i + \dim\mathcal{N}(H_i), m_i$ is the number of measurements of area i, and $\mathcal{N}(H_i)$ is the null space of H_i. We will perform bad data detection based on a statistical hypothesis by testing the value of $J_i(\hat{\mathbf{x}}_i)$ and the value of the normalized residue, that is,

$$J_i(\hat{\mathbf{x}}_i) < \alpha_i, \tag{10}$$

$$| r^N_{i,j} | < \beta_{i,j}, \tag{11}$$

where $r^N_{i,j}$ is the normalized residue of the jth bus in area i, α_i is the objective function detection threshold for area i, and $\beta_{i,j}$ is the normalized residue detection threshold for the jth bus in area i. Then, bad data are assumed to be present if any one of the above two tests fails. Once the existence of bad data is detected in an area, we will employ an existing bad data identification method, such as the method based on measurement compensation and linear residual calculation presented in [9], to identify either single or multiple bad data in that area.

5.3. The distributed bad data detection and identification schemes

The distributed bad data detection and identification schemes discussed in the preceding subsections can be stated as follows:

Each local computer system will perform the following operations once it receives from the root node the *command to detect* bad data.

Step 1: Calculate the value $J_i(\hat{\mathbf{x}}_i)$, the degrees of freedom d_i, and the normalized residue $r^N_{i,j}$ for all buses in area i. Then test the value of $J_i(\hat{\mathbf{x}}_i)$ and $| r^N_{i,j} |$ based on (10) and (11) with a properly selected α_i and $\beta_{i,j}$ to determine whether bad data are present.

Step 2: If bad data are detected in Step 1, then use the method based on measurement compensation and linear residual calculation to identify the bad data and eliminate them.

Step 3: Set flag $b_i = 1$ if area i has bad data; otherwise, set $b_i = 0$. Then, if node i is a leaf node, set $\hat{b}_i = b_i$; otherwise, perform $\hat{b}_i = b_i \wedge \hat{b}_{i_1} \wedge \cdots \wedge \hat{b}_{i_f}$ when the values of the $\hat{b}_{i_j}$'s from all the followers of node i are received, where $i_1, i_2, \cdots, i_f$ are the indices of the followers of node i. Once the value of $\hat{b}_i$ is determined, it will be sent to the predecessor.

Criteria for terminating distributed state estimation or restarting another cycle of the distributed state estimation: If the logical value $\hat{b}_r$ corresponding to the root node is $\hat{b}_r = 0$, then the root node will broadcast a signal to all other nodes to inform them that the state estimation is complete; otherwise, the root node will broadcast a signal to all other nodes to inform them that the whole process of distributed state estimation needs to be repeated using the new sets of measurement data.

VI. Test Results

The communication and broadcast of command and status signals between computer systems requires computer protocols. From the viewpoint of computer network technologies, the design of such simple protocols is not difficult; however, it is beyond the scope of this paper. Therefore, we will not use an actual computer network to simulate the proposed distributed state estimator. For this reason, in this section, we will refer to the partially asynchronous block Jacobi method as the distributed block Jacobi method most of the time. We simulated the numerical behavior induced by partial asynchronism and communication errors between decentralized computer systems in a sequential computer. We tested the distributed state estimator on the IEEE 30-bus and the IEEE 118-bus systems using a Sun 4/60 workstation. Due to the limitation on the length of this paper, we will present four test cases conducted on the larger system, the IEEE 118-bus system. Before presenting each individual case, we will describe the common setup of the system.

Area Partition: The system is partitioned into eight areas, as shown in Fig. 5, where each area is enclosed by a dashed contour. For easier addressing, we denote the eight areas by A1-A8, as marked beside the contours.

Measurement Setup: The measurement setup for all the test experiments is the flow measurements of all transmission lines

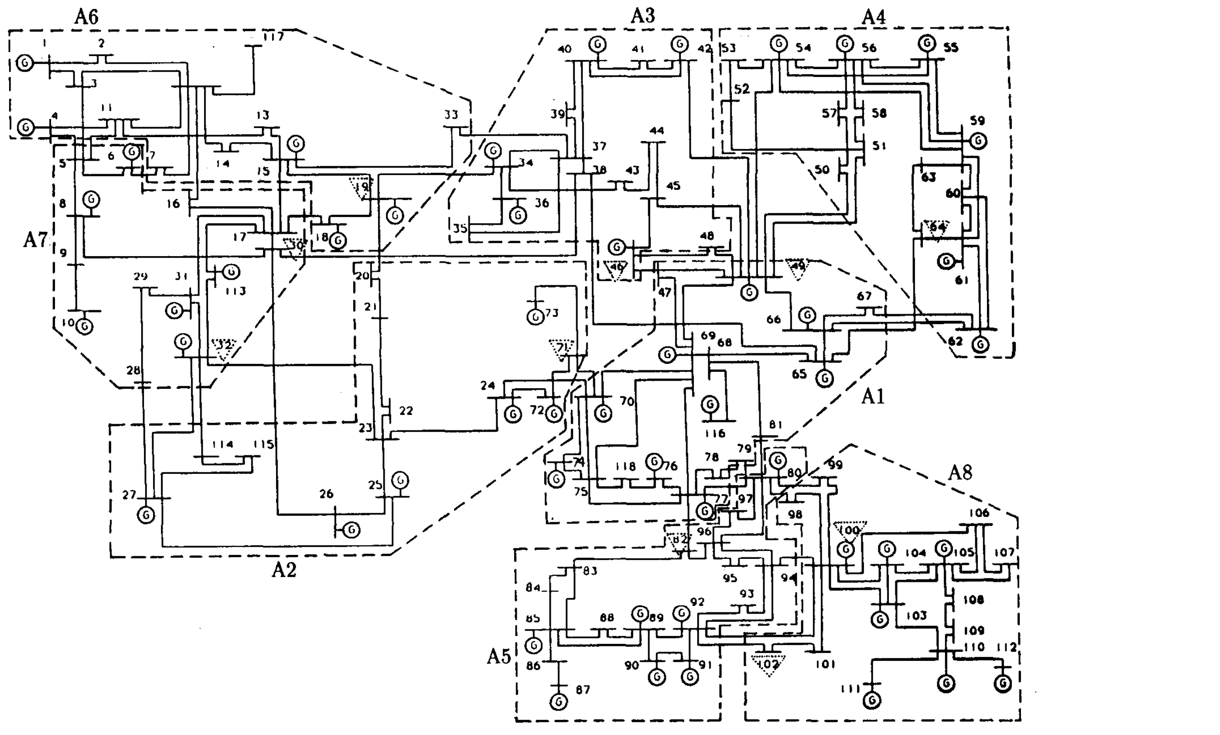


Fig. 5. The IEEE 118-bus system and its partioned 8 areas.

except for all the inter-area tie-lines, the real and reactive power injection measurements on all the non-boundary buses of each area, and the real and reactive power injection measurements on injection-measured boundary buses 19, 30, 32, 46, 49, 64, 71, 82, 100, and 102. For the sake of clarity in reading Fig. 5, we have marked the above 10 injection-measured boundary buses by dotted triangles. Thus, each area is an observable subnetwork. Furthermore, we have confirmed by testing that the above 10 real and reactive power injection measurements on the injection-measured boundary buses are numerically critical for a normal case. Hence, each area is a maximal non-critical area.

A Comment on Measurement Setup: As long as each area is a maximal non-critical area, the proposed distributed bad data detection and identification schemes presented in Section V always work. Therefore, we may also consider to include critical measurements of inter-area tie-line flows in our measurement setup; however, this will take considerable effort to explain why each area is a maximal non-critical area in the topological sense and cause confusion.

External Reference Boundary Buses: The external reference boundary buses can be observed directly from the locations of the injection-measured boundary buses in Fig. 5. Thus, the external reference boundary buses of each area are as follows: A1: {48, 54, 50, 51}, A2: {70}, A3: {47}, A4: {65}, A5: {77}, A6: {20, 34}, A7: {23, 27, 114, 26, 38}, and A8: {92, 94}.

Tier Structure: Since swing bus 69 is contained in area A1, A1 forms tier 1. According to the locations of the reference boundary buses, areas A2, A3, A4, and A5 form tier 2, and areas A6, A7, and A8 form tier 3.

Initial Guesses in the Initial Value Processing: The initial guesses of the values of all the states in the initial value processing procedures are 1.00 p.u. in voltage magnitude and 0 radians in phase angle.

The Stepsize: we set $\gamma = 1$ in all cases.

Test Case 1 and Results: In this case, we assumed that there are no bad data, and the measured values were obtained from a load flow solution. The sequence for solving the isolated area state estimation problem in the initial value processing procedures is based on the tier structure; the sequence started from area A1, followed by areas A2, A3, A4, and A5, and finally areas A6, A7, and A8. After the initial value processing procedures, the distributed block Jacobi method converged in one iteration for all areas, and the largest deviation of the voltage magnitude (p.u.) and the phase angle (radians) from the standard solution was less than 10^{-4}. Note that the effectiveness of the distributed block Jacobi method credits to the initial value processing procedures, the values of the states obtained from which are already close to the solution. The CPU time reported for our method on the Sun 4/60 workstation was 1.04 seconds. We also tested this case on the same workstation using the Newton-like method with a sparse matrix technique with initial guesses of states as $1.0\ \angle 0^0$; this method took 2.03 seconds of CPU time, and the resulting largest deviation of the voltage magnitude (p.u.) and phase angle (radians) from the standard solution was also on the order of 10^{-4}. This result shows that our method is not only good for distributed state estimation; it may also serve as an efficient centralized state estimator. This result is reasonable because (*i*) the initial value processing already makes the values of all the states close to the solution even though the initial guesses of the states in the initial value processing procedure are all from $1.0\angle 0^\circ$, and (ii) the k isolated area state estimation problems in the initial value processing procedures and the k independent sets of linear equations in one iteration of the distributed block Jacobi method are all small-scale problems and can be solved very fast.

Remark 6.1: The purpose of citing the CPU time in this and subsequent test cases is simply for a comparison of the effectiveness of our state estimator with a standard state estimator in the sense of centralized state estimation. We have no intent to claim that our state estimator is best among all the centralized state estimators, because our emphasis is distributed state estimation.

Test Case 2 and Results: We assumed that bad data appear in the case. Working from the load flow solution used in case 1, we arbitrarily set the measurement errors on the real power flow measurements of transmission lines $66 \longrightarrow 67, 41 \longrightarrow 42, 29 \longrightarrow 31$, the reactive power flow measurements of transmission lines $2 \longrightarrow 1, 68 \longrightarrow 65, 116 \longrightarrow 68, 95 \longrightarrow 94, 108 \longrightarrow 109$, the real power injection measurements on the non-boundary buses 25, 90, and 2, and the reactive power injection measurements on the non-boundary buses 55 and 2. Note that some areas have multiple bad data. After the initial value processing procedures, the distributed block Jacobi method converged in 3 iterations for all areas. The CPU time reported on the Sun 4/60 workstation was 1.32 seconds. When applied to this case, the Newton-like method with a sparse matrix technique used 6.94 seconds of CPU time. This result further confirms that our method is effective in the sense of centralized state estimation. We applied the distributed bad data detection method to each individual area by both chi-square test (10) and normalized residual test (11). The values of α_i's in (10) are determined by a false alarm probability 0.05, and the values of $\beta_{i,j}$ in (11) are set equal to $3w_{i,jj}\sigma_{i,jj}$ for all i, j, where $w_{i,jj}$ is the jth diagonal term of the sensitivity matrix W_i of area i, and $\sigma_{i,jj}$ is the jth diagonal term of R_i. Bad data were detected in all eight areas. Then, using the measurement compensation and linear residual calculation method, we successfully identified the presumed single and/or multiple bad data in each area. After eliminating the bad data, we re-estimated the states using our distributed state estimator and the new set of measurements and found that the final estimated states agreed with the standard solution. These results show that our method is a complete distributed state estimator, because of its distributed bad data processing capability.

Test Case 3 and Results: We assumed in this case that there were bad data in the measurements, and the locations and measured values of these bad data were assumed to be the same as in case 2. We arbitrarily assumed that there were some communication errors during the execution of the distributed block Jacobi method. In between the 1st and 2nd iteration of the distributed block Jacobi method, there were communication errors in the communication links from A1 to A4 and from A2 to A7. The states involved were those of buses 65, 23, 26, 27, and 114. The false voltage magnitude and phase angle of bus 65 were 0.535 p.u. less and 0.327 radians more than the correct values, respectively. For bus 23, the false voltage magnitude and phase angle were 0.427 p.u. more and 0.631 radians less than the correct values, respectively. For bus 26, the false voltage magnitude and phase angle were 0.328 p.u. more and 0.519 radians more than the correct values, respectively. For bus 27, the false voltage magnitude and phase angle were 0.147 p.u. less and 0.732 radians less than the correct values, respectively. For bus 114, the false voltage magnitude and phase angle were 0.369 p.u. less and 0.278 radians less than the correct values, respectively.

Including this erroneous iteration, the distributed block Jacobbi method converged in 4 iterations for all areas; compared to test case 2, only one extra iteration is used in this case. This is because (i) the communication errors occur only between the 1st and the 2nd iteration, and (ii) the areas A4 and A7 lie in the outmost tier. In general, if the communication errors occur between areas that lie in the tiers closer to tier 1, more iterations are needed to correct those errors.

The estimated states were almost exactly the same as those in case 2. Consequently, the results of the distributed bad data processing and the re-estimation of the states were exactly the same as that of case 2.

Test Case 4 and Results: In this case, we simulated the numerical outcomes in the presence of partial asychronism in the distributed block Jacobi method. We assumed that there were bad data, and the locations and the measured values of these bad data were the same as in case 2. Due to the different size of the areas and various communication delays between computer systems, in different areas the linear equations (8) may need to be solved a different number of times in the partially asynchronous block Jacobi method. We arbitrarily set the frequency of solving (8) in all areas as follows: when areas A3 and A5 solve (8) once, then areas A4, A6, and A7 solve (8) twice, and areas A1, A2, and A8 solve (8) three times. We will refer to the period of solving (8) in the above frequency as one cycle instead of one iteration of the partially asynchronous block Jacobi method. Note that each area uses the available updated states of the external reference boundary buses when solving (8). This is more or less a partially asynchronous situation. In this case, after the initial value processing procedures, the partially distributed block Jacobi method converged after 2 cycles. Then, using the same distributed bad data detection and identification procedures as in case 2, we identified all the bad measurements and eliminated them. Next, when re-estimating the states based on the new set of measurements, after the initial value processing procedures, the partially asynchronous block Jacobi method took one cycle to converge. The largest deviation of the voltage magnitude (p.u) and phase angle (radians) from the standard solution was less than 10^{-4}.

Remark 6.2: The setup of testing the partial asynchronism in this case seems simplistic; however, this is the limitation of simulating a partially asynchronous distributed algorithm in a sequential computer. To overcome this limitation, we are currently conducting a research of using transputers to simulate our distributed state estimator.

VII. Conclusion

We have presented an implementable distributed state estimator and distributed bad data processing for electric power systems. This implementable distributed state estimator provides a foundation for the future development of decentralized control for large electric power systems. To adapt to the real-world system where the measurement system and areas of local control center are predefined, only slight modifications on the measurements placed on the boundaries of areas are needed; for instance, it is possible that we may just disregard some redundant measurements placed on the boundaries of areas to make each area a maximal non-critical area.

One advantage of the proposed estimator that was not foreseen at the beginning of this research is that this distributed state estimator can serve as an efficient centralized state estimator as well, as we have shown in our test results. Since each local computer system in the network can perform its own computations independently (in the sense of partial asynchronism), the processing speed of the distributed state estimator carried out by the computer network should be fast, because all the local computer systems can process in parallel asynchronously. Thus, in addition to being implementable, the proposed distributed state estimator is also a very efficient state estimator.

Acknowledgment

This work was supported by National Science Council of Republic of China under Grant NSC82-0404-E009-166.

References

[1] S. Y. Lin, "A distributed state estimator for electric power systems," *IEEE Trans. on Power Systems,* vol.7, no.2, pp.551-557, May 1992.

[2] G.N. Korres and G.C. Contaxis, "A reduced model for bad data processing." *IEEE Trans. on Power Systems,* vol.6, no.2, pp.550-557, May 1991.

[3] K.A. Clements, G.R. Krumpholz and P.W. Davis, "Power system state estimation residual analysis: an algorithm using network topology." *IEEE Trans. on Power Apparatus and Systems,* vol. PAS-100, no.4, pp.1779-1786, April 1981.

[4] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and distributed computation*, Prentice-Hall International Limited, London, 1989.

[5] G.R. Krumpholz, K.A. Clements and P.W. Davis, "Power system observability: a practical algorithm using network topology." *IEEE Trans. on Power Apparatus and Systems,* vol. PAS-99, no.4, July/Aug 1980.

[6] A.Monticelli and A,Garcia, "Reliable bad data processing for real-time state estimation." *IEEE Trans. on Power Apparatus and Systems*, vol. PAS-102, no.5, pp.1126-1139, May 1983.

[7] A.Monticelli, F.F.Wu and Maosong Yen, "Multiple bad data identification for state estimation by combinatorial optimization."*Proc. of the PICA conf.* pp.452-460, May 1985.

[8] L.Mili, Th.Van Cutsem and M.Ribbens-Pavella, "Hypothesis testing identification: a new method for bad data analysis in power system state estimation." *IEEE Trans. on Power Apparatus and Systems*, vol. PAS-103. no.11, pp.3239-3254, November 1984.

[9] I.W.Slutsker, "Bad data identification in power system state estimation based on measurement compensation and linear residual calculation." *IEEE Trans. on Power Systems*, vol.4, no.1, pp.53-60, February 1989.

Appendix

Theorem of Convergence: (a) Let K_1 be the Lipschitz constant such that $\| \nabla J(x) - \nabla J(y)\|_2 \leq K_1\|x-y\|_2, \forall x, y \in \Re^{N'}$, where $N' = 2N-1$ and N is the total number of the buses of the system. (b) Let K_2 be the largest number of $\|C(\mathbf{x}_i(t))^{-1}\|_2$, $i = 1, \cdots, k$, that ever occurs in any time t. Then if the partially asynchronous assumptions hold, there exists a positive real number γ_0 whose value depends on N', B, K_1, and K_2 such that if $0 < \gamma < \gamma_0$, then $\lim_{t\to\infty} \nabla J(\mathbf{x}(t)) = 0$.

Remark A.1: The value $\gamma_0 = \frac{1}{(1+B+N'B)K_1K_2}$ has been derived in [4].

Remark A.2: The proof of this theorem follows directly from the general theorem in [4].

Autobiography

SHIN-YEU LIN was born in Taiwan, ROC, on June 26, 1953. He received the B.S. degree in electronics engineering from National Chiao Tung University, the M.S. degree in electrical engineering from University of Texas at El Paso and the D. Sc. degree in systems science and mathematics from Washington University in St. Louis, Missouri, in 1975, 1979, and 1983 respectively.

From 1984 to 1985, he was with Washington University working first as a Research Associate and then a Visiting Assistant Professor. From 1985 to 1986, he was with GTE Laboratories working in the Switching Department as a Senior MTS. He joined the Department of Control Engineering at National Chiao Tung University in 1987 and has been a Professor since 1992. His major research interests are Large-Scale Power Systems, Optimization Theory and Applications, and Distributed and Parallel Computations.

CH'I-HSIN LIN was born in Taiwan, ROC, on Aug. 29, 1965. He received the B.S. degree from Feng Chia University, Taiwan, and the M.S. degree from National Tsing Hua University, Taiwan, in 1989 and 1991, respectively, both in electrical engineering.

He has been studying for his Ph. D. degree in control engineering at Chiao Tung University, Taiwan, since September 1991.