State Estimation Distributed Processing

Reza Ebrahimian and Ross Baldick

***Abstract*—This paper presents an application of a parallel algorithm to Power Systems State Estimation. We apply the Auxiliary Problem Principle to develop a distributed state estimator, demonstrating performance on the Electric Reliability Council of Texas (ERCOT) and the Southwest Power Pool (SPP) systems.**

I. Introduction

TO HOST SCADA and Energy Management System software for operations of power systems, utilities have historically used mid-size computers to handle the tasks automatically and to provide interface for real-time interactive intervention by the operating staff in the control center of a control area. However with advancements of small computer technologies and networking, it is becoming attractive to use distributed processing. Although the emerging structure of the "independent system operator" (ISO) may link several utilities, distributed computing is likely to be preferable compared to a completely centralized implementation.

Traditionally, the maximum likelihood weighted least-squares method is applied to the state estimation problem, yielding a formulation that is an approximately quadratic and convex problem, which typically has a single optimum solution. The novelty of our research relates to the development of algorithms for distributed processing. We apply the "Auxiliary Problem Principle" (APP) [4] to the state estimation problem.

In Section II, we develop and present the mathematical equations necessary to apply the APP to form the distributed algorithm. In Section III, we describe the use of MATLAB [7], [8] to develop first centralized and then distributed state estimation software for comparison purposes. Several test case studies representing ERCOT and SPP systems demonstrate the effectiveness of the algorithm and we discuss bad data detection. We conclude in Section IV.

II. Decomposition and Distributed Implementation

A. Centralized State Estimation

Traditional maximum likelihood weighted least-squares state estimation calculates the voltage magnitudes and angles (assuming a voltage angle reference). In this method the objective is to minimize the sum of the squares of the weighted deviations of the estimated variables from the actual measurements [16]

$$\min_x J(x) = \min[z^{meas} - f(x)]^\dagger [R^{-1}][z^{meas} - f(x)], \quad (1)$$

where:

$J(x)$	is the objective function,
R	$= \mathrm{diag}[\sigma^2]$,
σ^2	is a vector of variances of the measurement errors,
f	is a vector of functions describing the measurements,
z^{meas}	is a vector of the measurements,
x	is a vector of the voltage magnitudes and angles, and
superscript †	denotes transpose.

Therefore, if the system is observable then the Gauss–Newton update equations [16] for this nonlinear optimization problem are:

$$\Delta x = \begin{bmatrix} \Delta|E| \\ \Delta\theta \end{bmatrix} = [H^\dagger R^{-1} H]^{-1} H^\dagger R^{-1}[z^{meas} - f(x)],$$
$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)},$$

where:

H	is the Jacobian of vector $f(x)$,
$\lvert E\rvert$ and θ	are vectors of voltage magnitude and angles, and superscript in parenthesis indicates the iteration count, so that
$x^{(k)}$	is the value of iterate at the kth iteration.

B. Distributed State Estimation

1) Problem Illustration and Formulation: Based on previous experience of applying APP to the Optimal Power Flow (OPF) problem [10], [11], we applied APP to distributed state estimation. To maximize the practical applicability, we formulated the problem such that: it is highly compatible to previous real world implementations; only a small amount of inter-processor communication is required; bad data analysis could be performed; and, such that the distributed state estimator yields the same solution as centralized state estimation. We believe that this is a significant achievement compared with other approaches to distributed and hierarchical state estimation such as described in [2], [3], [5], [6], [9], [12]–[14], [17], [18].

In the following paragraphs we will describe the application of the Auxiliary Problem Principle to the state estimation problem. This description is paraphrased from [10], [11], but ultimately depends on the properties of linearized augmented Lagrangians described in [4].

To develop a distributed state estimator consider Fig. 1, which shows a 3-bus system lying in Areas A and B. This system includes the border bus Bbetween areas *A* and *B* that is common to both areas.

Let x be the vector of voltage magnitudes and angles for Area *A*, including the voltage magnitude $|E_a|$ and angle θ_a at the

Manuscript received May 14, 1999; revised February 23, 2000. R. Baldick was supported in part by the National Science Foundation under Grant ECS-9457133.

The authors are with The University of Texas at Austin, TX 78712.

Publisher Item Identifier S 0885-8950(00)10354-2.

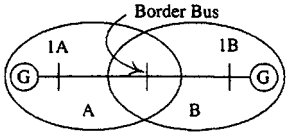

Fig. 1. Border bus between areas *A* and *B*.

border. Let y be the vector of voltage angles and magnitudes for Area *B*, including the voltage magnitude $|E_b|$ and angle θ_b at the border (this convention is slightly different to that in [11]). That is, x and y both include the voltage angle and magnitude at the border. We must require that $\theta_a = \theta_b$ and $|E_a| = |E_b|$ for x and y to be consistent. Further, we must also require that the real and reactive flow across the border be consistent. Now we can express the objective function in (1) as: $J_a(x) + J_b(y)$.

Consider the real and reactive power flow and the voltage angle and magnitude at the border. We can express these quantities in terms of x. That is, we can find the function f_a such that the vector of real and reactive power flows and voltage angles and magnitudes at the border are given by:

$$f_a(x) = [p_a(x) \quad q_a(x) \quad |E_a|(x) \quad \theta_a(x)]^\dagger.$$

The last two entries of f_a simply pick out the two appropriate entries of x corresponding to the voltage magnitude and angle. That is $|E_a|(x) = |E_a|$ and $\theta_a(x) = \theta_a$. The first two entries of f_a evaluate the real and reactive flow across the borders in terms of the vector x. Similarly, we can find a function f_b that expresses these same quantities in terms of y:

$$f_b(y) = [p_b(y) \quad q_b(y) \quad |E_b|(y) \quad \theta_b(y)]^\dagger;$$

where $|E_b|(y) = |E_b|$ and $\theta_b(y) = \theta_b$. For a valid solution, x and y must be such that the real and reactive power and angle and magnitude match at the border. That is: we must require $f_a(x) = f_b(y)$. Then the maximum likelihood weighted least-squares problem is:

$$\min_{(x,\,y)} \{J_a(x) + J_b(y) | f_a(x) = f_b(y)\}.$$

To solve this problem we will dualize the constraint $f_a(x) = f_b(y)$; however, to improve convergence we add a quadratic term. The problem becomes:

$$\min_{(x,\,y)} \left\{ J_a(x) + J_b(y) + \frac{\gamma}{2} ||f_a(x) - f_b(y)||^2 | f_a(x) = F_b(y) \right\},$$

where γ is any positive constant. Note that the added quadratic term does not change the problem because at any solution: $f_a(x) = f_b(y)$. Now to separate this objective for a distributed implementation we apply a decomposition algorithm referred to as the Auxiliary Problem Principle (APP) [1], [4], which is an iterative algorithm involving linearizing the augmented Lagrangian. This will yield two sub-problems and a multiplier update for evaluating x and y at each successive iteration of the form:

$$x^{(k+1)} = \arg\min_x \left\{ \begin{array}{l} J_a(x) + \dfrac{\beta}{2} ||f_a(x) - f_a(x^{(k)})||^2 \\ +\gamma f_a^\dagger(x)(f_a(x^{(k)}) - f_b(y^{(k)})) \\ +[\lambda^{(k)}]^\dagger (f_a(x)) \end{array} \right\}, \tag{2}$$

$$y^{(k+1)} = \arg\min_y \left\{ \begin{array}{l} J_b(y) + \dfrac{\beta}{2} ||f_b(y) - f_b(y^{(k)})||^2 \\ -\gamma f_b^\dagger(y)(f_a(x^{(k)}) - f_b(y^{(k)})) \\ -[\lambda^{(k)}]^\dagger (f_b(y)) \end{array} \right\}, \tag{3}$$

$$\lambda^{(k+1)} = \lambda^{(k)} + \alpha(f_a(x^{(k+1)}) - f_b(y^{(k+1)})), \tag{4}$$

where:

- k is the iteration number,
- λ is the Lagrange multiplier, and
- α and β are constants that must lie in particular ranges to guarantee convergence and can be tuned by trial and error.

The distributed implementation would require solving (2) or (3) separately in each area, exchanging border values between areas, and updating λ according to (4).

2) Mathematical Developments and Implementations: At the core of this software, we use the equations and algorithms of centralized state estimation. Prototype software was developed for centralized state estimation, considering sparsity issues and information matrix conditioning techniques. These features carry over to the distributed implementation.

To develop the necessary equations for solving distributed state estimation problems we consider area A (all the development will also apply to area B, mutatis mutandis). The objective in (2) is:

$$\begin{aligned} \hat{J}_a(x) = J_a(x) &+ \frac{\beta}{2} ||f_a(x) - f_a(x^{(k)})||^2 \\ &+ \gamma f_a^\dagger(x)(f_a(x^{(k)}) - f_b(y^{(k)})) \\ &+ [\lambda^{(k)}]^\dagger (f_a(x)), \end{aligned} \tag{5}$$

where, $\hat{J}_a(x)$ is the objective function for the area A . If area A owns t ties then the augmented Lagrangian of (5) would become, after some manipulations as shown below in (6).

The last term in (6) is constant, therefore, its derivative is zero. If we define virtual measurement z_{ai}^{virt} as:

$$z_{ai}^{virt} = \frac{\beta f_{ai}(x^{(k)}) - \gamma f_{ai}(x^{(k)}) = \gamma f_{bi}(y^{(k)}) - \lambda_i^{(k)}}{\beta},$$

$$\begin{aligned} \hat{J}_a(x) = J_a(x) &+ \sum_{i=1}^{4t} \frac{\beta}{2} \left[f_{ai}(x) - \frac{\beta f_{ai}(x^{(k)}) - \gamma f_{ai}(x^{(k)}) + \gamma f_{bi}(y^{(k)}) - \lambda_i^{(k)}}{\beta} \right]^2 \\ &+ \sum_{i=1}^{4t} \left\{ \frac{\beta}{2} (f_{ai}(x^{(k)}))^2 - \frac{\beta}{2} \left[\frac{-\beta f_{ai}(x^{(k)}) + \gamma f_{ai}(x^{(k)}) - \gamma f_{bi}(\gamma^{(k)}) + \gamma_i^{(k)}}{\beta} \right]^2 \right\} \end{aligned} \tag{6}$$

then the optimization problem of (6) is equivalent to:

$$\min_{(x)} \left\{ \sum_{i=1}^{m} \left(\frac{z_i^{meas} - f_i(x)}{\sigma_i} \right)^2 + \sum_{i=1}^{4t} \left(\frac{z_{ai}^{virt} - f_{ai}(x)}{\sigma_a} \right)^2 \right\}, \tag{7}$$

where:

- m is the number of actual measurements in area A,
- $4t$ is the number of virtual measurements introduced to reflect the APP terms into the form of the measurements equations,
- $f_i(x)$ is the measurement function corresponding to z_i^{meas}, and
- $\sigma_a = \sqrt{2/\beta}$.

The most important observation here is that this is now a state estimation problem with virtual measurements at the border.

The problem in (7) can be described as the minimum of the multiplication of the following matrices:

$$\min_x \begin{bmatrix} z^{meas} - f(x) \\ \text{------} \\ z_a^{virt} - F_a(x) \end{bmatrix}^{\dagger} \begin{bmatrix} R & 0 \\ 0 & R' \end{bmatrix}^{-1} \begin{bmatrix} z^{meas} - f(x) \\ \text{------} \\ z_a^{virt} - f_a(x) \end{bmatrix}, \tag{8}$$

or

$$\min_x \{[\hat{z}^{meas} - \hat{f}(x)]^{\dagger} [\hat{R}]^{-1} [\hat{z}^{meas} - \hat{f}(x)]\}, \tag{9}$$

where: $f(x)$ is the vector of measurement equations for area A, R' is a diagonal matrix of the virtual variances σ_a^2 and,

$$\hat{z}^{meas} = \begin{bmatrix} z^{meas} \\ \text{---} \\ z_a^{virt} \end{bmatrix}, \quad \hat{f}(x) = \begin{bmatrix} f(x) \\ \text{---} \\ f_a(x) \end{bmatrix} \quad \hat{R} = \begin{bmatrix} R & 0 \\ 0 & R' \end{bmatrix}.$$

To further describe the virtual measurements at the border virtual buses, we present the border virtual measurements as follows:

$$|E_{ai}^{virt}| = \frac{\beta |E_{ai}|^{(k)} - \gamma |E_{ai}|^{(k)} + \gamma |E_{bi}|^{(k)} - \lambda_i^{(k)}}{\beta},$$

$$\theta_{ai}^{virt} = \frac{\beta \theta_{ai}^{(k)} - \gamma \theta_{ai}^{(k)} + \gamma \theta_{bi}^{(k)} - \lambda_i^{(k)}}{\beta},$$

$$p_{ai}^{virt} = \frac{\beta p_{ai}^{(k)} - \gamma p_{ai}^{(k)} + \gamma p_{bi}^{(k)} - \lambda_i^{(k)}}{\beta},$$

$$q_{ai}^{virt} = \frac{\beta q_{ai}^{(k)} - \gamma q_{ai}^{(k)} + \gamma q_{bi}^{(k)} - \lambda_i^{(k)}}{\beta}.$$

To solve (9), which describes the implementation of distributed state estimation, we define $\hat{H}$ as a Jacobian matrix of size $(m+4t) \times (s+4t)$ (where s is the number of state variables) and of the following form:

$$\hat{H} = \left[\frac{\partial \hat{f}}{\partial x} \right].$$

We update x according to:

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}.$$

With some mathematical manipulation, we can show that the iterative solution for (8) is of the form:

$$\Delta x = \left[\hat{H}^{\dagger} \hat{R}^{-1} \hat{H} \right]^{-1} \hat{H}^{\dagger} \hat{R}^{-1} \Delta \hat{z}, \tag{10}$$

$$x^{(l+1)} = x^{(l)} + \Delta x^{(l)}, \tag{11}$$

Initialize variables
Telemeter measurements and topology data to control area computer
Calculate virtual measurements at the virtual border buses
Repeat{
  Increment k for APP algorithm iteration
  Solve state estimation for each area utilizing their own computers (All areas simultaneously)
  Exchange $f_a(x^{(k)})$ and $f_b(y^{(k)})$ between area computers (that is border variables for all areas)
  Update λ
} **Until** $f_a(x^{(k)})$ and $f_b(y^{(k)})$ converge to within tolerance

Fig. 2. Distributed state estimation algorithm.

$$\lambda^{(k+1)} = \lambda^{(k)} + \alpha (f_a(x^{(k+1)}) - F_b(y^{(k+1)})), \tag{12}$$

where

$$\Delta \hat{z} = \begin{bmatrix} z^{meas} - f(x) \\ \text{------} \\ z^{virt} - f_a(x) \end{bmatrix},$$

and l is iteration number for each individual area, and k is the iteration number for multiple area parallel solutions associated with APP algorithm. The counter k increments when all areas reach convergence. We refer to k as the outer loop iteration parameter and we refer to l as the inner loop iteration counter. Hence, at each Lagrange multiplier $\lambda^{(k)}$ update, each area is solved separately with $l_a, l_b, \cdots$, iterations. Using (10)–(12) we can solve power system state estimation problem in a distributed fashion. In the next subsection, we present the algorithm that utilizes these equations.

C. Distributed State Estimation Algorithm and Communication Issues

We present the algorithm to implement the distributed state estimation in Fig. 2, which is essentially the same as the OPF algorithm described in [10], [11]. The measurements for each area are telemetered to the computer for that area only, and only the computed border variables are exchanged between the adjacent areas. The neighboring areas will interchange the border variables at each iteration and calculate the updated variables for the next iteration.

At each iteration, a central computer will be informed of the status of each area for convergence. If all areas have met the convergence criteria, then the central computer will inform all areas that the entire interconnection has converged. In our implementation, we used a central controlling computer to perform these exchanges; however, they could be implemented with communication between adjacent areas only.

The amount of data communicated between each area and the central computer is very small. It is equal to the number of ties times 4 (voltage angle, voltage magnitude, real and reactive branch flows). The amount of computations conducted by this central computer at each iteration (basically checking for convergence at each iteration) is very small. Therefore the time required for telecommunications and computation at each iteration is likely to be negligible compared to the time to perform state estimation for an area. With a traditional centralized implementation the typical length of the communication path for telemetering data is on the order of the radius of the entire

TABLE I
CASE STUDY SYSTEMS

Case No.	Total Buses	No. Areas	Number of Branches in Each Area	Number of Buses in Each Area	Total Ties	Total Branches
1	238	2	196,104	153,85	8	300
2	809	4	196,104,393,367	153,85,279,292	13	1060
3	1567	6	196,104,393,367,301,602	153,85,279,292,236,522	46	1963
4	2529	8	196,104,393,367,301,602,604,583	153,85,279,292,236,522,484,478	142	3150
5	4972	8	428,113,819,935,584,687,1305,879	360,95,696,788,498,632,1139,764	136	5750
6	8047	8	2192,413,1624,1208,1412,2024,644,1908	1892,344,1338,994,1071,1054,434,920	190	11425

system, whereas in our distributed implementation the typical length of data path would be on the order of the radius of the regions. Telecommunication bottlenecks should be less significant in the case of the distributed implementation, even with the small amount of additional border data exchange.

III. CASE STUDIES AND RESULTS

To illustrate the convergence properties and the effectiveness of this algorithm we present several case studies using the Electric Reliability council of Texas (ERCOT) and Southwest Power Pool (SPP) systems. Table I gives a summary of the case study systems where: column 2 is the total number of buses including the border buses; column 3 is the number of areas; column 4 lists the number of branches in each area; column 5 lists the number of buses in each area; column 6 shows the total number of ties between the areas and the last column shows the total number of branches. Case studies 1, 2, 3, and 4 present the division of a 2529 bus representation of the ERCOT system into 8 areas starting with 2 areas then adding 2 areas until completion of the entire interconnection. These are essentially the same systems studied in [10], [11]. In addition, Case 5 is a 4972 bus representation of the ERCOT system decomposed into 8 areas. The two ERCOT (2529 bus and 4972 bus systems) cases do not have equivalent configurations and are not derived from each other. Case 6 is an 8047 bus representation of the SPP [15] system decomposed into 8 areas. These case studies intend to show the effectiveness of the algorithm with practical large-scale systems.

A. Regionalization

The speed of solution of the distributed state estimator is predominantly a function of the speed of the slowest system to reach a solution and the number of the outer loop iterations assuming that the inter-processor communication is relatively fast. The solution time for each area is not only a function of the size of the system, but also its configuration, the types, location and number of measurements, initial values, bad data and noise. The number of outer loop iterations depends on each individual system's configuration and how all areas interconnect. If we divide the system such that there is not much difference between most areas' solution times, then the performance of the distributed state estimator becomes largely a function of the number of the outer loop iterations. The following paragraphs describe our approach to dividing systems. We devised these methods of decomposition based on trial and error to reach a reasonable individual area solution time and number of outer loop iterations. Systematic approaches to dividing systems that yield optimum results is an area of future research.

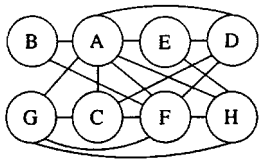

Fig. 3. Diagram of the ER-COT 2529 bus sytem interconnection after decomposition.

We divide the ERCOT 2529 bus, ERCOT 4972 bus and the SPP 8047 bus representation systems into 8 areas for our distributed implementation. In the division of the ERCOT systems, we leave most of the larger areas in their original divisions based on constituent companies and combine some of the smaller areas; however, we break the largest area into two areas. If we encounter islands in any area after the division, we rearrange the areas to ensure an internally interconnected system. Fig. 3 shows the diagram of the ERCOT 2529 bus system interconnection after decomposition. We will use this case in Section III-D to present the performance of the distributed state estimator relating to bad data detection. For the SPP 8047 bus representation system, first we combined the contiguous areas without breaking any of the areas such that the number of buses in each area is greater than 500 and less than 2000. We check each area for possible islands, and if we encounter islands we rearrange the areas such that an internally interconnected system emerges.

B. Performance of Distributed State Estimation Algorithm

We have developed prototype software in MATLAB and the following presentations are results produced using this software. We define *actual wall clock* time as the wall clock time for two to eight separate computers to solve the distributed state estimation. This is equal to the summation over all the outer iterations of the wall clock time for the slowest converging area at each outer iteration plus the time required for the central computer to check for convergence at each outer iteration. The case study results presented in this paper are conducted over a local area network with negligible communication delays. However, in an actual implementation for a large power system network, communication bottlenecks can increase the wall clock time for both the centralized as well as the distributed implementations, with potential communication delays being worse for the centralized implementation.

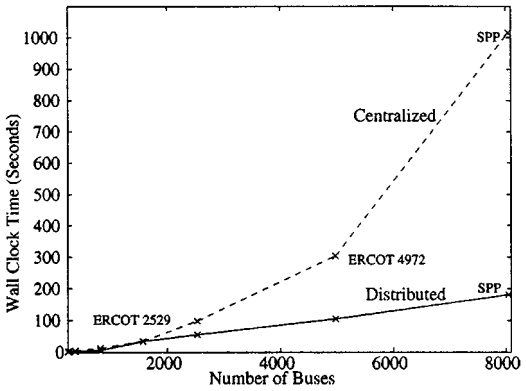

Fig. 4. Wall clock versus number of buses for distributed and centralized implementation (Sun Ultra workstations).

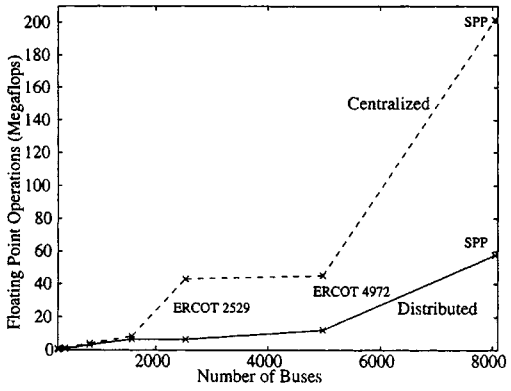

Fig. 5. Megaflops versus number of buses for distributed and centralized implementation (Sun Ultra workstations).

Fig. 4 shows the wall clock time versus the number of buses for both the centralized and distributed implementations for the case studies of Table I. The actual wall clock time is approximately equal to the wall clock time for the computer in the slowest converging area. For all case studies, the distributed implementation time is less than the centralized. Further, as the number of buses increases, the advantage of the distributed implementation over the centralized becomes more pronounced.

The wall clock time presented here using the MATLAB interpretive language is significantly higher than the wall clock time using compiled code and is not reflective of performance in a production environment. However, it is reasonable that the results presented here are proportional to conducting the same computations with compiled codes and provide qualitative information that is useful in judging the performance of an efficient implementation. Two factors that might affect this proportionality are use of loops, and memory fragmentation caused by resizing arrays within a MATLAB programs. We have taken special care to replace *for* and *while* loops with "vectorized" code, and preallocate large arrays to avoid memory fragmentation. For very large systems beyond 8000 buses, the plot of Fig. 4 may suggest a speed up of greater than 8. However, this may not be indicative of actual implementations; therefore the plot should not be extrapolated to larger systems.

Fig. 5 shows the megaflops versus the number of buses for both the centralized and distributed implementations for the case studies of Table I. The megaflops for the distributed implementation are *maximum megaflops* similar to the actual wall clock in Fig. 4, where the maximum megaflops is the summation over all the outer iterations of the largest total megaflops over individual areas at each outer iteration plus the total megaflops for the convergence check at each outer loop iteration. The megaflops for the distributed in all cases are less than the megaflops for the centralized because we combine megaflops by adding the largest number of megaflops over the processors for each iteration. The floating point operations for the centralized implementations of ERCOT 2529 and ERCOT 4972 systems appear to be almost equal. This is due to the configurations of these systems, and information matrix preconditioning and sparsity techniques that we have employed. However, typically, as the number of buses increases, the number of floating point operations also increases.

Table II shows the number of ties, state variables, measurements, outer loop iterations and the redundancy η for each case. Redundancy is the ratio of the number of the measurements to the number of the state variables. The total number of state variables in the distributed implementation is one more than the centralized implementation because we do not have a reference voltage angle in the distributed implementation. We calculate the final angles based on an assumed reference angle. That is, after reaching a solution, the computed "reference bus" angle is the amount that all angles may be shifted to make the angle estimation comparable to the centralized implementation with an actual reference angle. For uniformity we choose as measurements the bus voltage magnitude of all the generators (with an error standard deviation of 0.002) and the real and reactive branch flows in both directions of selected branches (with an error standard deviation of 0.02).

C. Convergence and Stopping Criterion

To guarantee convergence, and reduce the number of outer loop iterations, α, β, and γ must lie in particular ranges for different systems. Using trial and error we found that for $\alpha = \gamma = 1$, and $\beta = 2$ the convergence is reliable for the systems presented in this paper. In most cases, by the second iteration, more than 90% of the border variables converge within a 0.001 per unit tolerance for voltage magnitude and real and reactive flows, and within a 0.03 radian tolerance for voltage angle. In all of our cases, all of the variables reach convergence within this tolerance in a maximum of 6 iterations. After completion of the solution, over 99% of the state variables are within 0.1% difference from the centralized implementation.

D. Implementation of Bad Data Detection in the Distributed State Estimator

Practical state estimators require detection of bad data to improve the accuracy of their information and to avoid divergence. The sum of the square of the residuals $J(x)$ calculated after the convergence is small if there are no bad measurements present. In the presence of bad data $J(x)$ will be large. Traditionally the *normalized* measurement residual r_i^{norm} [16] is used to detect bad data and is calculated as:

$$r_i^{norm} = \frac{(f_i(x) - z_i)}{\sigma_{ri}},$$

TABLE II
CASE STUDIES OUTER LOOP ITERATIONS

Case No.	No. Ties	Number of State Variables	Number of Measurements	η	No. Outer Loop Iter.
1	8	306,170	813,420	2.59	2
2	13	306,170,558,584	809,420,1632,1505	2.70	2
3	46	306,170,558,584,472,1044	801,419,1628,1485,1234,2481	2.57	5
4	142	306,170,558,584,472,1044,968,956	796,419,1626,1485,1217,2445,2432,2171	2.49	5
5	136	720,190,1392,1576,996,1264,2278,1528	1723,459,3381,3788,2355,2789,5249,3532	2.34	5
6	190	3784,688,2676,1988,2142,2108,868,1840	8828,1649,6539,4858,5689,8224,2621,7802	2.87	6

TABLE III
ONE VOLTAGE MAGNITUDE AND ONE BRANCH REAL FLOW BAD DATA (TOTAL OF TWO) IN AREA A, UP TO ALL AREAS

Bad Data In Areas	Centr. Wall Clock	Centr. Mega-flops	Distr. Wall-Clock	Distr. Mega-flops
None	98.2	43.05	55.6	6.47
A	147.3	118.39	55.6	6.47
A–B	200.3	193.73	55.6	6.47
A–C	248.5	269.96	55.6	8.73
A–D	296.7	344.40	55.6	8.73
A–E	344.9	419.74	55.6	8.73
A–F	392.9	495.08	69.9	8.73
A–G	440.9	570.41	69.9	8.73
A–H	488.7	645.75	69.9	8.73

TABLE IV
THREE VOLTAGE MAGNITUDE AND THREE BRANCH REAL FLOW BAD DATA (TOTAL OF SIX) IN EACH AREA AT A TIME

Bad Data in Area	Centr. Wall-Clock	Centr. Mega-Flops	Distr. Wall-Clock	Distr. Mega-Flops
None	98.2	43.05	55.6	6.47
A	224.2	260.34	55.6	6.47
B	236.4	263.04	55.6	6.47
C	242.8	268.05	55.6	13.25
D	248.5	267.71	55.6	6.47
E	249.8	275.32	55.6	6.47
F	243.3	268.56	98.48	9.47
G	238.7	261.89	86.70	11.35
H	250.9	271.82	95.60	10.21

where:

f_i is the estimate,

z_i is the measurement, and

σ_{ri} is the standard deviation of the measurement residual r_i.

If the absolute value of r_i^{norm} is greater than three, then the associated measurement is assumed to be wrong and is removed from the measurements and the state estimator is resolved.

We have implemented this method of bad data detection to demonstrate the performance of the distributed state estimator in the presence of bad data. This method, although not the fastest, is reliable and provides a fair comparison with the centralized state estimator.

Each area at each outer loop iteration is examined for bad data. In all of the cases presented in this paper the distributed state estimator has detected bad data at the first outer loop iteration. However, in a worst case scenario, it is possible for the bad data to not be detected until the last outer loop iteration. Even with such a scenario, the distributed implementation solves faster than the centralized for a realistic number of bad data.

In a distributed implementation if the gross errors are spread out in different areas then bad data detection will take much less time than the centralized implementation because solving one area out of all the areas is much faster than resolving the entire system, so long as the gross errors can be reliably detected by each individual area. Even if all the gross errors are within one area, the distributed implementation reaches a final solution faster then the centralized so long as the bad data can be identified and discarded reliably. Table III shows the wall clock time and floating point operations for the ERCOT 2529 bus, 8 area representation system using the distributed and centralized algorithms with one voltage magnitude and one branch real flow gross error in each affected area starting with 2 gross errors in area A in the case given in row 1, and increasing to 2 gross errors in each area in the case given in row 8. This table shows that both the wall clock and megaflops performances of the distributed implementation are better than the centralized in the presence of bad data. It shows that the centralized wall clock and megaflops increase rapidly as the number of bad data increase; whereas, with the distributed implementation the wall clock and megaflops increase only when the slow converging areas contain bad data. Even in that case the increase is much less than the centralized implementation.

Table IV shows the wall clock time and megaflops for the ERCOT 2529 bus, 8 area using the distributed and centralized algorithms with 6 gross errors (3 voltage magnitude errors, and 3 branch real flow errors) in each area. This table perhaps shows the worst case performance scenario for the distributed implementation because each area containing the gross errors would solve seven times and assuming three inner loop iterations, this would result in a total of 21 iterations. However, even with this scenario, the wall clock and megaflops performance of the distributed are, by far, better than the centralized.

With the distributed implementation another issue is the presence of bad data near to the border buses and the ability to detect these accurately, since the information required to detect the bad data may need to "propagate" through the virtual measurements from an adjacent area. However, in our experience with our case studies this implementation detects bad data easily even if it is very close to the border buses.

Using the ERCOT 2529 bus, 8 area, Table V examines the ability of the distributed implementation to detect bad data from

TABLE V
BAD DATA DETECTION, AWAY FROM AND AT BORDERS

	Bad Data No. and Type	Centr. Wall-Clock	Centr. Mega-Flops	Distr. Wall-Clock	Distr. Mega-Flops
	None	98.2	43.05	55.6	6.47
At Borders of Area A	1 Volt. Mag.	122.75	80.07	55.6	6.47
	1 Real Flow	122.75	80.07	55.6	6.47
	3 Volt Mag.	171.85	156.03	55.6	6.47
	3 Real Flow	171.85	156.03	55.6	6.47
Away From Borders of Area A	1 Volt Mag.	122.75	80.07	55.6	6.47
	1 Real Flow	122.75	80.07	55.6	6.47
	3 Volt Mag	171.85	156.03	55.6	6.47
	3 Real Flow	171.85	156.03	55.6	6.47

meters that are located close to the borders with other areas and the ability to detect bad data from meters that are away from the borders. Cases 1 through 4 show results with bad data in area A at the border with area B, cases 5 through 8 show results with bad data in area A away from its borders. For the distributed implementation, the number of inner iterations for area A increases by the number of bad data times three, since it takes three inner loop iterations to solve area A. However, for the centralized implementation, the number of iterations increases by the number of gross errors times three (since it takes three iterations to solve the centralized state estimator). Table V shows the corresponding wall clock times and megaflops. Presence of bad injection measurements data instead of bad flow measurements data yields similar results. In summary we have shown that the distributed implementation detects bad data effectively and in all cases studied with less effort than a centralized implementation.

IV. CONCLUSIONS AND FURTHER STUDIES

In this paper, we have shown a robust distributed algorithm for power system state estimation with a minimal amount of modification required to existing state estimators, and demonstrated its effectiveness on ERCOT and SPP systems. With deregulation in the United States and the emergence of ISOs, large scale state estimation will become necessary to ensure secure operation of the electric power interconnections. Distributing the calculations onto multiple processors will become increasingly important. To our knowledge, our implementation of a distributed state estimator is the most practical and realistic that has been presented in the literature so far. In future studies, we would like to investigate the characteristics of each area and the ties between them as they relate to the convergence properties of the entire system. It may be potentially possible to reduce the number of outer loop iterations by some variations in the algorithm and system decomposition.

REFERENCES

[1] J. Batut and A. Renaud, "Daily generation scheduling optimization with transmission constraints: A new class of algorithms," *IEEE Trans. on Power Systems*, vol. 7, no. 3, pp. 982–989, Aug. 1992.

[2] C. W. Brice and R. K. Cavin, "Multiprocessor static state estimation," *IEEE Trans. on Power Apparatus and Systems*, vol. PAS-101, no. 2, pp. 302–308, Feb. 1982.

[3] K. A. Clements, O. J. Denison, and R. J. Ringlee, "A multi-area approach to state estimation in power system networks," in IEEE/PES Summer Meeting, July 1972, Paper C72 465-3.

[4] G. Cohen, "Auxiliary problem principle and decomposition of optimization problems," *Journal of Optimization Theory and Applications*, vol. 32, no. 3, pp. 277–305, Nov. 1980.

[5] T. Van Cutsem, J. L. Horward, and M. Ribben-Pavella, "A two-level static state estimator for electric power systems," *IEEE Trans. on Power Apparatus and Systems*, vol. PAS-100, no. 8, pp. 3722–3732, Aug. 1981.

[6] T. Van Cutsem and M. Ribben-Pavella, "Critical survey of hierarchical methods for state estimation of electric power systems," *IEEE Trans. on Power Apparatus and Systems*, vol. PAS-102, no. 10, pp. 3415–3424, Oct. 1983.

[7] D. M. Etter, *Engineering Problem Solving with MATLAB*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[8] W. Gander and J. Hrebicek, *Solving Problems in Scientific Computing Using Maple and MATLAB*, 2nd ed. Berlin/Heidelberg/New York: Springer-Verlag, 1993.

[9] 94 304H. Mukai, "Parallel multiarea state estimation," EPRI, Palo Alto, CA, Technical Report 1764-1, Jan. 1982.

[10] B. Kim, "A study on distributed optimal power flow," Ph.D. dissertation, University of Texas at Austin, Austin, 1997.

[11] B. H. Kim and R. Baldick, "Coarse-grained distributed optimal power flow," *IEEE Trans. on Power Systems*, vol. 12, no. 2, pp. 932–939, May 1997.

[12] H. Kobayashi, S. Narita, and M. S. A. A. Hamman, "Model coordination method applied to power system control and estimation problems," in *Proceedings of the 4th International Conference on Digital Computer Applications to Process Control*, 1974, pp. 114–128.

[13] 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.

[14] S. Y. Lin and C. H. Lin, "An implementable distributed state estimator and distributed bad data processing schemes for electric power systems," *IEEE Trans. on Power Systems*, vol. 9, no. 3, pp. 1277–1284, Aug. 1994.

[15] S. Miller, "Transmission access information Library," Commonwealth Associates Inc., Jackson, MI, Sept. 1997.

[16] A. J. Wood and B. F. Wollenberg, *Power Generation, Operation, and Control*, 2nd ed. New York, NY: Wiley, 1996.

[17] C.-C. Yang and Y.-Y. Hsu, "Estimation of line flows and bus voltages using decision trees," *IEEE Trans. on Power Systems*, vol. 9, no. 3, pp. 1569–1574, Aug. 1994.

[18] J. Zaborsky, K. W. Whang, and K. V. Prasad, "Ultra fast state estimation for the large electric power system," *IEEE Trans. on Automatic Control*, vol. AC-25, no. 4, pp. 839–841, Aug. 1980.

Reza Ebrahimian received his B.S. and Masters degrees in electrical engineering from Texas A&M University, and his Ph.D. degree from the University of Texas at Austin. He is currently a Senior Consulting Engineer at Austin Energy.

Ross Baldick received his B.Sc. and B.E. degrees from the University of Sydney, Australia and his M.S. and Ph.D. degrees from the University of California, Berkeley. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at the University of Texas at Austin.