Mathematical Programming 43 (1989) 187-207
North-Holland

SUCCESSIVE COLUMN CORRECTION ALGORITHMS FOR SOLVING SPARSE NONLINEAR SYSTEMS OF EQUATIONS

Guangye LI

Computer Center, Jilin University, Changchun, Jilin, People's Republic of China

Received 15 August 1986
Revised manuscript received 25 January 1988

This paper presents two algorithms for solving sparse nonlinear systems of equations: the CM-successive column correction algorithm and a modified CM-successive column correction algorithm. A q-superlinear convergence theorem and an r-convergence order estimate are given for both algorithms. Some numerical results and the detailed comparisons with some previously established algorithms show that the new algorithms have some promise of being very effective in practice.

Key words: Finite difference, Jacobian, nonlinear equation, sparsity.

1. Introduction

Consider a nonlinear system of equations

$$F(x) = 0, \tag{1.1}$$

where $F: R^n \to R^n$ is continuously differentiable on an open convex set $D \subset R^n$, and the Jacobian matrix $F'(x)$ is sparse. To solve the system, the following iteration is considered:

$$x^{k+1} = x^k - B_k^{-1} F(x^k), \quad k = 0, 1, \ldots, \tag{1.2}$$

where B_k is an approximation to the Jacobian with the same sparsity structure.

For convenience, we rewrite (1.2) as

$$\bar{x} = x - B^{-1} F(x), \tag{1.3}$$

where x and $\bar{x}$ indicate the current iterate and the new iterate respectively, and B is an approximation to the Jacobian $F'(x)$.

Currently, there are several algorithms to get a sparse approximation to the Jacobian. In this paper we will discuss three types of algorithms.

This research was partially supported by contracts and grants: DOE DE-AS05-82ER1-13016, AFOSR 85-0243 at Rice University, Houston, U.S.A. and Natural Sciences and Engineering Research Council of Canada grant A-8639.

(1) *The Sparse Broyden algorithm.* In 1970 Schubert [18] gave a sparse modification of Broyden's update. Broyden [2] also gave this algorithm independently. It is called the SB algorithm. In order to present the SB algorithm, we introduce the following notation concerning the sparsity pattern of the Jacobian:

Definition 1.1. For $j = 1, 2, \ldots,$ define the subspace $Z_j \subset R^n$ determined by the sparsity pattern of the jth row of the Jacobian:

$$Z_j \equiv \{v \in R^n: e_i^T v = 0 \text{ for all } i \text{ such that } [F'(x)]_{ji} = 0 \text{ for all } x \in R^n\},$$

where e_i is the ith column of the $n \times n$ identity matrix. Define the set of matrices Z that preserve the sparsity pattern of the Jacobian:

$$Z \equiv \{A \in L(R^n): A^T e_j \in Z_j \text{ for } j = 1, 2, \ldots, n\}.$$

Definition 1.2. For $j = 1, 2, \ldots, n,$ define the projection operator, $D_j \in L(R^n)$, that maps R^n onto Z_j:

$$D_j \equiv \operatorname{diag}(d_{j1}, d_{j2}, \ldots, d_{jn}),$$

where

$$d_{ji} = \begin{cases} 1 & \text{if } e_i \in Z_j, \\ 0 & \text{otherwise.} \end{cases}$$

For a scalar $\alpha \in R$, define the pseudo-inverse:

$$\alpha^+ = \begin{cases} \alpha^{-1} & \text{if } \alpha \neq 0 \\ 0 & \text{if } \alpha = 0. \end{cases}$$

Now the SB update can be written as

$$\bar{B} = B + \sum_{j=1}^{n} ([s]_j^T [s]_j)^+ e_j e_j^T (y - Bs)[s]_j^T, \tag{1.4}$$

where $[s]_j = D_j s$, $s = \bar{x} - x$ and $y = F(\bar{x}) - F(x)$.

Let

$$Q_{u,v} = \{A \in L(R^n): Au = v, \text{ for vectors } u, v \in R^n\}.$$

The following theorem, which we will use later, was proved by Reid [16] and Marwil [10] independently.

Theorem 1.1. *Let $B \in Z$; $y, s \in R^n$ with $s \neq 0$. Define $\bar{B}$ by* (1.4). *Then $\bar{B}$ is the unique solution to*

$$\min\{\|\hat{B} - B\|_F: \hat{B} \in Q_{y,s} \cap Z\}, \tag{1.5}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix.

The advantage of the SB algorithm is that at each iteration only one function value is required and it is q-superlinearly convergent (see Marwil [10]). However, it frequently requires more iterations than finite difference algorithms. Moreover, the matrix B, generated by the SB algorithm may not be a good approximation to the Jacobian when the problem is badly nonlinear, especially when the current step is far away from the solution. Therefore, $p_k = -B_k^{-1}F(x^k)$ may not be a good descent direction of the functional $f(x) = \frac{1}{2}\|F(x)\|^2$, where $\|\cdot\|$ denotes the l_2 vector norm. In this case, it may be not good to use a line search with Schubert's algorithm.

(2) *Finite difference algorithms.* In general, a finite difference algorithm can be formulated as follows: obtain direction vectors $d_1, d_2, \ldots, d_p$ such that B can be determined uniquely by the equations

$$Bd_i = F(x+d_i) - F(x), \quad i = 1, 2, \ldots, p.$$

In this paper, we assume that it is not convenient to evaluate the function values element by element, instead we only evaluate the value of $F(x)$ as a single entity. This is reasonable since in practice it is very common that the components of $F(x)$ have expensive common sub-expressions. In this case, to reduce the number of function evaluations, Curtis, Powell and Reid [5] proposed a finite difference algorithm, called the CPR algorithm, which is based on a partition of the columns of the Jacobian. Coleman and Moré [4] associate the partition problem with a graph coloring problem and gave some partitioning algorithms which can make the number of the function evaluations optimal or nearly optimal.

Following Coleman and Moré, we give some definitions concerning a partition of the columns of the Jacobian.

Definition 1.3. A partition of the columns of a matrix B is a division of the columns into groups $c_1, c_2, \ldots, c_p$ such that each column belongs to one and only one group.

Definition 1.4. A partition of the columns of a matrix B is consistent with the direct determination of B if whenever b_{ij} is a nonzero element of B, then the group containing column j has no other column with a nonzero element in row i.

As an example we consider the tridiagonal structure

$$\begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 \\ 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}. \tag{1.6}$$

A consistent partition of the columns of the matrix is $c_1 = \{1, 4\}$, $c_2 = \{2, 5\}$, and $c_3 = \{3, 6\}$.

The CPR algorithm now can be formulated as follows: for a given consistent partition of the columns of the Jacobian, obtain vectors $d_1, d_2, \ldots, d_p$ such that B is determined uniquely by the equations

$$Bd_i = F(x+d_i) - F(x) \equiv y_i, \quad i = 1, 2, \ldots, p. \tag{1.7}$$

Notice that for the CPR algorithm, the number of function evaluations at each iteration is $p+1$. Since the partition of the columns of the Jacobian plays an important role in the CPR algorithm, we call the CPR algorithm based on Coleman and Moré's algorithms the CPR-CM algorithm.

For the consistent partition given in example (1.6), if we take

$$d_1 = (h, 0, 0, h, 0, 0)^T,$$

$$d_2 = (0, h, 0, 0, h, 0)^T,$$

$$d_3 = (0, 0, h, 0, 0, h)^T,$$

then B is determined uniquely and the number of function evaluations required at each iteration is four.

The advantage of the CPR algorithm is that it usually generates better approximation matrices and therefore requires fewer iterations than the SB algorithm. However, it requires more function values at each iteration than the SB algorithm.

(3) *The successive column correction algorithms.* Polak [14] gave a successive column correction algorithm for unconstrained minimization. Feng and Li [8] developed a successive column correction algorithm for nonlinear system of equations, which is called the column-update quasi-Newton method. Using this algorithm, columns of B_k are displaced by differences successively and periodically. At each iteration, only two function values are required, but only one column is displaced.

In this paper, we propose two algorithms: the CM-successive column correction algorithm and the modified CM-successive column correction algorithm. The former is based on Coleman and Moré's algorithm and the column-update algorithm. The latter is a combination of the CM-successive column correction algorithm and the SB algorithm. Both algorithms require only two function values at each iterative step. Our numerical results show that the CM-successive column correction algorithms, especially the modified one, are competitive with the CPR algorithm and the SB algorithm.

The CM-successive column correction algorithm is given in Section 2. A Kantorovich-type analysis for this algorithm is given in Section 3. A q-superlinear convergence result and an r-convergence order estimate of the CM-successive column correction algorithm are given in Section 4. The modified CM-successive column correction algorithm is given in Section 5. Some numerical results and some detailed comparisons of the new algorithms with the CPR and the SB algorithms for the test problems are given in Section 6.

In this paper, $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, and $\|\cdot\|$ denotes the l_2-vector norm. For a sparse matrix B, we use M to denote the set of pairs of indices (i, j), where b_{ij} is a structurally nonzero element of B, i.e.

$$M = \{(i, j): b_{ij} \neq 0\}.$$

Moreover, we use $S(y, \delta)$ to denote the set $\{x \in R^n: \|x - y\| < \delta, y \in R^n\}$ and use $\bar{S}(y, \delta)$ to denote the closure of $S(y, \delta)$.

2. The CM-successive column correction algorithm and its properties

Given a consistent partition of the columns of the Jacobian, which divides the set $\{1, 2, \ldots, n\}$ into p subsets $c_1, c_2, \ldots, c_p$, let

$$d^k = \sum_{j \in c_{i_k}} h^k e_j, \tag{2.1}$$

where

$$i_k = k \pmod p, \quad k = 1, 2, \ldots,$$

and let

$$y^k = F(x^k + d^k) - F(x^k). \tag{2.2}$$

The CM-successive column correction algorithm can be formulated as follows: If $k \leq p$, then for $j \in c_k$, the jth column of B_k is determined uniquely by the equation

$$B_k d^k = y^k, \tag{2.3}$$

and the other columns of B_k are equal to the corresponding columns of B_{k-1}. If $k > p$, the columns of B_k are displaced as described above successively and periodically. In other words, for $j \in c_{i_k}$, the jth column of B_k is determined uniquely by (2.3), and the other columns of B_k are equal to the corresponding columns of B_{k-1}.

For example (1.6), at the first iteration we displace the first group $c_1 = \{1, 4\}$. At the second iteration we displace the second group $c_2 = \{2, 4\}$. At the third iteration we displace the third group $c_3 = \{3, 6\}$, and then we displace the three groups successively and periodically.

Note that one does not have to choose a uniform step length h^k for all components of x^k. In practice, it is preferred to choose different step length for different component of x^k as we do in the numerical examples. In fact, uniform step lengths will not affect our theoretical results. However, for simplicity, we assume a uniform step length for each step.

The CM-successive column correction algorithm with a global strategy is given below.

Algorithm 2.1. Given a consistent partition of the columns of the Jacobian, which divides the set $\{1, 2, \ldots, n\}$ into p subsets $c_1, c_2, \ldots, c_p$ (for convenience, c_i, $i = 1, 2, \ldots, p$, indicates both the sets of the columns and the sets of the indices of these

columns), and given an $x^0 \in R^n$ and a nonsingular matrix B_0, which has the same sparsity as the Jacobian, at the initial step:

(1) Set $l = 0$.

(2) Solve $B_0 s^0 = -F(x^0)$.

(3) Choose x^1 by $x^1 = x^0 + s^0$, or by a global strategy.

At each iteration $k > 0$,

(1) Choose a scalar h^k.

(2) If $l < p$, then set $l = l + 1$, otherwise set $l = 1$.

(3) Set

$$d^k = \sum_{j \in c_l} h^k e_j.$$

(4) If $j \in c_l$ and $(i, j) \in M$, then set

$$b_{ij}^k = \frac{1}{h^k} e_i^{\mathrm{T}}(F(x^k + d^k) - F(x^k)), \tag{2.4}$$

otherwise set

$$b_{ij}^k = b_{ij}^{k-1},$$

where $B_k = [b_{ij}^k]$.

(5) Solve $B_k s^k = -F(x^k)$.

(6) Choose x^{k+1} by $x^{k+1} = x^k + s^k$, or by a global strategy.

(7) Check for convergence.

The global strategy mentioned in (3) and (6) is for the global convergence of this algorithm. One may use a line search strategy or a trust region strategy. We used a line search strategy for our numerical examples.

Let

$$J_k = \int_0^1 F'(x^k + t d^k)\, \mathrm{d}t. \tag{2.5}$$

Then

$$J_k d^k = y^k. \tag{2.6}$$

Let $J_k = [J_{lm}^k]$. Since J_k has the same sparsity as the Jacobian, by (2.6), we have that if $(l, m) \in M$, then

$$J_{lm}^k = \frac{e_l^{\mathrm{T}} y^k}{h^k}, \tag{2.7}$$

where $m \in c_{i_k}$. Comparing (2.7) with (2.4), we have

$$B_k e_j = J_k e_j,$$

for $j \in c_{i_k}$.

The CM-successive column correction algorithm is also an update algorithm, and the update can be written as:

$$B_k = B_{k-1}\left(I - \sum_{j \in c_{i_k}} e_j e_j^{\mathrm{T}}\right) + \sum_{j \in c_{i_k}} J_k e_j e_j^{\mathrm{T}}. \tag{2.8}$$

From (2.8), it is easy to get the following result:

Lemma 2.2. Let B_k, $k = 1, 2, \ldots$, *be generated by Algorithm* 2.1. *If* $k \geq p$, *then*

$$B_k = \sum_{j=k-p+1}^{k} \sum_{l \in c_{i_j}} J_j e_l e_l^{\mathrm{T}}. \tag{2.9}$$

To study the properties of our algorithms, sometimes we assume that F' satisfies the following Lipschitz condition: there exist $\alpha_i > 0$, $i = 1, 2, \ldots, n$ such that

$$\|(F'(x) - F'(y))e_i\| \leq \alpha_i \|x - y\|, \quad x, y \in D. \tag{2.10}$$

Let $\alpha = (\sum_{i=1}^{n} \alpha_i^2)^{1/2}$. Then, it follows from (2.10) that

$$\|F'(x) - F'(y)\|_F \leq \alpha \|x - y\|, \quad x, y \in D. \tag{2.11}$$

Theorem 2.3. *Let* F' *satisfy Lipschitz condition* (2.10). *Also let* $\{x^j\}_{j=0}^{k} \subset D$ *and let* $\{B_j\}_{j=0}^{k}$ *be generated by Algorithm* 2.1 *with*

$$|h^k| \leq \frac{2}{\sqrt{n}} \|x^k - x^{k-1}\|.$$

If $\{x^j + d^j\}_{j=1}^{k} \subset D$, *then for* $k \geq p$,

$$\|B_k - F'(x^k)\|_F \leq \alpha \sum_{j=k-p+1}^{k} \|x^j - x^{j-1}\|. \tag{2.12}$$

Proof. By (2.5), (2.1) and Lipschitz condition (2.10),

$$\begin{aligned}
\|(F'(x^m) - J_m)e_j\| &= \left\| \left(\int_0^1 (F'(x^m + td^m) - F'(x^m)) \, \mathrm{d}t \right) e_j \right\| \\
&\leq \alpha_j \int_0^1 \|d^m\| t \, \mathrm{d}t = \frac{\alpha_j}{2} \|d^m\| \\
&= \frac{\alpha_j}{2} \left\| \sum_{j \in c_{i_m}} h^m e_j \right\| \\
&\leq \frac{\alpha_j}{2} \sqrt{n} |h^m| \leq \alpha_j \|x^m - x^{m-1}\|,
\end{aligned} \tag{2.13}$$

where $k-p+1 \leq m \leq k$. It follows from (2.9) and (2.13) that

$$\begin{aligned}
\|F'(x^k)-B_k\|_F^2 &= \sum_{m=k-p+1}^{k} \left\| \sum_{j\in c_{i_m}} (F'(x^k)-B_k)e_j e_j^T \right\|_F^2 \\
&= \sum_{m=k-p+1}^{k} \sum_{j\in c_{i_m}} \|(F'(x^k)-J_m)e_j\|^2 \\
&\leq \sum_{m=k-p+1}^{k} \sum_{j\in c_{i_m}} (\|(F'(x^k)-F'(x^m))e_j\| \\
&\quad + \|(F'(x^m)-J_m)e_j\|)^2 \\
&\leq \sum_{m=k-p+1}^{k} \sum_{j\in c_{i_m}} \alpha_j^2(\|x^k-x^m\|+\|x^m-x^{m-1}\|)^2 \\
&\leq \sum_{m=k-p+1}^{k} \sum_{j\in c_{i_m}} \alpha_j^2\left(\sum_{l=k-p+1}^{k} \|x^l-x^{l-1}\|\right)^2 \\
&= \alpha^2\left(\sum_{l=k-p+1}^{k} \|x^l-x^{l-1}\|\right)^2. \qquad (2.14)
\end{aligned}$$

Then, (2.12) follows from (2.14). □

To start iteration (1.2) for a given $x^0 \in D$, an initial matrix B_0 is needed. We suggest using the CPR-CM algorithm to get B_0 since it is easy to be implemented after we have a consistent partition of the columns of the Jacobian.

3. A Kantorovich-type analysis

By means of Theorem 2.3, we have the following Kantorovich-type analysis for the CM-successive column correction algorithm.

Theorem 3.1. *Assume that $F'(x)$ satisfies Lipschitz condition* (2.10). *Let $x^0 \in D$, and let B_0 be a nonsingular $n \times n$ matrix such that*

$$\|B_0-F'(x^0)\|_F \leq \delta, \quad \|B_0^{-1}\|_F \leq \beta, \quad \|B_0^{-1}F(x^0)\| \leq \eta,$$

$$h = \frac{\alpha\beta\eta}{(1-3\beta\delta)^2} \leq \tfrac{1}{6}, \qquad (3.1)$$

and

$$\beta\delta < \tfrac{1}{3},$$

where α, β and η are positive scalars. If $\bar{S}(x^0, 2t^*) \subset D$, where

$$t^* = \frac{1-3\beta\delta}{3\alpha\beta}(1-\sqrt{1-6h}), \qquad (3.2)$$

then $\{x^k\}$, *generated by the CM-successive column correction algorithm with* $|h^k| \leq (1/\sqrt{n})\|x^k - x^{k-1}\|$ *and without any global strategy, converges to* x^*, *which is the unique root of* $F(x)$ *in* $\bar{S}(x^0, \bar{t}) \cap D$, *where*

$$\bar{t} = \frac{1-\beta\delta}{\alpha\beta}\left(1+\left(1-\frac{2\alpha\beta\eta}{(1-\beta\delta)^2}\right)^{1/2}\right).$$

Proof. Consider the scalar iteration

$$t_{k+1} - t_k = \beta f(t_k), \quad t_0 = 0, \quad k = 0, 1, 2, \ldots, \tag{3.3}$$

where

$$f(t) = \tfrac{3}{2}\alpha t^2 - \left(\frac{1-3\beta\delta}{\beta}\right)t + \frac{\eta}{\beta}. \tag{3.4}$$

It follows from (3.3) that

$$f(t_{k-1}) = \frac{1}{\beta}(t_k - t_{k-1}).$$

Thus, by Taylor expansion,

$$\begin{aligned} f(t_k) &= f(t_{k-1}) + f'(t_{k-1})(t_k - t_{k-1}) + \tfrac{1}{2}f''(t_{k-1})(t_k - t_{k-1})^2 \\ &= 3\left[\frac{\alpha}{2}(t_k - t_{k-1}) + \alpha t_{k-1} + \alpha\delta\right](t_k - t_{k-1}). \end{aligned} \tag{3.5}$$

Substituting (3.5) into (3.3), we have

$$t_{k+1} - t_k = 3\beta\left[\frac{\alpha}{2}(t_k - t_{k-1}) + \alpha t_{k-1} + \delta\right](t_k - t_{k-1}), \quad k = 1, 2, \ldots. \tag{3.6}$$

Equation (3.6) can be rewritten as

$$t_{k+1} - t_k = 3\beta\left[\frac{\alpha}{2}(t_k + t_{k-1}) + \delta\right](t_k - t_{k-1}). \tag{3.7}$$

Noticing that $t_0 = 0$ and $t_1 = \eta > 0$, by induction, we have that $\{t_k\}$ is a strictly increasing sequence. Since t^* is the smallest root of $f(t)$,

$$\begin{aligned} t^* - t_{k+1} &= t^* - t_k - \beta f(t_k) \\ &= \beta\left\{[f(t^*) - f(t_k) - f'(t_k)(t^* - t_k)] + \left[f'(t_k) + \frac{1}{\beta}\right](t^* - t_k)\right\} \\ &= \beta\{\tfrac{3}{2}\alpha(t^* - t_k) + 3\alpha t_k + 3\delta\}(t^* - t_k) \\ &= \beta\{\tfrac{3}{2}\alpha(t^* + t_k) + 3\delta\}(t^* - t_k). \end{aligned}$$

Therefore, noticing that $t_0 = 0$ and $t^* > 0$, by induction, we obtain

$$t_k \leq t^*, \quad k = 0, 1, \ldots.$$

Hence, there exists a $\bar{t} \leq t^*$ such that

$$\lim_{k\to\infty} t_k = \bar{t}.$$

By (3.3), $f(\bar{t}) = 0$. Since t^* is the smallest root of f, we have that $\bar{t} = t^*$, i.e. we have

$$\lim_{k\to\infty} t_k = t^*.$$

Now, by induction, we will prove that

$$\|x^{k+1} - x^k\| \leq t_{k+1} - t_k, \quad k = 1, 2, \ldots, \tag{3.8}$$

$$\{x^k\} \subset \bar{S}(x^0, t^*), \quad k = 1, 2, \ldots, \tag{3.9}$$

$$\{x^k + d^k\} \subset \bar{S}(x^0, 2t^*), \tag{3.10}$$

and

$$\|B_k^{-1}\| \leq 3\beta, \quad k = 1, 2, \ldots. \tag{3.11}$$

For $k = 0$, we have

$$\|x^1 - x^0\| \leq \eta = t_1 - t_0 \leq t^*.$$

Thus,

$$\|x^1 + d^1 - x^0\| \leq \|x^1 - x^0\| + \|d^1\| \leq 2\|x^1 - x^0\| \leq 2t^*.$$

Suppose (3.8)-(3.11) hold for $k = 0, 1, \ldots, m-1$. Then

$$\|x^m - x^0\| \leq \sum_{i=0}^{m-1} \|x^{i+1} - x^i\| \leq \sum_{i=0}^{m-1} (t_{i+1} - t_i) = t_m \leq t^*.$$

Therefore, $x^m \in \bar{S}(x^0, t^*)$. Furthermore,

$$\begin{aligned}\|x^m + d^m - x^0\| &\leq \|x^m - x^0\| + \|x^m - x^{m-1}\| \\ &\leq 2(t_m - t_0) = 2t_m \leq 2t^*,\end{aligned}$$

which implies

$$\{x^m + d^m\} \subset \bar{S}(x^0, 2t^*).$$

From the proof of Theorem 2.3, it can be seen that for all k,

$$\|B_k - F'(x^k)\|_F \leq \|B_0 - F'(x^0)\|_F + \alpha \sum_{j=0}^{k} \|x^j - x^{j-1}\|. \tag{3.12}$$

Therefore,

$$\begin{aligned}\|B_0^{-1}(B_m - B_0)\| &\leq \|B_0^{-1}\|_F(\|B_m - F'(x^m)\|_F + \|F'(x^m) - F'(x^0)\|_F \\ &\quad + \|F'(x^0) - B_0\|_F) \\ &\leq \beta\left(2\alpha \sum_{i=0}^{m-1} \|x^{i+1} - x^i\| + 2\delta\right) \\ &\leq 2\beta(\alpha t_m + \delta) \leq 2\beta(\alpha t^* + \delta).\end{aligned} \tag{3.13}$$

By (3.2),

$$\alpha t^* + \delta \leq \frac{1}{3\beta}.$$

Hence, by (3.13),

$$\| B_0^{-1}(B_m - B_0) \| \leq \tfrac{2}{3}.$$

Thus, by Ortega and Rheinboldt's Perturbation Lemma [13, p. 45],

$$\| B_m^{-1} \| \leq \frac{\beta}{1-\frac{2}{3}} = 3\beta.$$

Since

$$F(x^{m-1}) + B_{m-1}(x^m - x^{m-1}) = 0,$$

by (3.12), (3.6) and Lemma 4.1.12 in [7],

$$\begin{aligned}
\| x^{m+1} - x^m \| &\leq \| B_m^{-1} \|_F \| F(x^m) \| \\
&= \| B_m^{-1} \|_F \| F(x^m) - F(x^{m-1}) - B_{m-1}(x^m - x^{m-1}) \| \\
&\leq \| B_m^{-1} \|_F \{ \| F(x^m) - F(x^{m-1}) - F'(x^{m-1})(x^m - x^{m-1}) \| \\
&\quad + \| F'(x^{m-1}) - B_{m-1} \|_F \| x^m - x^{m-1} \| \} \\
&\leq 3\beta \left[\frac{\alpha}{2} \| x^m - x^{m-1} \| + \alpha \sum_{i=0}^{m-2} \| x^{i+1} - x^i \| + \delta \right] \| x^m - x^{m-1} \| \\
&\leq 3\beta \left[\frac{\alpha}{2} (t_m - t_{m-1}) + \alpha t_{m-1} + \delta \right] (t_m - t_{m-1}) = t_{m+1} - t_m.
\end{aligned}$$

This completes the induction step. By (3.8), it is easy to show that there is an $x^* \in D$ such that

$$\lim_{k \to \infty} x^k = x^*.$$

The uniqueness of x^* in $\bar{S}(x^0, \bar{t}) \cap D$ can be obtained from Ortega and Rheinboldt's Theorem 12.5.5 [13, p. 418] by setting $G(x) \equiv x - B_0^{-1} F(x)$. □

4. Local convergence properties

To study the local convergence of our algorithms, we assume that $F: D \subset R^n \to R^n$ has the following property:

There is an $x^* \in D$, such that $F(x^*) = 0$ and $F'(x^*)$ is nonsingular. (4.1)

Theorem 4.1. *Let $F: D \subset R^n \to R^n$ satisfy* (4.1), *and let F' satisfy Lipschitz condition* (2.10). *Also let $\{x^k\}$ be generated by Algorithm* 2.1 *with $|h^k| \leq 1/\sqrt{n}\|x^k - x^{k-1}\|$ and without any global strategy. Then, there exist, $\varepsilon, \delta > 0$ such that if $x^0 \in D$ and B_0 satisfy*

$$\|x^0 - x^*\| < \varepsilon, \qquad \|B_0 - F'(x^*)\|_F \leq \delta,$$

then $\{x^k\}$ is well defined and converges q-superlinearly to x^*.

Proof. Notice that when ε and δ are small enough, we have that $h \leq \frac{1}{6}$, $\beta\delta < \frac{1}{3}$ and that $\bar{S}(x^0, 2t^*) \subset D$ where h, β and t^* are defined in Theorem 3.1. Therefore, by Theorem 3.1,

$$x^k + d^k \in D, \quad k = 0, 1, \ldots.$$

By (2.8),

$$B_k - F'(x^*) = (B_{k-1} - F'(x^*))\left(I - \sum_{j \in c_{i_k}} e_j e_j^{\mathrm{T}}\right) + \sum_{j \in c_{i_k}} (J_k - F'(x^*)) e_j e_j^{\mathrm{T}}. \quad (4.2)$$

Thus,

$$\begin{aligned} \|J_k - F'(x^*)\|_F &= \left\| \int_0^1 (F'(x^k + td^k) - F'(x^*))\, \mathrm{d}t \right\|_F \\ &\leq \alpha(\|x^k - x^*\| + \tfrac{1}{2}\|d^k\|) \\ &\leq \alpha(\|x^k - x^*\| + \|x^k - x^{k-1}\|) \\ &\leq \alpha(2\|x^k - x^*\| + \|x^{k-1} - x^*\|). \end{aligned} \quad (4.3)$$

Let $\sigma(x^{k-1}, x^k) = \max\{\|x^k - x^*\|, \|x^{k-1} - x^*\|\}$. Then it follows from (4.2) and (4.3) that

$$\begin{aligned} \|B_k - F'(x^*)\|_F &\leq \|B_{k-1} - F'(x^*)\|_F + \|J_k - F'(x^*)\|_F \\ &\leq \|B_{k-1} - F'(x^*)\|_F + 3\alpha\sigma(x^{k-1}, x^k). \end{aligned}$$

Thus, by Broyden, Dennis and Moré's Theorem 3.2 [3, p. 228], we know that $\{x^k\}$ converges at least q-linearly to x^*.

According to Dennis and Moré's Theorem 2.2 [6, p. 551], to get q-superlinear convergence, we need only to prove that

$$\lim_{k \to \infty} \frac{\|(B_k - F'(x^*))(x^{k+1} - x^k)\|}{\|x^{k+1} - x^k\|} = 0. \quad (4.4)$$

From (2.12), it follows that

$$\lim_{k \to \infty} \|B_k - F'(x^*)\| = 0. \quad (4.5)$$

This implies (4.4). □

Theorem 4.2. *Assume that F satisfies the hypotheses in Theorem* 4.1. *Then the r-convergence order of Algorithm* 2.1 *is not less than τ, where τ is the unique positive root of*

$$t^{p+1} - t^p - 1 = 0.$$

Proof. Notice that (4.5) implies that there exist k_0 and $\beta > 0$ such that $\|B_k^{-1}\| \leq \beta$ for all $k \geq k_0$. Thus, by Theorem 2.3 and Lemma 4.1.12 in [7],

$$\begin{aligned}
\|x^{k+1} - x^*\| &= \|x^k - x^* - B_k^{-1}F(x^k)\| \\
&= \|B_k^{-1}\{F(x^k) - B_k(x^k - x^*)\}\| \\
&\leq \|B_k^{-1}\|_F\{\|F(x^k) - F(x^*) - F'(x^*)(x^k - x^*)\| \\
&\quad + (\|F'(x^*) - F'(x^k)\|_F + \|F'(x^k) - B_k\|_F)\|x^k - x^*\|\} \\
&\leq \beta\left\{\tfrac{1}{2}\alpha\|x^k - x^*\|^2 \right. \\
&\quad \left. + \left(\alpha\|x^k - x^*\| + \alpha\sum_{j=k-p}^{k-1}\|x^{j+1} - x^j\|\right)\|x^k - x^*\|\right\} \\
&= \beta\left\{\tfrac{3}{2}\alpha\|x^k - x^*\| + \alpha\sum_{j=k-p}^{k-1}\|x^{j+1} - x^j\|\right\}\|x^k - x^*\| \\
&\leq \tfrac{5}{2}\alpha\beta\left(\sum_{j=k-p}^{k}\|x^j - x^*\|\right)\|x^k - x^*\|.
\end{aligned}$$

Thus, the desired result follows from Ortega and Rheinboldt's Theorem 9.2.9 [13, p. 291]. □

5. The modified CM-successive column correction algorithm

Estimate (2.12) shows that when p is small, B_k is a good approximation to $F'(x^k)$. However, B_k still retains information from the previous p steps. Therefore, the following question is reasonable: Can we have a better approximation to $F'(x^k)$ without more function evaluations? Notice that when we get B_k by Algorithm 2.1, we did not use the value of $F(x^k)$. The main idea of the modified CM-successive column correction algorithm stated below is to use all the information we already have to improve our approximation to $F'(x^k)$.

Algorithm 5.1. Given a consistent partition of the columns of the Jacobian, a vector x^0 and a nonsingular matrix B_0 with the same sparsity as the Jacobian, at the initial step:

(1) Set $l = 0$ and $\bar{B}_0 = B_0$.
(2) Solve $\bar{B}_0 s^0 = -F(x^0)$.
(3) Choose x^1 by $x^1 = x^0 + s^0$, or by a global strategy.

At each iteration $k > 0$:

(1) Update B_{k-1} by Algorithm 2.1 to get B_k.
(2) Update B_k by the SB update to get $\bar{B}_k$.
(3) Solve $\bar{B}_k s^k = -F(x^k)$.
(4) Choose x^{k+1} by $x^{k+1} = x^k + s^k$, or by a global strategy.
(5) Check for convergence.

Our numerical results show that Algorithm 5.1 requires fewer iterations than Algorithm 2.1 for many problems. Especially, when the problem is not well behaved, and a global strategy is used, the modified algorithm behaves significantly better than Algorithm 2.1. The cost of the improvement is the computation of the SB update. However, since the Jacobian is sparse, the SB update is relatively cheap. For example, if the number of nonzeros in the matrix is $O(n)$, then only $O(n)$ operations are required. We feel that it is worth doing this rather than computing more function values and solving more linear systems.

Now we will briefly discuss the convergence properties of Algorithm 5.1. Let

$$\bar{J}_k = \int_0^1 F'(x^{k-1} + t(x^k - x^{k-1}))\, dt. \tag{5.1}$$

Since $\bar{J}_k$ performs exactly the same as the secant factor

$$\frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}$$

in one dimensional problems, we call $\bar{J}_k$ the secant operator. It is easy to show the following result.

Lemma 5.1. *If F' satisfies Lipschitz condition* (2.10), *then*

$$\|\bar{J}_k - F'(x^k)\|_F \leq \frac{\alpha}{2}\|x^k - x^{k-1}\|. \tag{5.2}$$

Estimate (5.2) *shows that $\bar{J}_k$ is a good approximation to $F'(x^k)$ when $\|x^k - x^{k-1}\|$ is small.*

Theorem 5.2. *Let F' satisfy Lipschitz condition* (2.10). *If $\{B_k\}$ and $\{\bar{B}_k\}$ are generated by Algorithm* 5.1, *then*

$$\|\bar{B}_k - \bar{J}_k\|_F \leq \|B_k - \bar{J}_k\|_F. \tag{5.3}$$

If, in addition, $\bar{B}_k \neq B_k$, then the strict inequality holds.

Proof. Since $\bar{J}_k \in Q_{y,s} \cap Z$, where Z is defined in Definition 1.1, by Theorem 1.1, we have

$$\|\bar{B}_k - \bar{J}_k\|_F^2 + \|\bar{B}_k - B_k\|_F^2 = \|B_k - \bar{J}_k\|_F^2. \tag{5.4}$$

Then, (5.3) follows from (5.4). □

Notice that in general, $\bar{B}_k \neq B_k$. Therefore, by Theorem 5.2, $\bar{B}_k$ is usually closer to the secant operator $\bar{J}_k$ than B_k. Thus, $\bar{B}_k$ should be a better approximation to the Jacobian than B_k when B_k retains some information from previous steps. But

theoretically, we have not been able to get a better estimate for $\|\bar{B}_k - F'(x^k)\|_F$ than that for $\|B_k - F'(x^k)\|_F$. However, we can get the following result:

Theorem 5.3. *Let $F: R^n \to R^n$ satisfy Lipschitz condition* (2.10). *Also let $\{B_k\}$ and $\{x^k\}$ be generated by Algorithm* 5.1. *Then,*

$$\|\bar{B}_k - F'(x^k)\|_F \leq 2\alpha \sum_{j=k-p+1}^{k} \|x^j - x^{j-1}\|. \tag{5.5}$$

Proof. By (5.3),

$$\begin{aligned}\|\bar{B}_k - F'(x^k)\|_F &\leq \|\bar{B}_k - \bar{J}_k\|_F + \|\bar{J}_k - F'(x^k)\|_F \\ &\leq \|B_k - \bar{J}_k\|_F + \|\bar{J}_k - F'(x^k)\|_F \\ &\leq \|B_k - F'(x^k)\|_F + 2\|\bar{J}_k - F'(x^k)\|_F.\end{aligned}$$

Then, from (2.12) and (5.2), we obtain (5.5). □

From estimate (5.5), it is easy to prove that Algorithm 5.1 has at least the same local convergence properties as Algorithm 2.1.

6. Numerical results

To see how the new algorithms work in practice, we computed six examples by the CPR algorithm, the SB algorithm, Algorithm 2.1, and Algorithm 5.1. In this section, we compare the numerical results from these four algorithms. The global strategy we used in computing the examples is the line search with backtracking strategy (see Dennis and Schnabel [7, p. 126]). For the CPR algorithm, if $p^k = -B_k^{-1}F(x^k)$ is not a descent direction, then we try $-p^k$. If it is not a descent direction either, then the algorithm fails. For the other algorithms, if p^k is not a descent direction, i.e. if the line search fails, then we try $-p_k$. If it is not a descent direction either, then we try the CPR direction. For the CPR algorithm, Algorithm 2.1 and Algorithm 5.1, at step k, we use different h_j^k for each component of x^k instead of one uniform h^k. According to Dennis and Schnabel [7, p. 98], we choose

$$h_j^k = \sqrt{\text{macheps}}\, x_j^k.$$

where macheps is the machine precision. The stopping test we used is

$$\max_{1\leq i\leq n} \frac{|x_i^{k+1} - x_i^k|}{\max\{|x_i^{k+1}|, \text{typ } x_i\}} \leq \varepsilon,$$

where typ x_i is a typical value of x_i given by users. We used double precision, and the machine precision is $2.22d-16$, with typ $x_i = 10^{-8}$ and $\varepsilon = 10^{-5}$. The merit function we used is $\frac{1}{2}\|F(x)\|_2^2$. For all the four algorithms, the initial matrix B_0 is generated by the CPR algorithm.

Example 6.1 is the Extended Rosenbrock Function given by Spedicato [19] (also see Moré, Garbow and Hillstrom [12]). Examples 6.2, 6.3 and 6.4 were given by the author in a previous paper [9], and they can also be seen to be the extensions of the Rosenbrock [17] function (also see [12]) to nonlinear systems of equations with tridiagonal, five-diagonal and seven-diagonal structures. Example 6.5 was given by Broyden [1] (also see [12]). Example 6.6 was given by Moré and Cosnard [11] (also see [12]).

The results are shown in the tables below, where IT is the number of iterations, FM is the number of function ($F(x)$) evaluations used for forming the matrices B_k and checking stopping. FL is the number of function ($F(x)$) evaluations used in line searches. LN is the number of line searches in which the step length $\lambda < 1$. ND is the number of nondecrease directions. x^0 is the initial guess. In the examples, p is the number of groups for an optimal partition of the columns of the Jacobian.

Example 6.1 (Extended Rosenbrock Function)

$$f_{2j-1}(x) = 10(x_{2j} - x_{i-1}^2),$$

$$f_{2j}(x) = 1 - x_{2j-1}, \quad j = 1, \ldots, \frac{n}{2},$$

$$n = 20,$$

$$p = 2,$$

$$x1 = (-1.2, 1, \ldots, -1.2, 1)^{\mathrm{T}}, \quad x2 = (-5, -5, \ldots, -5)^{\mathrm{T}}.$$

Table 6.1

Algorithms	$x^0 = x1$					$x^0 = x2$				
	IT	FM	FL	LN	ND	IT	FM	FL	LN	ND
CPR	15	46	12	12	0	17	52	13	13	0
SB	22	25	27	17	0	40	43	124	36	6
Alg. 2.1	17	36	14	17	0	25	52	36	21	1
Alg. 5.1	18	38	16	13	0	17	36	15	12	0

Example 6.2 (tridiagonal)

$$f_1(x) = 4(x_1 - x_2^2),$$

$$f_j(x) = 8x_j(x_j^2 - x_{j-1}) - 2(1 - x_j) + 4(x_j - x_{j+1}^2), \quad j = 2, \ldots, n-1,$$

$$f_n(x) = 8x_n(x_n^2 - x_{n-1}) - 2(1 - x_n),$$

$$n = 36,$$

$$p = 3,$$

$$x1 = (-2, -2, \ldots, -2)^{\mathrm{T}}, \quad x2 = (12, 12, \ldots, 12)^{\mathrm{T}}.$$

Table 6.2

Algorithms	$x^0 = x1$					$x^0 = x2$				
	IT	FM	FL	LN	ND	IT	FM	FL	LN	ND
CPR	16	65	7	7	0	41	165	28	28	0
SB	40	47	51	16	5	306	313	918	229	33
Alg. 2.1	28	59	27	14	0	55	113	130	35	7
Alg. 5.1	22	47	8	8	0	50	103	54	30	1

Example 6.3 (five-diagonal)

$$f_1(x) = 4(x_1 - x_2^2) + x_2 - x_3^2,$$
$$f_2(x) = 8x_2(x_2^2 - x_1) - 2(1 - x_2) + 4(x_2 - x_3^2) + x_3 - x_4^2,$$
$$f_j(x) = 8x_j(x_j^2 - x_{j-1}) - 2(1 - x_j) + 4(x_j - x_{j+1}^2) + x_{j-1}^2 - x_{j-2} + x_{j+1} - x_{j+2}^2, \quad j = 3, \ldots, n-2,$$
$$f_{n-1}(x) = 8x_{n-1}(x_{n-1}^2 - x_{n-2}) - 2(1 - x_{n-1}) + 4(x_{n-1} - x_n^2) + x_{n-2}^2 - x_{n-3},$$
$$f_n(x) = 8x_n(x_n^2 - x_{n-1}) - 2(1 - x_n) + x_{n-1}^2 - x_{n-2},$$
$$n = 36,$$
$$p = 5,$$
$$x1 = (-2, -2, \ldots, -2)^T, \quad x2 = (-3, -3, \ldots, -3)^T.$$

Table 6.3

Algorithms	$x^0 = x1$					$x^0 = x2$				
	IT	FM	FL	LN	ND	IT	FM	FL	LN	ND
CPR	32	193	35	24	0	16	97	7	7	0
SB	66	78	85	32	8	81	92	108	39	8
Alg. 2.1	90	185	541	71	26	33	71	17	13	0
Alg. 5.1	28	61	27	14	1	20	45	5	4	0

Example 6.4 (seven-diagonal)

$$f_1(x) = 4(x_1 - x_2^2) + x_2 - x_3^2 + x_3 - x_4^2,$$
$$f_j(x) = 8x_j(x_j^2 - x_{j-1}) - 2(1 - x_j) + 4(x_j - x_{j+1}^2) + x_{j-1}^2 - x_{j-2} + x_{j+1} - x_{j+2}^2 + x_{j-2}^2 - x_{j-3} + x_{j+2} - x_{j+3}^2, \quad j = 2, \ldots, n-1,$$
$$f_n(x) = 8x_n(x_n^2 - x_{n-1}) - 2(1 - x_n) + x_{n-1}^2 - x_{n-2} + x_{n-2}^2 - x_{n-3},$$
$$n = 36,$$
$$p = 7,$$
$$x1 = (-2, -2, \ldots, -2)^T, \quad x2 = (-3, -3, \ldots, -3)^T.$$

Table 6.4

Algorithms	$x^0 = x1$					$x^0 = x2$				
	IT	FM	FL	LN	ND	IT	FM	FL	LN	ND
CPR	19	153	10	10	0	18	145	8	8	0
SB	43	59	28	15	3	77	93	146	34	16
Alg. 2.1	34	75	46	18	2	fail				
Alg. 5.1	23	53	15	7	1	27	61	22	12	1

Example 6.5 (Broyden Tridiagonal Function)

$$f_i(x) = (3-2x_i)x_i - x_{i-1} - 2x_{i+1} + 1,$$

$$x_0 = x_{n+1} = 0,$$

$$n = 36,$$

$$p = 3,$$

$$x1 = (-1, -1, \ldots, -1)^{\mathrm{T}}, \quad x2 = (-16, -16, \ldots, -16)^{\mathrm{T}}.$$

Table 6.5

Algorithms	$x^0 = x1$					$x^0 = x2$				
	IT	FM	FL	LN	ND	IT	FM	FL	LN	ND
CPR	5	21	0	0	0	9	37	0	0	0
SB	7	11	0	0	0	33	37	4	4	0
Alg. 2.1	6	15	0	0	0	13	29	0	0	0
Alg. 5.1	6	15	0	0	0	11	25	0	0	0

Example 6.6 (Discrete Boundary Value Function)

$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + \frac{h^2}{2}(x_i + t_i + 1)^3$$

$$h = \frac{1}{n+1}, \quad t_i = ih, \quad x_0 = x_{n+1} = 0,$$

$$n = 36,$$

$$x1 = (\eta_j)^{\mathrm{T}}, \quad \eta_j = t_j(t_j - 1), \quad x2 = (50, 50, \ldots, 50)^{\mathrm{T}},$$

From the numerical results it can be seen that the number of function evaluations required by Algorithm 5.1 is always less (sometimes much less) than that required by the CPR algorithm, and in at least eight cases (Example 6.1, $x^0 = x2$, Example 6.2, 6.3, 6.4, Example 6.5, $x^0 = x2$ and Example 6.6, $x^0 = x2$) it is even less than that required by the SB algorithm.

On the other hand, the number of iterations required by Algorithm 5.1 is usually greater than that required by the CPR algorithm. However, the difference is usually

Table 6.6

Algorithms	$x^0 = x1$					$x^0 = x2$				
	IT	FM	FL	LN	ND	IT	FM	FL	LN	ND
CPR	3	13	0	0	0	12	49	0	0	0
Schubert	4	8	0	0	0	30	34	3	3	0
Alg. 2.1	4	11	0	0	0	19	41	0	0	0
Alg. 5.1	4	11	0	0	0	16	35	0	0	0

not significant. Meanwhile, the number of iterations required by Algorithm 5.1 is less than that required by the SB algorithm in eleven of the twelve cases, and when the problem is badly nonlinear and (or) the starting point is far away from the solution, for example, in the eight cases mentioned above, the difference is significant. Therefore, in this case Algorithm 5.1 may be very effective.

Moreover, it is interesting to note that the CPR algorithm never takes nondecrease directions for the test problems. Algorithm 5.1 sometimes takes some nondecrease directions. However, the number of the nondecrease directions for Algorithm 5.1 is usually much less than that for the SB algorithm. Algorithm 2.1 takes more nondecrease directions than Algorithm 5.1, but the number of nondecrease directions is usually less than that for the SB algorithm.

Furthermore, the CPR algorithm takes the fewest line search steps for the test problems. Algorithm 5.1 sometimes takes more line search steps than the CPR algorithm, but the difference is usually not significant. The SB algorithm takes a lot more line search steps than the CPR algorithm and Algorithm 5.1 in the eight cases mentioned above. Algorithm 2.1 usually takes fewer line search steps than the SB algorithm.

Also, note that in the eight cases mentioned above, Algorithm 5.1 performs much better than Algorithm 2.1, and for Example 6.5, $x^0 = x1$, and Example 6.6, $x^0 = x1$ they perform exactly the same. Therefore, it seems that Algorithm 5.1 can really improve Algorithm 2.1 when the problem is badly nonlinear or the starting point is far away from the solution, and when the problem has a good behavior and starting point is close to the solution, these two algorithms perform almost the same.

Finally, from the results of Example 6.1 it can be seen that when p, the number of groups in the partition of the columns of the Jacobian, is small, Algorithm 2.1 may have a very good efficiency. When p increases (Examples 6.3 and 6.4), its efficiency may decrease. In this case, Algorithm 5.1 may have a better efficiency.

7. Concluding remarks

We have presented two algorithms for solving sparse nonlinear systems of equations. The CM-successive column correction algorithm (Algorithm 2.1) is based

on Coleman and Moré's partitioning algorithm and the column-update algorithm. This algorithm uses only two function values at each iterative step, and it is q-superlinearly convergent. Using this algorithm, one group of the columns of B_k is displaced at each step. Note that it is not necessary to update just one group at each iterative step. Instead, we can displace several groups at each iteration, and this gives the algorithm a faster convergence rate. However, if one more group is displaced, then one more function value is required. Therefore, the efficiency of the algorithm depends on the number of the groups displaced at each iterative step.

The modified CM-successive column correction algorithm (Algorithm 5.1) is a combination of the CM-successive column correction algorithm and the SB algorithm. It is also q-superlinearly convergent. Our numerical results indicate that the modified successive column correction algorithm behaves much better than the CM-successive column correction algorithm in some cases. However, we have not been able to prove better theoretical convergence results for the modified CM-successive column correction algorithm than those for the unmodified one. Additional numerical results indicate that these two new algorithms, especially the modified one, are competitive with the CPR-CM algorithm and the SB algorithm.

The idea of the CM-successive column correction algorithms can also be used with Powell and Toint's [15] work, which will lead to methods for unconstrained optimization problems. This will be our future work.

Acknowledgement

This work was started while the author was a Ph.D. student at Rice University, Houston, USA and was completed while the author was visiting the University of Waterloo, Canada. The author would like to express his deepest thanks to Professor John Dennis for his many helpful suggestions and corrections after several readings of preliminary drafts of this paper. The author also would like to thank Dr. Jorge Moré, Professor Richard Tapia and the referees for their helpful suggestions and corrections. The author is grateful to Professor Andy Conn for providing him an opportunity to do more numerical experiments and to revise this paper.

References

[1] C.G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of Computation* 19 (1965) 577-593.

[2] C.G. Broyden, "The convergence of an algorithm for solving sparse nonlinear systems," *Mathematics of Computation* 25 (1971) 285-294.

[3] C.G. Broyden, J.E. Dennis, Jr. and J.J. Moré, "On the local and superlinear convergence of quasi-Newton methods," *Journal of the Institute of Mathematics and its Application* 12 (1973) 223-246.

[4] T.F. Coleman and J.J. Moré, "Estimation of sparse Jacobian and graph coloring problems," *SIAM Journal on Numerical Analysis* 20 (1983) 187-209.

[5] A.R. Curtis, M.J.D. Powell and J.K. Reid, "On the estimation of sparse Jacobian matrices," *IMA Journal of Applied Mathematics* 13 (1974) 117–119.

[6] J.E. Dennis, Jr. and J.J. Moré, "A characterization of superlinear convergence and its application to quasi-Newton methods," *Mathematics of Computation* 28 (1974) 549–560.

[7] J.E. Dennis, Jr. and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983).

[8] Guocheng Feng and Guangye Li, "Column-update quasi-Newton method," *Numerical Mathematics Journal of Chinese Universities* 5 (1983) 139–147.

[9] Guangye Li, "The secant/finite difference algorithm for solving sparse nonlinear systems of equations," Technical Report 86-1, Mathematical Sciences Department, Rice University (Houston, TX, 1986), to appear in *SIAM Journal on Numerical Analysis.*

[10] E. Marwil, "Convergence results for Schubert's method for solving sparse nonlinear equations," *SIAM Journal on Numerical Analysis* 16 (1979) 588–604.

[11] J.J. Moré and M.Y. Cosnard, "Numerical solution of nonlinear equations," *ACM Transactions on Mathematical Software* 5 (1979) 64–85.

[12] J.J. Moré, B.S. Garbow and K.E. Hillstrom, "Testing unconstrained optimization software," *ACM Transactions on Mathematical Software* 7 (1981) 17–41.

[13] J.M. Ortega and W.C. Rheinboldt, *Iterative solution of nonlinear equations in several variables* (Academic Press, New York and London, 1970).

[14] E. Polak, "A modified secant method for unconstrained minimization," Memorandum No. ERL-M373, Electronics Research Lab, College of Engineering, University of California (Berkeley, CA, 1973).

[15] M.J.D. Powell and PH.L. Toint, "On the estimation of sparse Hessian matrices," *SIAM Journal on Numerical Analysis* 16 (1979) 1060–1074.

[16] J.K. Reid, "Least squares solution of sparse systems of non-linear equations by a modified Marquardt algorithm," *Proceedings of the NATO Conference at Cambridge* (North Holland, Amsterdam, 1972) pp. 437–445.

[17] H.H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *Computer Journal* 3 (1960) 175–184.

[18] L.K. Schubert, "Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian," *Mathematics of Computation* 24 (1970) 27–30.

[19] E. Spedicato, "Computational experience with quasi-Newton algorithms for minimization problems of moderately large size," Report CISE-N-175, Segrate (Milano, 1975).