[10] T. J. Su and C. G. Huang, "Robust stability of delay dependence for linear uncertain systems," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1656–1659, 1992.
[11] J. N. Franklin, *Matrix Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1968.
[12] R. K. Yedavalli and Z. Liang, "Reduced conservatism in stability robustness bounds by state transformation," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 863–866, 1986.

An Efficient Method for Unconstrained Optimization Problems of Nonlinear Large Mesh-Interconnected Systems

Shin-Yeu Lin and Ch'i-Hsin Lin

***Abstract*—We present a new efficient method for solving unconstrained optimization problems for nonlinear large mesh-interconnected systems. This method combines an approximate scaled gradient method with a block Gauss–Seidel with line search method which is used to obtain an approximate solution of the unconstrained quadratic programming subproblem. We prove that our method is globally convergent and demonstrate by several numerical examples its superior efficiency compared to a sparse matrix technique based method. In an example of a system of more than 200 variables, we observe that our method is 3.45 times faster than the sparse matrix technique based Newton-like method and about 50 times faster than the Newton-like method without the sparse matrix technique.**

I. Introduction

In this paper we consider the following unconstrained optimization problem for a nonlinear large mesh-interconnected system

$$\min_{x\in\Re^n} J(x) \tag{1}$$

where the objective function $J:\Re^n \to \Re$ is continuously differentiable, bounded from below and satisfies the Lipschitz condition that there exists a constant $K>0$ such that $||\nabla J(x_1) - \nabla J(x_2)||_2 \le K||x_1 - x_2||_2$, $\forall x_1, x_2 \in \Re^n$. A general descent algorithm, which is called a scaled gradient method in [1], for solving problem (1) uses the following iterations

$$x^{k+1} = x^k + \gamma^k s^k \tag{2}$$

where k denotes the iteration index, γ^k is a step-size, and s^k is the solution of

$$\min_{s\in\Re^n} \tfrac{1}{2} s^T C(x^k) s + \nabla J(x^k)^T s \tag{3}$$

in which $C(x^k)$ is a positive definite matrix.

For a large mesh-interconnected system, if $C(x^k)$ is selected so that (2) is a Newton or Newton-like method, $C(x^k)$ may be a sparse matrix. The solution of (3), which is the solution of the linear system

$$C(x^k)s = -\nabla J(x^k) \tag{4}$$

can be obtained using the sparse matrix technique, which is a very powerful means for solving linear equations in a circuit system [2] or an electric power system [3]. This technique effectively reduces the amount of computer memory needed and improves the computation time dramatically because it stores only nonzero elements and ignores operations involving zeros in the solution process [2]. Thus, a sparse matrix technique based Newton or Newton-like method is much more efficient than a method with quadratic convergence rate.

In this paper, we will present a new method for solving (1) for a large mesh-interconnected system to compete with a sparse matrix technique based method. Our method combines an approximate scaled gradient method with a block Gauss–Seidel with line search (BGSLS) method. Although each of the above methods is well known, combining them into one method is a new approach. The basic idea behind this method is to preserve the advantages and discard the disadvantages of the block Gauss-Seidel method. We have observed in many numerical experiments [4] that the block Gauss–Seidel method approaches its convergence point very fast in the first few iterations and then slows down around that point. This fact indicates that the block Gauss–Seidel method is better suited for use as a descent direction generator rather than as an optimization algorithm by itself. To improve the quality of the descent direction further, we use an exact line search at the end of each cycle of the block Gauss–Seidel method and thus form a BGSLS method. Executing the BGSLS method for a finite number of iterations with appropriate stopping criteria will generate the descent direction needed for the approximate scaled gradient method. We will show in this paper that our method is a globally convergent descent algorithm. It is difficult to give an analytical convergence rate for our method because of the nature of the method. However, since the major computation required by our method lies in the execution of the BGSLS method, our method will be efficient if the BGSLS method can generate an effective descent direction in several iterations. Intuitively, our method should be efficient and effective, for two reasons: i) only small minimization problems are involved in each iteration of the BGSLS method and ii) the exact line search used at the end of each iteration of the BGSLS method greatly improves the quality of the descent direction.

II. Solution Method

A. The Approximate Scaled Gradient Method

The approximate scaled gradient method [1] is

$$x^{k+1} = x^k + \gamma^k \hat{s}^k \tag{5}$$

where $\hat{s}^k$ is an approximate solution of (3) and γ^k is a step-size. To ensure global convergence, we use Armijo-type rule to determine the step-size by

$$\gamma^k = \beta^{m_k}\lambda \tag{6}$$

where $0<\beta<1$, $\lambda>0$, and m_k is the smallest nonnegative integer m that makes the following inequality hold for some positive constant K_2

$$J(x^k + \beta^m\lambda\hat{s}^k) - J(x^k) \le -\frac{K_2}{2}\beta^m\lambda||\hat{s}^k||_2^2. \tag{7}$$

Remark 2.1: a) Condition (7) is to ensure a sufficient decrement of the objective function obtained in each iteration of (5); the satisfaction of this condition serves as a terminating criteria for our Armijo-type step-size rule (6). b) As will be shown in Theorem 2.1, the matrix $\frac{1}{2}C(x^k) - K_2 I$ being nonnegative definite is a sufficient condition for (5) to converge. This sufficient condition provides a value for K_2.

Manuscript received October 19, 1993; revised March 8, 1994. This research work is supported by National Science Council in R.O.C. under Grants NSC-82-0404-E-009-166 and NSC-79-0404-E-009-48.

The authors are with the Department of Control Engineering, National Chiao Tung University, Hsinchu, Taiwan.

IEEE Log Number 9407217.

Lemma 2.1: Suppose (3) is solved by a descent iterative algorithm starting from $s = 0$ for any arbitrary number of iterations, and let $\hat{s}^k$ denote the final value of s. Then $\nabla J(x^k)^T \hat{s}^k < 0$.

This lemma can easily be verified by the fact that $\frac{1}{2}(\hat{s}^k)^T C(x^k)\hat{s}^k + \nabla J(x^k)^T \hat{s}^k < 0$. Thus, our approximate scaled gradient method (5) will be an efficient descent algorithm for solving (1) if the descent algorithm we employ to obtain $\hat{s}^k$ is efficient.

B. Block Gauss–Seidel with Line Search (BGSLS) Method

1) One Block Gauss-Seidel Cycle: Let us partition s into p subvectors such that $s = [s_1 s_2 \cdots s_p]^T$. Then one block Gauss–Seidel cycle is to perform

$$\min_{s_i} \frac{1}{2} s^T C(x^k) s + \nabla J(x^k)^T s \tag{8}$$

from $i = 1$ to p. In (8), the subvector s_i is taken as the vector of minimizing variables while the variables in the subvectors $s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_p$ are held fixed at their current values. Note that compared to (3), (8) is a small unconstrained minimization problem for every i.

Remark 2.2: On the partition of the s-vector, there are two extremes corresponding to $p = 1$ and $p = n$. The case of $p = 1$ is not the interest of this paper. For the case of $p = n$, the descent direction generated by (8) has very poor quality. Thus, a good partition should take the following two factors into account: a) computational burden of solving (8) and b) the quality of descent direction generated. In fact, the above two factors have conflicting interests; at this stage, we have not yet achieved an optimal way to partition the s-vector. Nonetheless, for a network-structure like system, it would be beneficial if each partitioned subnetwork is mesh-interconnected, and the sizes of all subnetworks do not differ much.

2) Optimal Step-Size: Let $s^{(i)}$ denote the value of s after solving (8) for s_i. Then $s^{(0)}$ represents the initial value and $s^{(p)}$ the final value of s for one block Gauss–Seidel cycle. Suppose $s^{(0)}$ is not the optimal solution of (3). Then the following inequality holds

$$\begin{aligned} &\tfrac{1}{2}(s^{(0)})^T C(x^k) s^{(0)} + \nabla J(x^k)^T s^{(0)} \\ &\quad > \tfrac{1}{2}(s^{(p)})^T C(x^k) s^{(p)} + \nabla J(x^k)^T s^{(p)}. \end{aligned} \tag{9}$$

Furthermore, because $C(x^k)$ is positive definite, $\frac{1}{2} s^T C(x^k)\, s + \nabla J(x^k)^T s$ is a convex function in s. Based on this fact, we may verify that

$$\begin{aligned} &\tfrac{1}{2}(s^{(p)})^T C(x^k) s^{(p)} + \nabla J(x^k)^T s^{(p)} \\ &\quad \geq \tfrac{1}{2}(s^{(0)})^T C(x^k) s^{(0)} + \nabla J(x^k)^T s^{(0)} \\ &\qquad + [C(x^k) s^{(0)} + \nabla J(x^k)]^T (s^{(p)} - s^{(0)})\}. \end{aligned} \tag{10}$$

Then, combining (9) and (10), we obtain the following lemma.

Lemma 2.2: Let : $s^{(0)}$ and $s^{(p)}$ denote the initial and final values of s-variables, respectively, for one block Gauss–Seidel cycle of the BGSLS method. Suppose $s^{(0)}$ is not an optimal solution of (3). Then

$$[C(x^k) s^{(0)} + \nabla J(x^k)]^T (s^{(p)} - s^{(0)}) < 0. \tag{11}$$

The above inequality implies that $s^{(p)} - s^{(0)}$ is a descent direction of $\frac{1}{2} s^T C(x^k) s + \nabla J(x^k)^T s$ at $s = s^{(0)}$.

Therefore, we can determine the exact optimal step-size $\hat{\alpha}$ to update the variable s by

$$s^u = s^{(0)} + \hat{\alpha}(s^{(p)} - s^{(0)}) \tag{12}$$

where s^u denotes the updated s, and the optimal step-size

$$\hat{\alpha} = -\frac{[C(x^k) s^{(0)} + \nabla J(x^k)]^T (s^{(p)} - s^{(0)})}{[s^{(p)} - s^{(0)}]^T C(x^k)[s^{(p)} - s^{(0)}]} \tag{13}$$

is obtained by solving the following one-dimensional minimization problem

$$\begin{aligned} \min_{\alpha \geq 0} \{ &\tfrac{1}{2}[s^{(0)} + \alpha(s^{(p)} - s^{(0)})]^T C(x^k)[s^{(0)} + \alpha(s^{(p)} - s^{(0)})] \\ &+ \nabla J(x^k)^T [s^{(0)} + \alpha(s^{(p)} - s^{(0)})]\}. \end{aligned} \tag{14}$$

3) One Iteration of the BGSLS Method: The following three operations form one iteration of the BGSLS method: i) execute one block Gauss–Seidel cycle; ii) determine $\hat{\alpha}$; and iii) update s^u. s^u will be the initial value $s^{(0)}$ for the next iteration of the BGSLS method.

4) Convergence of the BGSLS Method: This iterative BGSLS method will converge to the solution of (3), as described in the following lemma. The proof of the lemma is given in the Appendix.

Lemma 2.3: Assuming that there exists a constant $K_2 > 0$ such that $\frac{1}{2} C(x^k) - K_2 I$ is nonnegative definite for all x^k, then a) the BGSLS method is a descent method, and b) any limit point of the sequence generated by the BGSLS method is a solution of (3).

5) Stopping Criteria of the BGSLS Method: As pointed out in Section I, the BGSLS method approaches a solution point of (3) very quickly in the first few iterations; however, it then slows down around that point. In fact, it was this characteristic of the BGSLS method that gave us the idea of combining it with the approximate scaled gradient method. Therefore, to obtain the $\hat{s}^k$ needed in (5), we do not need to execute the BGSLS method until it converges. In fact, we can stop the BGSLS method after a finite number of iterations. Consequently, one of the stopping criteria of this method is if the improvement of the objective function satisfies

$$\frac{Q(t) - Q(t+1)}{Q(t)} \times 100\% < \xi\% \tag{15}$$

where $Q(t) = \frac{1}{2} s(t)^T C(x^k) s(t) + \nabla J(x^k)^T s(t)$, t denotes the iteration index of the BGSLS method, and $\xi\%$ is a preselected percentage. To ensure that (5) converges, another stopping criteria

$$\|s(t+1)\|_2 < K_1 \|\nabla J(x^k)\|_2 \tag{16}$$

for some $K_1 > 0$, for all $t > t'$, where t' is a finite positive integer, should also be satisfied. Then, the $s(t+1)$ which satisfies stopping criteria (15) and (16) will be set as $\hat{s}^k$. An explanation of the need for (16) will be given in the proof of Theorem 2.1. The existence of K_1 and t' is ensured by the following corollary, the proof of which is also given in the Appendix.

Corollary 2.1: Let $\{s(t)\}$ denote a sequence generated by the BGSLS method. Under the assumption of Lemma 2.3, there exists a t' such that $\|s(t)\|_2 > K_1 \|\nabla J(x^k)\|_2$ for some $K_1 > 0$ and for all $t > t'$.

C. The Algorithm for the Solution Method and its Convergence

Combining Lemma 2.1 and Lemma 2.3 then proves that our method is a descent method.

Lemma 2.4: The combination of the approximate scaled gradient method (5) and the BGSLS method, supplemented by the stopping criteria (15) and (16), is a descent method for solving (1).

1) The Algorithm: We are now ready to state our algorithm as follows: *Given data:* i) $s = [s_1, s_2, \cdots, s_p]^T$, ii) the values of $\lambda(>0)$ and $\beta(0 < \beta < 1)$ in Armijo-type rule, iii) the value of $\xi\%(\geq 0\%)$ in (15), and iv) a positive constant K_2 which meets the assumption in Lemma 2.3.

Step 0: Set the values $x^{(0)}$ and set $k = 0$.

Step 1: Set $t = 1$ and set $s(t) = 0$.

Step 2: Set $i = 1$ and $s^{(0)}(t) = s(t)$.

Step 3: Solve (8) to obtain $s^{(i)}$. If $i = p$, go to Step 5; otherwise, set $i = i + 1$ and repeat this step.

Step 4: Compute $s^{(p)}(t) - s^{(0)}(t)$ and determine $\hat{\alpha}(t)$ by (13).
Step 5: Update $s(t+1) = s^{(0)}(t) + \hat{\alpha}(t)[s^{(p)}(t) - s^{(0)}(t)]$.
Step 6: If (15) and (16) are satisfied, set $\hat{s}^k = s(t+1)$ and go to Step 7; otherwise, set $t = t+1$ and return to Step 2.
Step 7: Set $m = 0$.
Step 8: If $J(x^k + \beta^m \lambda \hat{s}^k) - J(x^k) \leq -(K_2/2)\beta^m \lambda \|\hat{s}^k\|_2^2$, set $\gamma^k = \beta^m \lambda$; otherwise, set $m = m+1$, and repeat this step.
Step 9: Update $x^{k+1} = x^k + \gamma^k \hat{s}^k$, set $k = k+1$, and return to Step 1.

2) Convergence of the Algorithm: The following theorem ensures the convergence of our algorithm. A proof of the theorem is given in the Appendix.

Theorem 2.1: Let $\{x^k\}$ denote the sequence generated by our algorithm. Under the assumption of Lemma 2.3, $\lim_{k\to\infty} \nabla J(x^k) = 0$.

Remark 2.3: The values of λ, β, and $\xi\%$ do not affect the convergence but will influence the efficiency of our method. Thus, these values can be determined empirically for individual systems.

3) Stopping Criteria of the Algorithm: From Theorem 2.1, we see that as $k \to \infty$, $\hat{s}^k \to 0$; however, for practical considerations, our algorithm will stop when $|\hat{s}^k|_\infty < \varepsilon$ for a reasonable accuracy.

III. Examples

Most electric power systems are nonlinear large mesh-interconnected systems. The weighted least squares problem in power system state estimation is a typical unconstrained optimization problem and thereby is an adequate example for demonstrating the computational efficiency of our method.

A. The Weighted Least Squares Problem in Power System State Estimation

The power system state estimation problem [6] can be briefly described as follows: For an l-bus[1] power system, the voltage magnitudes and phase angles of all buses constitute the states of the system. Because the bus phase angle cannot be measured, the states of the system have to be estimated based on available measurements, which are functions of the states, such as power transmission line flow, bus voltage magnitude, bus power injection, and transformer tap. All such measurements can be expressed in a general form as $z = h(x) + \eta$, where x is a $2l$-dimensional vector of state variables, z is an m-dimensional vector of measurements, h denotes m nonlinear measurement functions which are twice continuously differentiable, and η represents an m-dimensional Gaussian random vector of measurement errors with an $m \times m$ diagonal covariance matrix R. A common formulation [5] of this state estimation problem is to solve the following weighted least squares problem[2]

$$\min_x J(x) (= \tfrac{1}{2}[z - h(x)]^T R^{-1}[z - h(x)]). \tag{17}$$

Since $h(x)$ is twice continuously differentiable, $J(x)$ in (17) is twice continuously differentiable and thereby satisfies Lipschitz condition. Taking $n = 2l$, the $J(x)$ meets the assumptions given in Section I.

Note: For an authentic power system state estimation [6], bad data processing has to be associated with the solution of (17); if bad measurements are present, they should be eliminated, and the states re-estimated. However, for the purposes of this paper, we will focus on solving (17) only.

[1] The buses of the power systems considered here are similar to nodes in an electrical network.

[2] The phase angle of the slack bus is considered to be a known value.

B. Conventional Approach

In general, the Newton method may fail to solve (17) because the Hessian matrix of (17) may not be positive definite. Thus, the conventional approach in the power system literature [6] uses a Newton-like method, which is (2) with $C(x^k) = 2H(x^k)^T R^{-1} H(x^k)$ in (3), where $H(x^k) = \nabla h(x^k)$. However, $H(x^k)^T R^{-1} H(x^k)$ may be singular or ill-conditioned, since it is only positive semidefinite. To cope with this difficulty, we can modify the above $C(x^k)$ to ensure the positive definiteness by setting $C(x^k) = 2H(x^k)^T R^{-1} H(x^k) + \delta I$, where δ is a small positive real constant and I is the identity matrix. We see that under this modification, the only assumption needed in Theorem 2.1, $\frac{1}{2}C(x^k) - K_2 I$ is nonnegative definite, is satisfied by taking $K_2 = (\delta/2)$. Furthermore, the numerical stability of our algorithm is guaranteed if δ is not too small. Consequently, (4) becomes

$$[2H(x^k)^T R^{-1} H(x^k) + \delta I]s = -\nabla J(x^k). \tag{18}$$

The matrix $2H(x^k)^T R^{-1} H(x^k) + \delta I$ and the matrix $H(x^k)^T R^{-1} H(x^k)$ are sparse, and (18) can be solved using the sparse matrix technique.

Remark 3.1: a) Because $H(x^k)^T R^{-1} H(x^k)$ may be ill conditioned, methods that use an orthogonal transformation technique to solve $2H(x^k)^T R^{-1} H(x^k)s = -\nabla J(x^k)$ has been developed [7]. However, these methods are computationally very inefficient. b) In [6], the step-size γ^k is set to be 1 for all k. In fact, with a constant step-size such as this there is no guarantee that (2) will converge. c) There are other methods in the power system literature that can obtain an approximate solution for state estimation using a decoupling approach [8]. However, these methods are beyond the scope of this paper, since we are concerned only with methods that solve (1) exactly.

C. Application of the Proposed Method

Let us partition the power system network into $p(>1)$ subnetworks such that each subnetwork is a connected graph in the topological sense. Let B_i denote the set of buses of the ith subnetwork; then s_{B_i} denotes the subvector of s corresponding to states of the ith subnetwork. Replacing s_i, $C(x^k)$, and $\nabla J(x^k)$ in the proposed algorithm by s_{B_i}, $2H(x^k)^T R^{-1} H(x^k) + \delta I$, and $H(x^k)R^{-1}(z - h(x^k))$, respectively, we can apply the proposed algorithm directly to (17). Analogous to (8), we see that $s_{B_1}, \cdots, s_{B_{i-1}}, s_{B_{i+1}}, \cdots, s_{B_p}$ are held fixed in the ith minimization problem

$$\min_{s_{B_i}} \tfrac{1}{2} s^T C(x^k) s + \nabla J(x^k)^T s \tag{19}$$

whose solution can be obtained by solving a small set of linear equations involving only the ith subnetwork and the boundary buses outside the ith subnetwork [9].

D. Test Results

We applied our method, the sparse matrix technique based Newton-like method, and the Newton-like method without sparse matrix technique to the weighted least squares problem in state estimation of the IEEE 30-bus power system and the IEEE 118-bus power system. For the cases under consideration, the IEEE 30-bus system was partitioned into three subnetworks, as shown in Fig. 1, in which each subnetwork is indicated by closed dashed contours. The IEEE 118-bus system is partitioned into eight subnetworks. Because of the limitation on the length of this paper, we can not include here the figure of the IEEE 118-bus system, however, this figure can be found in [10]. Nonetheless, we list the bus number of the buses of each subnetwork in the following. Subnetwork 1 contains buses 1 to 14,

TABLE I
COMPARISON OF OUR METHOD WITH NEWTON-LIKE METHOD WITH AND WITHOUT SPARSE MATRIX TECHNIQUE

System	Final objective value			Average CPU time (seconds)			Speedup ratio		
	NLWTS$^{(\triangle)}$ method	NLWS$^{(\square)}$ method	Our method	NLWTS$^{(\triangle)}$ method (I)	NLWS$^{(\square)}$ method (II)	Our method (III)	$\frac{I}{II}$	$\frac{I}{III}$	$\frac{II}{III}$
IEEE-30 bus	15.88	15.88	15.88	0.99	0.46	0.27	2.15	3.67	1.70
IEEE-118 bus	43.05	43.05	43.05	76.53	5.49	1.59	13.94	48.13	3.45

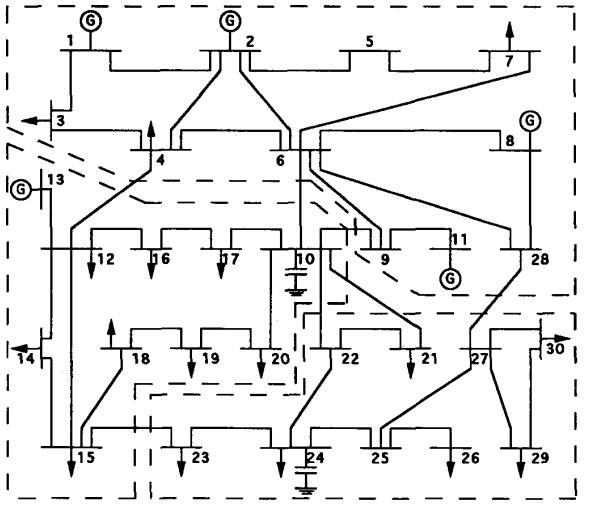

Fig. 1. The IEEE 30-bus system and three partitioned subnetworks.

and bus 117. Subnetwork 2 contains buses 15 to 19, buses 27 to 32, and buses 113 and 114. Subnetwork 3 contains buses 20 to 26, buses 70 to 76, and bus 118. Subnetwork 4 contains buses 33 to 47. Subnetwork 5 contains buses 50 to 61, and buses 63 and 64. Subnetwork 6 contains buses 48, 49, and 62, buses 65 to 69, buses 77 to 81, and buses 97, 98, and 116. Subnetwork 7 contains buses 82 to 96. Subnetwork 8 contains buses 99 to 112. Note that each subnetwork contains about 10 to 20 buses in mesh-interconnected structure though the bus numbers are not consecutive.

In both cases, we took the real-power flow and the reactive-power flow of all transmission lines of each individual system as the measurement data and assumed that some of these are bad data. We set the parameters in our algorithm as follows: $\delta = 0.01$, $\beta = 0.9$ for both systems, $\xi\% = 50\%$ for the IEEE 30-bus system, and $\xi\% = 10\%$ for the IEEE 118-bus system, $\lambda = 1.05, 1.10, 1.15$ for the IEEE 30-bus system, and $\lambda = 1.25, 1.30, 1.35$ for the IEEE 118-bus system. Note that each value of λ corresponds to one computer run. We used various values of λ to test Armijo-type rule and average the CPU time. The accuracy ϵ in the stopping criterion of our algorithm, which is described in Subsection II-C-3), was set to be 10^{-2} for both cases. We used a Sun 4/60 workstation to test our algorithm. The simulation results for the case of the IEEE 30-bus system and the case of the IEEE 118-bus system are shown in Table I. The average CPU time of our algorithm is the average CPU time of the computer runs with various values of λ reported on the Sun 4/60 workstation. We also solved the same cases for both systems using (2), the sparse-matrix technique based Newton-like method, with the same setup of Armijo-type rule and same initial guess as our method on the same Sun 4/60 workstation. We used $|s^k|_\infty < \varepsilon = 10^{-2}$ as the stopping criteria for the sparse matrix technique based Newton-like method, the test results for which are also shown in Table I. We also applied the Newton-like method without using the sparse matrix technique to the same cases on the same workstation with the same setup of Armijo-type rule, same initial guess, and same stopping criteria as the sparse matrix technique based Newton-like method. The resulting average CPU times are also reported in Table I for the cases of both systems. We see that our algorithm achieved the same final objective value as the Newton-like methods. As expected, the sparse matrix technique based Newton-like method was much faster than the Newton-like method without the sparse matrix technique in both cases, especially in the case of the IEEE 118-bus system, which has more sparsity. Compared to the sparse matrix technique based Newton-like method, our method also performs better with the larger system. From the speedup ratio shown in Table I, we see that in the case of the IEEE 118-bus system, our method is 3.45 times faster than the sparse matrix technique based Newton-like method and 48.13 times faster than the Newton-like method without the sparse matrix technique. This demonstrates the dramatic increase in efficiency provided by our method. In order to better appreciate the merits of our algorithm, in Fig. 2 we describe the details of the progression of our algorithm and the Newton-like method with the sparse matrix technique in solving the case of the IEEE 118-bus system when $\lambda = 1.35$. The result of our algorithm is shown by the curve marked with circles ∘ and associated with the Arabic numeral iteration index. Each circle indicates the CPU time accumulated (horizontal axis) at that iteration versus the corresponding value of the objective function (vertical axis) in the progression. The curve marked with asterisks * and associated with the Roman numeral iteration index corresponds to the Newton-like method with the sparse matrix technique. Our algorithm takes 12 iterations to meet the stopping criteria $|\hat{s}^k|_\infty < 10^{-2}$, while the sparse matrix technique based Newton-like method takes only 11 iterations to meet the stopping criteria $|s^k|_\infty < 10^{-2}$. However, we see that the amount of CPU time taken up by each iteration of our algorithm is far less than that for each iteration of the Newton-like method with the sparse matrix technique. This shows the effectiveness and efficiency of using the BGSLS method to generate the descent direction. Furthermore, when our algorithm has nearly reached the optimal objective value, the Newton-like method with the sparse matrix technique has just finished its first iteration and the corresponding objective value is still quite far away from the optimal value. The efficiency of our algorithm is obvious.

IV. CONCLUSION

We have developed a globally convergent algorithm for unconstrained optimization problems of nonlinear large mesh-interconnected systems. Although, due to the nature of our method, an analytical convergence rate is not available, the dramatic increase in efficiency provided by our method can be observed both from the method itself and from the simulation results. It is worth noting that the BGSLS method can be processed by parallel processors if the order of the partitioned subnetworks is suitably arranged [11]. Thus, the speed of our method may be further increased by using parallel processors.

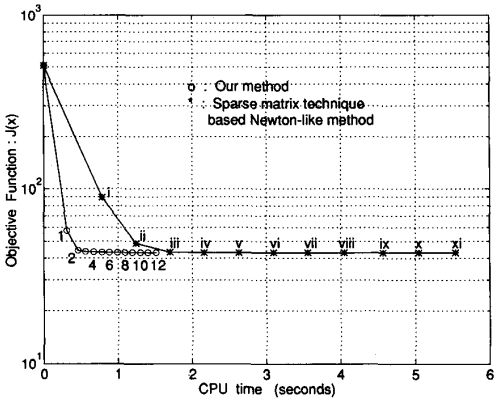

Fig. 2. Details of the progression of our method and the sparse matrix technique based Newton-like method in solving the weighted least squares problem for the IEEE 118-bus system.

APPENDIX

Proof of Lemma 2.3: a) Let $T = \{s \in \Re^{2n} | C(x^k)s + \nabla J(x^k) = 0\}$ denote the solution set of (3). Then if $s \notin T$, the descent property of the BGSLS method follows directly from Lemma 2.2 and (12)–(14). According to the Global Convergence Theorem [12, pp. 187–188], b) can be proven if we show that i) the BGSLS method is a descent method, ii) the sequence generated by the BGSLS method lies in a compact set, and iii) the mapping of any iteration of the BGSLS method is closed. Clearly, i), which is a), has been shown. In the following, we will prove ii) and iii).

Because $C(x^k)$ is positive definite, the unique minimum solution of (3) is

$$s^* = -C(x^k)^{-1}\nabla J(x^k). \tag{A1}$$

Let the set $S \equiv \{s \in \Re^{2n} | -\frac{1}{2}\nabla J(x^k)^T C(x^k)^{-1}\nabla J(x^k) \le \frac{1}{2}s^T C(x^k)s + \nabla J(x^k)^T s \le 0\}$. Then $S \neq \emptyset$, since $0 \in S$. We claim that S is compact. Clearly, S is closed. Thus, it is enough to show that S is bounded. Suppose not, $\exists s \in S \ni$

$$||s||_2 > \max\left(\frac{||\nabla J(x^k)||_2 + 1}{K_2}, \left|\frac{1}{2}\nabla J(x^k)^T C(x^k)^{-1}\nabla J(x^k)\right| + 1\right). \tag{A2}$$

For this s and by the assumption that $\frac{1}{2}C(x^k) - K_2 I > 0$, we have

$$\begin{aligned}|\tfrac{1}{2}s^T C(x^k)s + \nabla J(x^k)^T s| &\ge |\tfrac{1}{2}s^T C(x^k)s| - |\nabla J(x^k)^T s| \\ &\ge K_2||s||_2^2 - ||\nabla J(x^k)||_2||s||_2 \\ &= (K_2||s||_2 - ||\nabla J(x^k)||_2)||s||_2. \end{aligned}\tag{A3}$$

From (A2)

$$\begin{aligned}(K_2||s||_2 - ||\nabla J(x^k)||_2)||s||_2 &> ||s||_2 \\ &> |\tfrac{1}{2}\nabla J(x^k)^T C(x^k)^{-1}\nabla J(x^k)|. \end{aligned}\tag{A4}$$

Thus, from (A3) and (A4), we obtain that $|\frac{1}{2}s^T C(x^k)s + \nabla J(x^k)^T s| > \frac{1}{2}|\nabla J(x^k)^T C(x^k)^{-1}\nabla J(x^k)|$. This inequality contradicts $s \in S$. Hence S must be bounded. From (a), the BGSLS method is a descent method. Thus, every point in the sequence $\{s(t)\}$ generated by the BGSLS method starting from $s = 0$ should satisfy

$$\tfrac{1}{2}s(t)^T C(x^k)s(t) + \nabla J(x^k)s(t) \le 0, \quad \forall t \tag{A5}$$

moreover, by the descent property

$$\begin{aligned}&\tfrac{1}{2}s^{*T}C(x^k)s^* + \nabla J(x^k)^T s^* \\ &\quad \le \tfrac{1}{2}s(t)^T C(x^k)s(t) + \nabla J(x^k)^T s(t), \quad \forall t. \end{aligned}\tag{A6}$$

From (A1) the left-hand side of (A6) equals $-\frac{1}{2}\nabla J(x^k)^T C(x^k)^{-1}\nabla J(x^k)$, thus from (A5) and (A6), the sequence $\{s(t)\}$, $\forall t$, must lie inside the compact set S. This proves ii). Next, we will prove iii). Based on the fact that the composition of closed mappings is a closed mapping, and the exact line search method is also a closed mapping [12, p. 210], it is enough to show iii) if we can show that the mapping of Step 3 of our algorithm is closed. This is true because (8) is a bounded, unconstrained quadratic minimization problem with positive definite $C(x^k)$. This completes the proof. □

Proof of Corollary 2.1: Since by Lemma 2.3, the sequence $\{s(t)\}$ converges to a point s^* satisfying $C(x^k)s^* + \nabla J(x^k) = 0$, then $||C(x^k)||_2||s^*||_2 \ge ||\nabla J(x^k)||_2$. Let $K_0 = (1/sup_k||C(x^k)||_2)$. We have $||s^*||_2 \ge K_0||\nabla J(x^k)||_2$. Let $K_1 = (K_0/2) > 0$ and let $\tau = K_1||\nabla J(x^k)||_2$. Then $\tau > 0$ since we only need to consider the case $||\nabla J(x^k)||_2 \neq 0$. Because $\{s(t)\}$ converges to s^*, there exists a t' such that $||s^* - s(t)||_2 < \tau$ for all $t > t'$. Consequently, $||s(t)||_2 > ||s^*||_2 - \tau \ge K_0||\nabla J(x^k)||_2 - K_1||\nabla J(x^k)||_2 = K_1||\nabla J(x^k)||_2$ for all $t > t'$ □

Proof of Theorem 2.1: From Corollary 2.1, we see that $\forall k$, with $\nabla J(x^k) \neq 0$, there must exist a t' such that the stopping criteria (16)

$$||s(t)||_2 > K_1||\nabla J(x^k)||_2, \quad \text{for some } K_1 > 0, \quad \forall t > t' \tag{A7}$$

is satisfied. Furthermore, by assumption, we have $\frac{1}{2}s(t)^T C(x^k)s(t) - s(t)^T K_2 s(t) \ge 0$, $\forall t$; and from (A5), we have $-\frac{1}{2}s(t)^T C(x^k)s(t) - \nabla J(x^k)^T s(t) \ge 0$, $\forall t$; these two inequalities lead to

$$-K_2||s(t)||_2^2 \ge \nabla J(x^k)^T s(t), \quad \forall t. \tag{A8}$$

Then, the following proof mostly follows the proof of the convergence theorem for descent algorithms given in [1, pp. 203–204]. A Descent Lemma given in [1, pp. 203–204] states that if $J(\cdot)$ is continuously differentiable and there exists a positive constant K such that $\|\nabla J(x) - \nabla J(y)\| \le K\|x - y\|_2$, $\forall x, y \in \Re^n$, then $J(x + y) \le J(x) + \nabla J(x^k)^T y + (K/2)\|y\|_2^2$, $\forall x, y \in \Re^n$. By the assumptions on $J(\cdot)$ given in Section I, we may use the Descent Lemma and (A8) to obtain

$$\begin{aligned}J(x^k + \gamma^k \hat{s}^k) &\le J(x^k) + \gamma^k \nabla J(x^k)^T \hat{s}^k + \frac{K}{2}\gamma^{k2}||\hat{s}^k||_2^2 \\ &\le J(x^k) - \gamma^k\left(K_2 - \frac{K\gamma^k}{2}\right)||\hat{s}^k||_2^2. \end{aligned}\tag{A9}$$

If $0 < \gamma^k < (K_2/K)$, we can derive that

$$-\gamma^k\left(K_2 - \frac{K\gamma^k}{2}\right) \le -\frac{K_2}{2}\gamma^k. \tag{A10}$$

By (A9) and (A10), we have

$$J(x^k + \gamma^k \hat{s}^k) \le J(x^k) - \frac{K_2}{2}\gamma^k||\hat{s}^k||_2^2. \tag{A11}$$

If $\lambda \ge (K_2/K)$, $0 < \beta < 1$, since $\{\beta^m\lambda\}$, $m = 0, 1, \cdots$ is a monotonic decreasing sequence approaching 0, there must exist an m such that $\beta^m\lambda < (K_2/K)$. Let m_k be the smallest nonnegative integer satisfying the above inequality, then the following holds

$$\beta\frac{K_2}{K} \le \beta^{m_k}\lambda < \frac{K_2}{K}. \tag{A12}$$

If $0 < \lambda < (K_2/K)$, $0 < \beta < 1$, then

$$\beta\lambda < \lambda < \frac{K_2}{K}. \tag{A13}$$

In this case, we can view $m_k = 0$ which is the smallest nonnegative integer for $\beta^{m_k}\lambda < (K_2/K)$. Now, for any $\lambda > 0$, $0 < \beta < 1$, and let m_k be either the m_k determined by (A12) or $m_k = 0$ by (A13); from (A11), there must exist an $m' \leq m_k$ (or $m' = 0$ if $m_k = 0$) such that

$$J(x^k + \beta^{m'}\lambda\hat{s}^k) \leq J(x^k) - \frac{K_2}{2}\beta^{m'}\lambda\|\hat{s}^k\|_2. \tag{A14}$$

Note that $0 < \gamma^k < (K_2/K)$ is sufficient for (A11) to hold explains why $m' \leq m_k$. This shows that Step 8 of our algorithm will terminate for certain m'. Since $\beta^{m'} \geq \beta^{m_k}$, from (A12) and (A13), $\beta^{m'}\lambda \geq \min\{\beta\lambda, \beta(K_2/K)\}$. Let $\tau \equiv \frac{1}{2}K_2 \cdot \min\{\beta\lambda, \beta(K_2/K)\}$, then τ is finite and positive, and

$$J(x^k + \beta^{m'}\lambda\hat{s}^k) \leq J(x^k) - \tau\|\hat{s}^k\|_2^2. \tag{A15}$$

Then, each iteration of our algorithm ensures a decrement of the objective function by at least the amount $\tau\|\hat{s}^k\|_2^2$. Since $J(x)$ is bounded from below, we assume $c \in \Re$ is a lower bound of $J(x)$, then from (A15), we have

$$0 \leq J(x^{k+1}) - c \leq J(x^k) - c - \tau\|\hat{s}^k\|_2^2, \quad \forall k. \tag{A16}$$

Then, by (A16), $\sum_{k=0}^{\infty}\|\hat{s}^k\|_2^2 \leq (J(x^0) - c/\tau) < \infty$, and (A7) shows that $\lim_{k\to\infty}\nabla J(x^k) = 0$. □

REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. London: Prentice-Hall, 1989.

[2] L. O. Chua and P. M. Lin, *Computer Aided Analysis of Electronic Circuits*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[3] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proc. IEEE*, vol. 55, no. 11, pp. 1801–1809, Nov. 1967.

[4] C.-H. Lin, "A new parallel processing algorithm for power system state estimation problems," master's thesis, Dept. of Elec. Engr., Nat'l Tsing Hua Univ., Hsinchu, Taiwan, 1991 (in Chinese).

[5] F. C. Schweppe and J. Wildes, "Power system static-state estimation, part I: exact model," *IEEE Trans. Power App. Syst.*, vol. PAS-89, no. 1, pp. 120–125, Jan. 1970.

[6] F. C. Schweppe, " Power system static-state estimation, part III: implementation," *IEEE Trans. Power App. Syst.*, vol. PAS-89, no. 1, pp. 130–135, Jan. 1970.

[7] A. Simoes-Costa and V. H. Quintana, "A robust numerical technique for power system state estimation," *IEEE Trans. Power App. Syst.*, vol. PAS-100, pp. 691–698, Feb. 1981.

[8] A. Garcia, A. Monticelli, and P. Abreu, "Fast decoupled state estimation and bad data processing," *IEEE Trans. Power App. Syst.*, vol. PAS-98, pp. 1645–1652, Sep. 1979.

[9] S.-Y. Lin, C.-H. Lin, and S.-L. Yu, "An efficient descent algorithm for a class of unconstrained optimization problems of nonlinear large mesh-interconnected systems," in *Proc. 32nd IEEE Conf. Dec. & Contr.*, Dec. 1993.

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

[11] S.-Y. Lin, "A parallel processing multi-coordinate descent method with line search – algorithm and convergence," in *Proc. 30th IEEE Conf. Deci. Contr.*, Dec. 1991, pp. 2096–2097.

[12] D. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.

On the Possible Divergence of the Projection Algorithm

Erjen Lefeber and Jan Willem Polderman

***Abstract*— It is shown by means of an example that the projection algorithm does not always converge.**

I. INTRODUCTION

It is well known that parameter identification of linear systems depends very much on the excitation of the signals. Generally speaking, all identification algorithms require the signals to be sufficiently exciting. In applications such as adaptive control, however, excitation is often not possible. The question then arises how useful the standard identification schemes are. In this note we consider the case where the data can be modeled exactly by a linear time invariant discrete-time model. It is a fact, that for such systems recursive least squares always produce a convergent sequence of parameter estimates, although it is of course not guaranteed that the limit will be the true parameter [1].

For the projection algorithm a similar result or its negation is to the best of our knowledge not available in the literature. Properties that can be derived without any assumptions on the signals can be found in [1]. Nothing is said about convergence there (see also [2, Problem 12.14]). In [3], the algorithm is used for adaptive pole assignment. Since the adaptive algorithm could be analyzed without proving convergence of the parameter estimates, the possible convergence is not studied there either.

In this note we show by means of an example that the projection algorithm does not necessarily converge. This is in contrast with recursive least squares.

The construction of the counter example is as follows. Firstly we construct a sequence of real vectors that satisfies at least some of the properties of the projection algorithm and which does not converge. Secondly we show that the sequence could as well have been obtained by applying the projection algorithm to an appropriate input/output system. Hence, rather than fitting the estimates to the data, we fit the data to the estimates.

II. THE PROJECTION ALGORITHM

For the sake of completeness, we briefly describe the projection algorithm. Let the system be described by

$$y(k+1) = \bar{\theta}^T\phi(k) \quad \bar{\theta} \in \mathbb{R}^n. \tag{1}$$

The projection algorithm is defined as follows: Suppose that the estimate of $\bar{\theta}$ at time k is θ_k, define $G_{k+1} := \{\theta \in \mathbb{R}^n \mid y(k+1) = \theta^T\phi(k)\}$. Define θ_{k+1} as the orthogonal projection of θ_k on G_{k+1}. The recursion is given by

$$\theta_{k+1} = \theta_k + \frac{\phi(k)}{\|\phi(k)\|^2}(y(k+1) - \theta_k^T\phi(k)). \tag{2}$$

Notice that G_{k+1} contains the true parameter $\bar{\theta}$. Regardless of the input sequence, the following two properties hold.

Property 2.1: 1) For all k: $\|\bar{\theta} - \theta_{k+1}\| \leq \|\bar{\theta} - \theta_k\|$.

2) $\lim_{k\to\infty}(\theta_{k+1} - \theta_k) = 0$.

It is obvious that from Property 2.1 we cannot conclude that θ_k is a fundamental sequence, and in fact we will see that it need not be.

Manuscript received November 10, 1993.

The authors are with the Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.

IEEE Log Number 9407235.