symmetry

MDPI

Article

A Class of Sparse Direct Broyden Method for Solving Sparse Nonlinear Equations

Huiping Cao [1,*] and Jing Han [2]

[1] School of Science, Xi'an Polytechnic University, Xi'an 710048, China
[2] College of Science, Central South University of Forestry and Technology, Changsha 410004, China; jhan@csuft.edu.cn
* Correspondence: huiping_cao@hnu.edu.cn

Abstract: In our paper, we present a sparse quasi-Newton method, called the sparse direct Broyden method, for solving sparse nonlinear equations. The method can be seen as a Broyden-like method and is a least change update satisfying the sparsity condition and direct tangent condition simultaneously. The local and q-superlinear convergence is presented based on the bounded deterioration property and Dennis–Moré condition. By adopting a nonmonotone line search, we establish the global and superlinear convergence. Moreover, the unit step length is essentially accepted. Numerical results demonstrate that the sparse direct Broyden method is effective and competitive for large-scale nonlinear equations.

Keywords: sparse nonlinear equations; quasi-Newton method; direct tangent condition; an approximation to the Jacobian matrix; global and superlinear convergence

MSC: 65K05; 65H10; 90C53

check for updates

Citation: Cao, H.; Han, J. A Class of Sparse Direct Broyden Method for Solving Sparse Nonlinear Equations. *Symmetry* **2022**, *14*, 1552. https://doi.org/10.3390/sym14081552

Academic Editors: Calogero Vetro and Dorian Popa

Received: 11 June 2022
Accepted: 25 July 2022
Published: 28 July 2022

1. Introduction

We consider the nonlinear equation

$$F(x) = 0, x \in R^n, \tag{1}$$

where $F : R^n \to R^n$ is a continuously differentiable mapping. We denote $F'(x)$ as the Jacobian matrix of $F(x)$ at x and pay attention to the case $F'(x)$ having sparse or special structures. Specifically, one has

$$F(x) = (F_1(x), F_2(x), \cdots, F_n(x))^T,$$

and

$$F'(x) = (\nabla F_1(x), \nabla F_2(x), \cdots, \nabla F_n(x))^T.$$

Nonlinear equations arise from many scientific and engineering problems and have various applications in the fields such as physics, biology, and many other fields [1].

The linearization of nonlinear Equation (1) at an iterative point x_k is

$$F(x) \approx F(x_k) + F'(x_k)(x - x_k) = 0;$$

when $F'(x_k)$ is nonsingular, we obtain the Newton–Raphson method

$$x_{k+1} = x_k - F'(x_k)^{-1}F(x_k).$$

Newton's method is theoretically efficient because it is locally quadratically convergent when the Jacobian matrix is nonsigular and Lipschitz continuous at the solution of $F(x)$ [2]. However, at each iteration, Newton's method must compute the exact Jacobian matrix to keep the quadratic convergence rate. The idea of quasi-Newton methods is to approximate the Jacobian matrix $F'(x_k)$ by a quasi-Newtonian matrix B_k with an acceptable reduction of convergence rate. However, at each iteration, Newton's method must compute the exact Jacobian matrix. To avoid computing the derivatives directly, quasi-Newton methods have been proposed, where $F'(x_k)$ is approximated by a quasi-Newton matrix $B_k \in R^{n\times n}$. Thus, quasi-Newton methods generate an iteration as follows:

$$x_{k+1} = x_k + \alpha_k d_k,$$

where the step length $\alpha_k > 0$ is determined by some line search strategies, and d_k is the quasi-Newton direction obtained by solving the subproblem

$$F(x_k) + B_k d_k = 0.$$

Usually, as an approximation to the Jacobian matrix $F'(x_k)$, matrix B_k usually satisfies the so-called quasi-Newton condition

$$B_{k+1}s_k = y_k,$$

where

$$\begin{aligned} s_k &= s_{k+1} - s_k = \alpha_k d_k, \\ y_k &= F(x_{k+1}) - F(x_k). \end{aligned}$$

The quasi-Newton matrix B_k can be updated by kinds of quasi-Newton update formulae, such as Broyden's method, Powell's symmetric Broyden method, BFGS method, and DFP methods [3,4].

Quasi-Newton methods are popular among small and medium-scale problems, since they possess local and superlinear convergence without computing the Jacobian [5–7]. However, when the dimension of nonlinear equations is large, the matrix B_k will be dense. Then, the computation and time complexity will be high. There are two considerations to motivate us to consider the sparse quasi-Newton methods for solving sparse nonlinear equations in this paper. One is the fact that there are lots of nonlinear equations with sparse or special Jacobian. Moreover, quasi-Newton methods for solving (1) have a good property that they can maintain the sparse structure of Jacobian matrices. Thus, in this paper, we are interested in constructing a sparse quasi-Newton method for solving sparse nonlinear equations, where the Jacobian matrix $F'(x_k)$ has sparse or special structure. Earlier work on sparse quasi-Newton methods was carried out by Schubert [8] and Toint [9], where Schubert modified Broyden's method by updating B_k row by row so that the sparsity can be maintained and Toint studied sparse and symmetric quasi-Newton methods. There also have been many kinds of methods for solving large-scale nonlinear systems, such as limited-memory quasi-Newton methods [10,11], partitioned quasi-Newton methods [12–14], diagonal quasi-Newton method [15,16], and column updating method [17].

However, the global convergence of quasi-Newton methods for nonlinear equations is a relatively difficult topic, not to mention the dense case. This mainly results from the fact that the quasi-Newton direction may not be a descent direction of the merit function

$$\theta(x) = \frac{1}{2}\|F(x)\|^2.$$

Griewank [18] and Li and Fukushima [19] have proposed some line search techniques to establish the global convergence of the quasi-Newton method.

The purpose of our paper is to develop a sparse quasi-Newton method and study its local and global convergence. We consider Broyden's method

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k}.$$

If we replace y_k with $F'(x_{k+1})s_k$, we can obtain the following update

$$B_{k+1} = B_k + \frac{(F'(x_{k+1}) - B_k)s_k s_k^T}{s_k^T s_k},$$

which fulfills the direct tangent condition [20,21]

$$B_{k+1}s_k = F'(x_{k+1})s_k.$$

We call the corresponding method the direct Broyden method. Then, we will develop a sparse direct Broyden method, which enjoys the following nice properties: (a) the new sparse quasi-Newton method is a least change update satisfying the direct tangent condition; (b) the proposed method can preserve the sparsity property of the original Jacobian matrix $F'(x)$ exactly; and (c) the sparse direct Broyden method is globally and superlinearly convergent. Presented limited numerical results demonstrate that our algorithm has better performance than Schubert's method and the direct Broyden method in iteration counts, function evaluation counts, and Broyden's mean convergence rate.

The paper is organized as follows: in Section 2, we propose a sparse direct Broyden method and list its nice property. For the full step sparse direct Broyden method, local and superlinear convergence is also given. By adopting a nonmonotone line search, we prove the global and superlinear convergence of the method proposed in Section 2. Moreover, after finitely many iterations, the unit step length will always be accepted. In Section 4, we do some preliminary numerical experiments to test the efficiency of the proposed method. In the last section, we give the conclusion.

2. A New Sparse Quasi-Newton Update and Local Convergence

We pay attention to nonlinear Equation (1), whose Jacobian matrix is sparse or has a special structure. Firstly, we introduce some notations to describe the sparsity structure of the Jacobian as that in [22]. Define the sparsity features of the ith row of $F'(x)$

$$V_i = \{v \in R^n : e_j^T v = 0 \text{ for all } j \text{ such that } (F'(x))_{ij} = e_i^T F'(x)e_j = 0 \text{ for all } x \in R^n\},$$

where e_j is the jth column of identity matrix. Then, we can obtain the set of matrices V that preserve the sparsity pattern of $F'(x)$:

$$V = \{A \in R^{n\times n} : \ A^T e_i \in V_i, \ i = 1, 2, \cdots, n\}.$$

Define a projection operator S_i, $i = 1, 2, \ldots, n$, which maps R^n onto V_i:

$$\Big(S_i(s_k)\Big)_j = (s(i)_k)_j = \begin{cases} (s_k)_j, & \text{if } v_j \neq 0, \\ 0, & \text{if } v_j = 0. \end{cases}$$

Similar to the derivation of Schubert's method [8], we consider the sparse extension of direct Broyden update [2]

$$B_{k+1} = B_k + \frac{(F'(x_{k+1}) - B_k)s_k s_k^T}{s_k^T s_k},$$

which fulfills the direct tangent condition

$$B_{k+1}s_k = F'(x_{k+1})s_k. \tag{2}$$

Then, we can obtain a compact representation of the new sparse quasi-Newton update as

$$B_{k+1} = B_k + \sum_{i=1}^{n} (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k)s_k e_i s(i)_k^T, \tag{3}$$

where the pseudo-inverse of $\alpha \in R$ is defined by

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

The new sparse quasi-Newton method (3) updates the quasi-Newton matrix row by row to preserve the zero and nonzero structure of the Jacobian.

Then, we can obtain a quasi-Newton method as

$$x_{k+1} = x_k + \alpha_k d_k,$$

where d_k can be obtained by solving the following subproblem

$$F(x_k) + B_k d_k = 0,$$

and B_k is updated by sparse direct Broyden update

$$B_{k+1} = B_k + \sum_{i=1}^{n} (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k e_i s(i)_k^T.$$

We call the corresponding method the sparse direct Broyden method. When $\alpha_k \equiv 1$, we refer to it as a full step sparse direct Broyden method.

Lemma 1. *The B_{k+1} defined by (3) is the unique solution to the following minimization problem:*

$$\min\{\|B - B_k\|_F : B \in V \cap Q(F'(x_{k+1}), s_k)\}, \tag{4}$$

where $Q(F'(x_{k+1}), s_k) = \{B \in R^{n\times n} \,|Bs_k = F'(x_{k+1})s_k\}$.

Proof. Firstly, we will prove that $B_{k+1} \in V \cap Q(F'(x_{k+1}), s_k)\}$. For $i = 1, 2, \cdots, n$, multiply both sides of (3) by e_i^T, to obtain

$$e_i^T B_{k+1} = e_i^T B_k + (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k s(i)_k^T.$$

Since $B_k^T e_i \in V_i$ and $s_k \in V_i$, then we have $B_{k+1}^T e_i \in V_i$, which implies $B_{k+1} \in V$.

If $s(i)_k \neq \mathbf{0}$, one has

$$e_i^T B_{k+1} s_k = e_i^T B_k s_k + (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k s(i)_k^T s_k. \tag{5}$$

According to the definition of the operator S_i, we have

$$s(i)_k^T s(i)_k = s(i)_k^T s_k \text{and} (s(i)_k^T s(i)_k)^+ = (s(i)_k^T s(i)_k)^{-1}.$$

Then, (5) can be written as

$$e_i^T B_{k+1} s_k = e_i^T B_k s_k + e_i^T (F'(x_{k+1}) - B_k) s_k = e_i^T F'(x_{k+1}) s_k.$$

If $s(i)_k = \mathbf{0}$, we have

$$e_i^T F'(x_{k+1}) s_k = e_i^T F'(x_{k+1}) s(i)_k = 0,$$

thus $e_i^T B_{k+1} s_k = e_i^T F'(x_{k+1}) s_k$, which implies $B_{k+1} s_k = F'(x_{k+1}) s_k$. Therefore, $B_{k+1} \in Q(F'(x_{k+1}), s_k)\}$.

Then, we will prove the uniqueness. Suppose that $\bar{B}_{k+1} \in Q(F'(x_{k+1}), s_k)\}$. Since $\bar{B}_{k+1} s_k = F'(x_{k+1}) s_k$ and $(\bar{B}_{k+1} - B_k) s_k = (\bar{B}_{k+1} - B_k) s(i)_k$, one has

$$B_{k+1} = B_k + \sum_{i=1}^{n} (s(i)_k^T s(i)_k)^+ e_i^T (\bar{B}_{k+1} - B_k) s_k e_i s(i)_k^T.$$

Taking the Frobenius norm,

$$\begin{aligned}
\|B_{k+1} - B_k\|_F &= \left(\sum_{i=1}^{n} \|e_i^T (B_{k+1} - B_k)\|^2 \right)^{1/2} \\
&= \left(\sum_{i=1}^{n} \|(s(i)_k^T s(i)_k)^+ e_i^T (\bar{B}_{k+1} - B_k) s_k s(i)_k^T \|^2 \right)^{1/2} \\
&= \left(\sum_{i=1}^{n} |(s(i)_k^T s(i)_k)^+ e_i^T (\bar{B}_{k+1} - B_k) s(i)_k|^2 \cdot \|s(i)_k\|^2 \right)^{1/2} \\
&\le \left(\sum_{i=1, s(i)_k \neq \mathbf{0}}^{n} \|e_i^T (\bar{B}_{k+1} - B_k)\|^2 \right)^{1/2} \\
&\le \left(\sum_{i=1}^{n} \|e_i^T (\bar{B}_{k+1} - B_k)\|^2 \right)^{1/2} \\
&= \|\bar{B}_{k+1} - B_k\|_F,
\end{aligned}$$

where the first inequality follows from the triangle inequality. Since the function $f(B) = \|B - B_k\|_F$ is strictly convex and the constraint condition (4) is convex, we can obtain the uniqueness. □

To analyze the local convergence of the full step sparse direct Broyden method, first we show that the bounded deterioration property

$$\|B_{k+1} - F'(x^*)\|_F \le (1 + \alpha_1 \sigma_k) \|B_k - F'(x^*)\|_F + \alpha_2 \gamma_k, \quad (6)$$

is satisfied with some constants $\alpha_1, \alpha_2 \ge 0$, where $\gamma_k = \max\{\|x_k - x^*\|_2, \|x_{k+1} - x^*\|_2\}^2$.

Lemma 2. *Suppose that $F : R^n \to R^n$ is continuously differentiable in D_0, which is an open and convex set. Let $x^* \in D_0$ be a solution of (1) at which $F'(x^*)$ is nonsingular. Suppose that there exists $K = (k_1, k_2, \cdots, k_n) \in R^n$ with $k_i \ge 0$, for $i = 1, 2, \ldots, n$, such that*

$$\|e_i^T (F'(x) - F'(y))\| \le k_i \|x - y\|, \quad \forall x, y \in D_0.$$

Then, one has the estimation

$$\|B_{k+1} - F'(x^*)\|_F^2 \le \|B_k - F'(x^*)\|_F^2 - \frac{\|(B_k - F'(x^*)) s_k\|^2}{\|s_k\|^2} + L^2 \gamma_k,$$

where $L = \|K\|_2$.

Proof. For the case $s_k = \mathbf{0}$, then it is obvious that $F(x_k) = \mathbf{0}$ and $x_k = x^*$. For the case $s_k \neq \mathbf{0}$, subtracting $F'(x^*)$ from both sides of the update formula

$$B_{k+1} = B_k + \sum_{i=1}^{n} (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k e_i s(i)_k^T,$$

and multiplying by e_i^T, $i = 1, 2, \cdots, n$, one has

$$\begin{aligned}
& e_i^T (B_{k+1} - F'(x^*)) \\
= \; & e_i^T (B_k - F'(x^*)) + (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k s(i)_k^T \\
= \; & e_i^T (B_k - F'(x^*)) (I - (s(i)_k^T s(i)_k)^+ s(i)_k s(i)_k^T) \\
& + (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - F'(x^*)) s_k s(i)_k^T.
\end{aligned}$$

Taking norms yields

$$\begin{aligned}
&\|e_i^T(B_{k+1}-F'(x^*))\|_F^2 \\
= \ &\|e_i^T(B_k-F'(x^*))(I-(s(i)_k^Ts(i)_k)^+s(i)_ks(i)_k^T)\|^2 \\
&+(s(i)_k^Ts(i)_k)^+|e_i^T(F'(x_{k+1})-F'(x^*))s_k|^2 \\
= \ &\|e_i^T(B_k-F'(x^*))\|_2^2-(s(i)_k^Ts(i)_k)^+|e_i^TE_ks(i)_k|^2 \\
&+(s(i)_k^Ts(i)_k)^+|e_i^T(F'(x_{k+1})-F'(x^*))s_k|^2 \\
\leq \ &\|e_i^TE_k\|_2^2-\frac{|e_i^TE_ks_k|^2}{\|s_k\|^2}+(s(i)_k^Ts(i)_k)^+|e_i^T(F'(x_{k+1})-F'(x^*))s_k|^2.
\end{aligned} \quad (7)$$

If $s(i)_k=\mathbf{0}$, then we have $(s(i)_k^Ts(i)_k)^+=0$. It is obvious that

$$0=(s(i)_k^Ts(i)_k)^+|e_i^T(F'(x_{k+1})-F'(x^*))s_k|^2\leq k_i^2\sigma_k.$$

If $s(i)_k\neq\mathbf{0}$, it follows that

$$\begin{aligned}
&(s(i)_k^Ts(i)_k)^+|e_i^T(F'(x_{k+1})-F'(x^*))s_k|^2 \\
&\quad = \ (s(i)_k^Ts(i)_k)^+|e_i^T(F'(x_{k+1})-F'(x^*))s(i)_k|^2 \\
&\quad \leq \ \|e_i^T(F'(x_{k+1})-F'(x^*))\|^2 \\
&\quad \leq \ k_i^2\|x_{k+1}-x^*\|^2 \\
&\quad \leq \ k_i^2\sigma_k^2.
\end{aligned}$$

Thus, (7) reduces to

$$\|e_i^T(B_{k+1}-F'(x^*))\|_F^2\leq\|e_i^T(B_k-F'(x^*))\|^2-\frac{|e_i^T(B_k-F'(x^*))s_k|^2}{\|s_k\|^2}+k_i^2\gamma_k,$$

Make a summation to obtain

$$\|(B_{k+1}-F'(x^*))\|_F^2\leq\|(B_k-F'(x^*))\|_F^2-\frac{\|(B_k-F'(x^*))s_k\|^2}{\|s_k\|^2}+L^2\gamma_k. \quad (8)$$

□

Based on the classical framework of Dennis and Moré, we give the following local convergence, which can be proved similar to the case of Broyden's method [6,7].

Theorem 1. *Let the conditions in Lemma 2 hold. Then, there exist constants $\epsilon,\delta>0$ such that, if $\|x_0-x^*\|_2<\epsilon$ and $\|B_0-F'(x^*)\|_F<\delta$, the sequence $\{x_k\}$ is well defined and converges to x^*. Furthermore, the convergence rate is superlinear.*

Proof. According to Lemma 2, one has

$$\|(B_{k+1}-F'(x^*))\|_F\leq\|(B_k-F'(x^*)\|_F+L\gamma_k,$$

which means that the estimation (6) is satisfied with $\alpha_1=0$ and $\alpha_2=L$. Then, we obtain the local and linear convergence of $\{x_k\}$.

Next, we will show the Dennis–Moré condition [7]

$$\lim_{k\to\infty}\frac{\|(B_k-F'(x^*))s_k\|}{\|s_k\|}=0 \quad (9)$$

is satisfied. According to (8), one has

$$\|(B_{k+1} - F'(x^*))\|_F \le \left(\|(B_k - F'(x^*))\|_F^2 - \frac{\|(B_k - F'(x^*))s_k\|^2}{\|s_k\|^2} \right)^{1/2} + L\gamma_k;$$

then, the result can be proved similar to that in [7]. □

3. Algorithm and Global Convergence

In this section, by the use of LF condition [19], we propose a global sparse Broyden method, whose specific steps are listed in the following Algorithm 1.

Algorithm 1 (Sparse direct Broyden Method for solving sparse nonlinear equations)

Step 0. Given constant $\sigma_1, \sigma_2 > 0$ and $\rho, r \in (0,1)$. Given a positive sequence $\{\eta_k\}$ satisfying

$$\sum_{k=0}^{\infty} \eta_k \le \eta < \infty. \tag{10}$$

Given $x_0 \in R^n$, stop tolerance $\epsilon > 0$, and a nonsingular matrix $B_0 \in R^{n\times n}$. Set $k := 0$.
Step 1. Stop if $\|F(x_k)\| \le \epsilon$.
Step 2. Solve the subproblem

$$F(x_k) + B_k d_k = 0 \tag{11}$$

to obtain the quasi-Newton direction d_k.
Step 3. If

$$\|F(x_k + d_k)\| \le \rho\|F(x_k)\| - \sigma_1\|d_k\|^2, \tag{12}$$

then let $\alpha_k := 1$ and go to Step 5. Else, go to Step 4.
Step 4. Set $\alpha_k = r^{i_k}$, where i_k is the smallest nonnegative integer i satisfying

$$\|F(x_k + r^i d_k)\| \le \|F(x_k)\| - \sigma_2\|r^i d_k\|^2 + \eta_k\|F(x_k)\|, \tag{13}$$

where η_k is defined as in (10).
Step 5. Set $x_{k+1} := x_k + \alpha_k d_k$.
Step 6. Update B_k to obtain B_{k+1} by sparse direct Broyden update

$$B_{k+1} = B_k + \sum_{i=1}^{n} (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k)s_k e_i s(i)_k^T, \tag{14}$$

Set $k := k+1$. Go to Step 1.

Remark 1. *It is noticed that the update formula (14) may be singular when B_k is nonsingular. In this case, we use a similar technique in [22,23] and give the following discussion about the nonsingular sparse direct Broyden update.*

Set $H_0 = B_k$, and for $i = 1, 2, \ldots, n$, let

$$\begin{aligned} H_i &= H_0 + \sum_{j=1}^{i} \theta_k^j (s(j)_k^T s(j)_k)^+ e_j^T (F'(x_{k+1}) - B_k)s_k e_j s(j)_k^T \\ &= H_{i-1} + \theta_k^i (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k)s_k e_i s(i)_k^T. \end{aligned}$$

Since $e_i^T H_0 = e_i^T H_1 = \cdots = e_i^T H_{i-1}$, then

$$H_i = H_{i-1} + \theta_k^i (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - H_{i-1})s_k e_i s(i)_k^T.$$

For a scalar $\alpha \in (0,1)$, θ_k^i can be chosen such that

$$| \, det \, H_i \, | \geq | \, \sqrt[n]{\alpha} \, | \, \, det \, H_{i-1} \, |, \theta_k^i \in \left[\frac{1-\sqrt[n]{\alpha}}{1+\sqrt[n]{\alpha}}, 1 \right].$$

Thus, $| \, det B_{k+1} \, | \geq \alpha \, | \, det B_k \, |$ and θ_k^i can be chosen so that

$$B_{k+1} \text{ is nonsingular, and } \, | \, \theta_k^i - 1 \, | \leq \hat{\theta} < 1.$$

Thus, we can define the sparse direct Broyden-like update formula as

$$B_{k+1} = B_k + \sum_{i=1}^{n} \theta_k^i (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k e_i s(i)_k^T.$$

Remark 2. *It can be seen that the update formula (14) includes $F'(x_{k+1})$, but it does not need to compute $F'(x_{k+1})$ in practice. Automatic differentiation is a chain-rule-based technique for evaluating the derivatives with respect to the input variables of functions defined by a high-level computer program. Automatic Differentiation has two basic modes of operations, the forward mode and the reverse mode. In the forward mode, the derivatives are propagated throughout the computation using the chain rule, while in the reverse mode the adjoint derivatives are propagated backwards. The forward mode and reverse mode of automatic differentiation provide the possibility to compute $F'(x)s$ and $\sigma^T F'(x)$ exactly within machine accuracy for given vectors x, s and σ.*

To establish the global convergence, we need the following conditions.

Assumption 1. *(1) F is continuously differentiable on Ω, which is a bounded level set defined by*

$$\Omega = \{x \in R^n \mid \|F(x)\| \leq e^{\eta} \|F(x_0)\|\}.$$

(2) $F'(x)$ is Lipschitz continuous on Ω with Lipschitz constant $L > 0$

$$\|F'(x) - F'(y)\| \leq L\|x - y\|, \forall x, y \in \Omega.$$

(3) $F'(x)$ is nonsingular for any $x \in \Omega$.

First, we give the following important lemmas.

Lemma 3. *The sequence $\{x_k\}$ generated by Algorithm 1 is contained in Ω. Moreover, it holds that*

$$\sum_{k=0}^{\infty} \|s_k\|^2 < \infty, \tag{15}$$

and the sequence $\{\|F(x_k)\|\}$ converges.

Proof. According to the line search (12) and (13), one has for any k

$$\begin{aligned}\|F(x_{k+1})\| &\leq (1+\eta_k)\|F(x_k)\| \\ &\vdots \\ &\leq \|F(x_0)\|\left[\Pi_{j=0}^{k}(1+\eta_j)\right] \\ &\leq \|F(x_0)\|\left[\frac{1}{k+1}\sum_{j=0}^{k}(1+\eta_j)\right]^{k+1} \\ &= \|F(x_0)\|\left[1+\frac{1}{k+1}\sum_{j=0}^{k}\eta_j\right]^{k+1} \\ &\leq \|F(x_0)\|\left[\left(1+\frac{\eta}{k+1}\right)^{\frac{k+1}{\eta}}\right]^{\eta} \\ &\leq e^{\eta}\|F(x_0)\|.\end{aligned}$$

Thus, $\{x_k\}$ is contained in level set Ω. Moreover, the sequence $\{\|F(x_k)\|\}$ is bounded.

On the basis of (12) and (13), we have for each k that

$$\sigma_0\|s_k\|^2 = \sigma_0\|x_{k+1}-x_k\|^2 \leq \|F(x_k)\| - \|F(x_{k+1})\| + \eta_k\|F_k\|,$$

where $\sigma_0 = \min\{\sigma_1, \sigma_2\}$. We can obtain (15) by taking summation on both sides for k from 0 to ∞.

Finally, since $\{\|F(x_k)\|\}$ satisfies

$$\|F(x_k+\alpha_k d_k)\| \leq (1+\eta_k)\|F(x_k)\|,$$

and $\{\eta_k\}$ satisfies

$$\sum_{k=0}^{\infty}\eta_k \leq \eta < \infty,$$

we then obtain the convergence of $\{\|F(x_k)\|\}$. □

Denote

$$\delta_k = \frac{\|(F'(x_{k+1})-B_k)s_k\|}{\|s_k\|} = \frac{\|F'(x_{k+1})s_k + F(x_k)\|}{\|s_k\|}.$$

Lemma 4. *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1, and $F'(x)$ is Lipschitz continuous with a common Lipschitz constant $L > 0$. If*

$$\sum_{k=0}^{\infty}\|s_k\|^2 < \infty,$$

then we have

$$\lim_{t\to\infty}\frac{1}{t}\sum_{k=0}^{t-1}\delta_k^2 = 0. \quad (16)$$

In addition, there exists a subsequence of $\{\delta_k\}$ tending to zero. If

$$\sum_{k=0}^{\infty}\|s_k\| < \infty,$$

then we have

$$\sum_{k=0}^{\infty}\delta_k^2 < \infty. \quad (17)$$

In addition, the whole sequence $\{\delta_k\}$ *converges to zero.*

Proof. According to the update (14), we have

$$e_i^T B_{k+1} = e_i^T B_k + (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k s(i)_k^T.$$

Subtracting $e_i^T F'(x_{k+1})$, we obtain

$$\begin{aligned} & e_i^T (B_{k+1} - F'(x_{k+1})) \\ = \;& e_i^T (B_k - F'(x_{k+1})) + (s(i)_k^T s(i)_k)^+ e_i^T (F'(x_{k+1}) - B_k) s_k s(i)_k^T \\ = \;& e_i^T (B_k - F'(x_{k+1})) \Big(I - (s(i)_k^T s(i)_k)^+ s_k s(i)_k^T \Big) \\ = \;& e_i^T (B_k - F'(x_{k+1})) \Big(I - (s(i)_k^T s(i)_k)^+ s(i)_k s(i)_k^T \Big). \end{aligned}$$

Taking norms yields

$$\begin{aligned} & \|e_i^T (B_{k+1} - F'(x_{k+1}))\|^2 \\ = \;& \|e_i^T (B_k - F'(x_{k+1}))(I - (s(i)_k^T s(i)_k)^+ s(i)_k s(i)_k^T)\|^2 \\ = \;& \|e_i^T (B_k - F'(x_{k+1}))\|^2 - (s(i)_k^T s(i)_k)^+ (e_i^T (B_k - F'(x_{k+1})) s_k)^2 \\ \leq \;& \|e_i^T (B_k - F'(x_{k+1}))\|^2 - \frac{\|e_i^T (B_k - F'(x_{k+1})) s_k\|^2}{\|s_k\|^2}. \end{aligned}$$

Since $\|B_{k+1} - F'(x_{k+1})\|_F^2 = \sum_{i=1}^n \|e_i^T (B_{k+1} - F'(x_{k+1}))\|^2$, making summation from $i = 1$ to n yields

$$\begin{aligned} & \|B_{k+1} - F'(x_{k+1})\|_F^2 \\ \leq \;& \sum_{i=1}^n \left(\|e_i^T (B_k - F'(x_{k+1}))\|^2 - \frac{\|e_i^T (B_k - F'(x_{k+1})) s_k\|^2}{\|s_k\|^2} \right) \\ = \;& \|(B_k - F'(x_{k+1}))\|_F^2 - \frac{\|(B_k - F'(x_{k+1})) s_k\|^2}{\|s_k\|^2} \\ = \;& \|(B_k - F'(x_{k+1}))\|_F^2 - \delta_k^2. \end{aligned}$$

Denote

$$D_k = B_k - F'(x_k) \text{ and } E_k = F'(x_{k+1}) - F'(x_k).$$

Then, one has that, for $k \geq 1$,

$$\begin{aligned} \|D_k\|_F &\leq \|B_{k-1} - F'(x_k)\|_F \leq \|D_{k-1}\|_F + \|E_{k-1}\|_F \\ &\leq \cdots \leq \|D_0\|_F + \sum_{j=0}^{k-1} \|E_j\|_F, \end{aligned}$$

and

$$\begin{aligned} \delta_k^2 &\leq \|B_k - F'(_{k+1})\|_F^2 - \|B_{k+1} - F'(x_{k+1})\|_F^2 \\ &= \|D_k - E_k\|_F - \|D_{k+1}\|_F^2 \\ &\leq \|D_k\|_F^2 - \|D_{k+1}\|_F^2 + \|E_k\|_F^2 + 2\|E_k\|_F \|D_k\|_F \\ &\leq \|D_k\|_F^2 - \|D_{k+1}\|_F^2 + \|E_k\|_F^2 + 2\|E_k\|_F \cdot \left(\|D_0\|_F + \sum_{j=0}^{k-1} \|E_j\|_F \right). \end{aligned}$$

Making summation on both sides from $k = 0$ to $t - 1$, we have for $1 \le p < t$

$$\begin{aligned} \sum_{k=1}^{t-1} \delta_k^2 &\le \|D_0\|_F^2 + \sum_{k=0}^{t-1} \|E_k\|_F^2 + 2\sum_{k=1}^{t-1} \|E_k\|_F \left(\|D_0\|_F + \sum_{j=0}^{k-1} \|E_j\|_F \right) \\ &\le \|D_0\|_F^2 + \sum_{k=0}^{t-1} \|E_k\|_F^2 + 2\|D_0\|_F \sum_{k=1}^{t-1} \|E_k\|_F + 2\sum_{k=1}^{t-1} \|E_k\|_F \sum_{j=0}^{k-1} \|E_j\|_F \\ &= \|D_0\|_F^2 + 2\|D_0\|_F^2 \sum_{k=1}^{t-1} \|E_k\|_F + 2\left(\sum_{k=1}^{t-1} \|E_k\|_F \right)^2 \\ &= \left(\|D_0\|_F + \sum_{k=0}^{t-1} \|E_k\|_F \right)^2 \\ &\le \left(\|D_0\|_F + \sum_{k=0}^{p-1} \|E_k\|_F + \sum_{k=p}^{t-1} \|E_k\|_F \right)^2 \\ &\le 2\left(\|D_0\|_F + \sum_{k=0}^{p-1} \|E_k\|_F \right)^2 + 2\left(\sum_{k=p}^{t-1} \|E_k\|_F \right)^2 \\ &\le 2\left(\|D_0\|_F + \sum_{k=0}^{p-1} \|E_k\|_F \right)^2 + 2(t-p) \sum_{k=p}^{t-1} \|E_k\|_F^2. \end{aligned} \tag{18}$$

Dividing both sides by t and letting $t \to \infty$, we have

$$\lim_{t\to\infty} \frac{1}{t} \sum_{k=1}^{t-1} \delta_k^2 \le 2 \lim_{t\to\infty} \frac{t-p}{t} \sum_{k=p}^{t-1} \|E_k\|_F^2 \le 2 \sum_{k=p}^{\infty} \|E_k\|_F^2.$$

If $\sum_{k=0}^{\infty} \|s_k\|^2 < \infty$, then the Lipschitz continuity of $F'(x)$ together with the last inequality implies

$$\lim_{t\to\infty} \frac{1}{t} \sum_{k=0}^{t-1} \delta_k^2 = 0.$$

Then, there is a subsequence of $\{\delta_k\}$ tending to zero. If $\sum_{k=0}^{\infty} \|s_k\| < \infty$, then (17) comes from (18). Moreover, the whole sequence $\{\delta_k\}$ converges to zero. This completes the proof. □

Theorem 2. *Let the conditions in Assumption 1 hold. Then, the sequence $\{x_k\}$ generated by Algorithm 1 converges to the unique solution x^* of (1).*

Proof. We first verify

$$\liminf_{k\to\infty} \|F(x_k)\| = 0. \tag{19}$$

According to Lemma 3, the sequence $\{\|F(x_k)\|\}$ converges. Thus, we only need to prove that there is an accumulation point of $\{x_k\}$, which is the unique solution of (1). If there are infinitely many α_k, which is obtained by the line search condition (12), then

$$\|F(x_{k+1})\| \le \rho \|F(x_k)\|$$

holds for infinitely many k. This indicates $\liminf_{k\to\infty} \|F(x_k)\| = 0$.

There are only finite many α_k, which is obtained by the line search condition (12). By (15) and Lemma 4, there is a subsequence $\{\delta_k\}_{k\in K}$ converging to zero. Since $\{x_k\}_K$ is bounded, we may assume that $\{x_k\}_K \to x^*$ without loss of generality. Hence, $\{F'(x_{k+1})\}$

tends to $F'(x^*)$, and there exists a constant C_1 such that $\|F'(x_{k+1})\| \leq C_1$ for all sufficiently large $k \in K$. According to the subproblem (11) and the definition of δ_k, one has

$$\begin{aligned} \|d_k\| &= \|F'(x_{k+1})^{-1}((F'(x_{k+1}) - B_k)d_k - F(x_k))\| \\ &\leq \|F'(x_{k+1})^{-1}\| \Big(\|(F'(x_{k+1}) - B_k)d_k\| + \|F(x_k)\| \Big) \\ &\leq C_1(\delta_k \|d_k\| + \|F(x_k)\|), \end{aligned}$$

which indicates that there exists a constant M_1 such that

$$\|d_k\| \leq M_1 \|F(x_k)\|$$

holds for all sufficiently large $k \in K$. Thus, the subsequence $\{d_k\}_K$ is bounded, and we can assume that $\{d_k\}_K \to d^*$. Since $\|(F'(x_{k+1}) - B_k)d_k\| = \delta_k \|d_k\|$, then we have

$$B_k d_k \to F'(x^*)d^*, k \to \infty, k \in K.$$

Taking limit in the subproblem (11) as $k \to \infty, k \in K$, one has

$$F'(x^*)d^* + F(x^*) = 0. \tag{20}$$

Denote $\alpha^* = \limsup_{k\to\infty, k\in K} \alpha_k$. It is clear that $\alpha^* \geq 0$ and $\alpha^* d^* = 0$. If $\bar{\alpha} > 0$, then $d^* = 0$; hence, it follows from (20) that $F(x^*) = 0$. If $\alpha^* = 0$, or equivalently $\lim_{k\to\infty} \alpha_k = 0$. According to the line search rule, when $k \in K$ is sufficiently large, $\alpha_k < 1$ and hence

$$\|F(x_k + \rho^{-1}\alpha_k d_k)\| - \|F(x_k)\| > -\sigma_2 \|\rho^{-1}\alpha_k d_k\|^2. \tag{21}$$

Multiplying both sides of (21) by $(\|F(x_k + \rho^{-1}\alpha_k d_k))\| + \|F(x_k)\|)/(\rho^{-1}\alpha_k)$ and taking limit as $k \to \infty, k \in K$, we obtain

$$F(x^*)^T F'(x^*)d^* \geq 0.$$

Combined with (20), we have $F(x^*) = 0$. Then, we complete the proof. □

In what follows, we will show that, when k is sufficiently large, the $\alpha_k \equiv 1$ will be accepted.

Theorem 3. *Suppose Assumption 1 holds and $\{x_k\}$ is generated by Algorithm 1. Then, there exist a constant $\delta > 0$ and an index $\bar{k}$ such that $\alpha_k = 1$ whenever $\delta_k \leq \delta$ and $k \geq \bar{k}$. Furthermore, the inequality (12) holds for all $k \geq \bar{k}$ satisfying $\delta_k \leq \delta$.*

Proof. According to Theorem 2, $\{x_k\}$ converges to the solution x^* of (1). Then, there exists a constant $M_2 > 0$ such that $\|F'(x_{k+1})^{-1}\| \leq M_2$ for all k sufficiently large. Moreover, it can be deduced similarly that there exists constant $M_3 > 0$ such that, when $\delta_k \leq \delta$ and k is large enough,

$$\| d_k \| \leq M_3 \| F(x_k) \| . \tag{22}$$

By the subproblem (11), one has

$$\begin{aligned} F'(x_{k+1})(x_k + d_k - x^*) &= F'(x_{k+1})(x_k - x^*) + (F'(x_{k+1}) - B_k)d_k - F(x_k) \\ &= (F'(x_{k+1}) - F'(x^*))(x_k - x^*) + (F'(x_{k+1}) - B_k)d_k \\ &\quad -F(x_k) + F(x^*) + F'(x^*)(x_k - x^*). \end{aligned}$$

This implies

$$\begin{aligned}
\|x_k + d_k - x^*\| &\leq \|F'(x_{k+1})^{-1}\|\Big(\|F'(x_{k+1}) - F'(x^*)\|\,\|x_k - x^*\| + \|(F'(x_{k+1}) - B_k)d_k\| \\
&\quad + \|F(x_k) - F(x^*) - F'(x^*)(x_k - x^*)\|\Big) \\
&\leq M_2(o(\|x_k - x^*\|) + \delta_k\|d_k\|) \\
&\leq M_2(o(\|x_k - x^*\|) + \delta_k M_3\|F(x_k) - F(x^*)\|) \\
&\leq M_2(o(\|x_k - x^*\|) + \delta_k M_3 M_4\|x_k - x^*\|),
\end{aligned}$$

where M_4 an upper bound of $\|F'(x)\|$ on the level set Ω. Then, by the last inequality, we have

$$\begin{aligned}
\|F(x_k + d_k)\| &= \|F(x_k + d_k) - F(x^*)\| \\
&\leq M_4\|x_k + d_k - x^*\| \\
&\leq M_2 M_4(o(\|x_k - x^*\|) + \delta_k M_3 M_4\|x_k - x^*\|),
\end{aligned}$$

On the other hand, by the nonsingularity of $F'(x^*)$ and the convergence of $\{x_k\}$, there is a constant $m > 0$ such that the inequality

$$\|F(x_k)\| = \|F(x_k) - F(x^*)\| \geq m\|x_k - x^*\| \tag{23}$$

holds for all k sufficiently large. Thus, we deduce from (22) and (23) that, when $\delta_k \leq \delta$,

$$\begin{aligned}
&\|F(x_{k+1})\| - \rho\|F(x_k)\| + \sigma_1\|d_k\|^2 \\
\leq\ & M_2 M_4(o(\|x_k - x^*\|) + \delta_k M_3 M_4\|x_k - x^*\| - \rho m\|x_k - x^*\| + \sigma_1 M_3^2\|F(x_k)\|^2 \\
\leq\ & (M_2 M_3 M_4^2\delta_k - \rho m)\|x_k - x^*\| + o(\|x_k - x^*\|) + \sigma_1 M_2^2 M_3^2\|x_k - x^*\|^2 \\
\leq\ & -(\rho m - M_2 M_3 M_4^2\delta_k)\|x_k - x^*\| + o(\|x_k - x^*\|).
\end{aligned}$$

Let $\delta = \min\{\delta, \frac{1}{2}\rho m(M_2 M_3 M_4^2)^{-1}\}$; then, we complete the proof. □

The following theorem presents that Algorithm 1 is superlinearly convergent.

Theorem 4. *Let the Assumption 1 hold. Then, the sequence $\{x_k\}$ generated by Algorithm 1 converges to the unique solution x^* of (1) superlinearly.*

Proof. Let δ and $\bar{k}$ be as defined by Theorem 3. Then, according to Lemma 4, we have that

$$\frac{1}{k}\sum_{j=0}^{k-1}\delta_j^2 \leq \frac{1}{2}\delta^2$$

holds for all $k \geq \tilde{k}$, which implies that, in this case, there are at least $\lceil \frac{k}{2} \rceil$ many indices $j \leq k$ satisfying $\delta_j \leq \delta$. Let $k' = \max\{\bar{k}, \tilde{k}\}$. Moreover, on the basis of Theorem 3, for any $k \geq 2k'$, there are at least $\lceil \frac{k}{2} \rceil - k'$ many indices $j \leq k$, which make $\alpha_j = 1$ and

$$\|F(x_{j+1})\| = \|F(x_j + d_k)\| \leq \rho\|F(x_j)\|. \tag{24}$$

Define $J_k = \{j \mid (24) \text{ holds }\}$ and $|J_k|$ as the number of the elements in J_k. Then, $|J_k| \geq \frac{k}{2} - k' - 1$. On the other side, for each $j \notin J_k$, we have

$$\|F(x_{j+1})\| \leq (1 + \eta_k)\|F(x_j)\|. \tag{25}$$

Multiplying inequalities (24) with $j \in J_k$ and (25) with $j \notin J_k$ from $j = k'$ to k yields

$$\|F(x_{k+1})\| \leq \lambda^{j_k}\|F(x_{k'})\|[\Pi_{j=k'}^{k}(1 + \eta_j)] \leq \|F(x_{k'})\|\lambda^{\frac{k}{2} - k' - 1}e^{\eta}.$$

Thus, we obtain $\sum_{k=0}^{\infty} \|F(x_k)\| < \infty$. This together with (23) implies $\sum_{k=0}^{\infty} \|x_k - x^*\| < \infty$, and hence

$$\sum_{k=0}^{\infty} \|s_k\| < \infty.$$

Then, following from Lemma 4, one has

$$\delta_k = 0.$$

Consequently, according to (23), the sequence $\{x_k\}$ converges to x^* superlinearly. □

4. Numerical Experiments

In this section, we compare the SDBroyden method with Schubert's method [8]. We also compare the SDBroyden method with a direct Broyden method and Newton's method. All the methods are written in MATLAB R2018a and run in an iMac with 16G. The product $F'(x)s$ is computed by the automatic differentiation tool TOLMAB [24].

The testing problems are listed in Appendix A. The Jacobian matrices of the tested problems have different structures such as: diagonal (Problem 1, 2), tridiagonal (Problems 3, 4, 5, 6, 7, 8), block-diagonal (Problems 9, 10, 11), and special structure (Problem 12). The parameters in Algorithm 1 are specified as [19]

$$\epsilon = 10^{-5},\ \rho = 0.9,\ \sigma_1 = \sigma_2 = 0.001,\ \beta = 0.45,\ \eta_k = \frac{1}{(k+1)^2}.$$

For all the methods, we also stop the iteration if the number of iterations exceeds 200. We report the numerical performance of the above four methods in Tables 1–7 and Figures 1 and 2, where the meaning of each column is as follows:

Schubert:	Schubert's method;
SDBroyden:	sparse direct Broyden method with LF condition;
Pro	the number of the test problem;
Dim:	the dimension of the problem;
Ite	the total number of iterations;
Nfun:	the total number of function evaluations;
R:	Broyden's mean convergence rate;
Time(s):	CPU time in second;
Fail:	the stopping criterion was not satisfied.

(1) In the first set of our numerical experiments, we test the performance of the SDBroyden method and Schubert's method. When B_0 is chosen as unit matrix I, the results are listed in Tables 1 and 2, respectively. For SDBroyden method and Schubert's method, we compute the problems with dimensions ($n = 10, 20, 50, 100, 200, 500, 1000, 2000, 5000$, 10,000, 20,000, 50,000), but we select a subset of the dimensions ($n = 10, 100, 1000, 2000$, 10,000, 20,000, 50,000) to improve the readability of the corresponding tables. The two methods fail on two problems (3, 8). Considering the iteration counts, the SDBroyden method is more efficient than Schubert's method on seven problems (1, 2, 4, 5, 10, 11, 12), equivalent to Schubert's method on three problems (6, 7, 9). For the total number of function evaluations, the SDBroyden method has better performance on seven problems (1, 2, 4, 9, 10, 11, 12), while Schubert's method needs less function evaluations on one problem (5), and both methods are equivalent on two problems (6, 7). As for the Broyden's mean convergence rate, SDBroyden works well on seven problems (1, 2, 4, 6, 10, 11, 12), equal to Schubert's method on three problems (5, 7, 9). It can be seen that the SDBroyden method outperforms Schubert's method in iteration counts, function evaluation counts, and Broyden's mean convergence rate.

Table 1. Results of Schubert's method with $B_0 = I$.

Pro(Dim)	**10**	**100**	**1000**	**2000**	**10,000**	**20,000**	**50,000**
(1)Ite	6	6	6	6	6	6	6
(1)Nfun	7	7	7	7	7	7	7
(1)R	0.8915	1.1098	1.1326	1.1338	1.1349	1.1350	1.1351
(1)Time(s)	0.0600	0.0000	0.0200	0.0000	0.0400	0.0400	0.0800
(2)Ite	7	7	7	7	7	7	7
(2)Nfun	8	8	8	8	8	8	8
(2)R	1.0407	1.0831	1.0919	1.0924	1.0929	1.0929	1.0929
(2)Time(s)	0.0100	0.0000	0.0100	0.0000	0.0200	0.0400	0.0900
(4)Ite	12	12	12	13	16	14	14
(4)Nfun	20	20	21	23	26	21	22
(4)R	0.3429	0.3440	0.3496	0.3441	0.3215	0.4183	0.4178
(4)Time(s)	0.0100	0.0000	0.0000	0.0100	0.1300	0.2400	0.4100
(5)Ite	20	17	25	22	22	16	21
(5)Nfun	63	32	64	73	40	36	48
(5)R	0.1051	0.2288	0.1106	0.1052	0.1902	0.2461	0.1664
(5)Time(s)	0.0400	0.0000	0.0000	0.0000	0.1400	0.1600	0.3400
(6)Ite	4	3	2	2	2	2	1
(6)Nfun	5	4	3	3	3	3	2
(6)R	1.5553	2.5624	3.0388	3.4398	4.3713	4.7641	3.8427
(6)Time(s)	0.0600	0.0000	0.0100	0.0000	0.0200	0.0300	0.0300
(7)Ite	10	8	6	6	4	4	3
(7)Nfun	11	11	8	8	6	6	5
(7)R	0.4980	0.3935	0.5299	0.5489	0.5362	0.5613	0.5820
(7)Time(s)	0.0300	0.0000	0.0000	0.0000	0.0300	0.0700	0.1100
(9)Ite	4	4	4	4	4	4	4
(9)Nfun	7	7	7	7	7	7	7
(9)R	Inf	Inf	Inf	Inf	Inf	Inf	Inf
(9)Time(s)	0.0400	0.0000	0.0000	0.0000	0.5100	0.9400	1.5100
Dim	12	102	1002	2001	10,002	20,001	50,001
(10)Ite	4	4	5	5	5	5	5
(10)Nfun	6	6	7	7	7	7	7
(10)R	1.0563	1.0563	1.5586	1.5586	1.5586	1.5586	1.5586
(10)Time(s)	0.0100	0.0000	0.1100	0.1300	0.2600	0.4500	0.8000
Dim	12	102	1002	2001	10,002	20,001	50,001
(11)Ite	6	6	7	7	7	7	7
(11)Nfun	8	8	9	9	9	9	9
(11)R	0.9175	0.9175	1.4972	1.4972	1.4972	1.4972	1.4972
(11)Time(s)	0.0300	0.0100	0.3000	0.1000	0.5200	0.7900	1.7100
(12)Ite	5	5	5	5	5	6	6
(12)Nfun	6	6	6	6	6	7	7
(12)R	1.1312	1.1156	1.1142	1.1142	1.1141	1.5918	1.5985
(12)Time(s)	0.0400	0.0000	0.0000	0.0000	0.0200	0.0500	0.1200

Table 2. Results of the SDBroyden method with $B_0 = I$.

Pro(Dim)	10	100	1000	2000	10,000	20,000	50,000
(1)Ite	5	4	5	5	5	5	5
(1)Nfun	6	5	6	6	6	6	6
(1)R	1.6865	1.2113	2.1390	2.1415	2.1435	2.1437	2.1439
(1)Time(s)	0.0700	0.3600	1.4700	2.5200	13.2600	27.9500	88.0700
(2)Ite	5	5	5	5	5	6	6
(2)Nfun	6	6	6	6	6	7	7
(2)R	1.1023	1.1587	1.1705	1.1712	1.1718	1.9449	1.9450
(2)Time(s)	0.0300	0.2200	0.7800	1.7400	7.8200	20.2500	59.1100
(4)Ite	12	12	12	12	13	13	13
(4)Nfun	17	18	18	20	19	19	20
(4)R	0.4108	0.4253	0.4335	0.4085	0.4443	0.4747	0.4364
(4)Time(s)	0.2850	1.2600	11.2800	22.5000	139.3500	299.0400	834.3150
(5)Ite	16	16	20	18	19	20	
(5)Nfun	29	35	57	48	52	48	56
(5)R	0.2179	0.2396	0.1505	0.1674	0.1738	0.1664	0.1675
(5)Time(s)	0.1400	0.5100	4.6200	17.5600	80.7900	182.6200	375.0000
(6)Ite	3	2	2	2	2	2	1
(6)Nfun	4	3	3	3	3	3	2
(6)R	1.4566	2.4834	4.4685	5.0549	5.6778	5.1461	3.8427
(6)Time(s)	0.0500	0.0800	0.6200	1.1100	6.5800	13.2100	20.2600
(7)Ite	10	8	6	6	4	4	3
(7)Nfun	11	11	8	8	6	6	5
(7)R	0.4980	0.3935	0.5299	0.5489	0.5362	0.5613	0.5820
(7)Time(s)	0.1700	0.3800	2.0700	4.1000	14.1400	29.3900	69.4400
(9)Ite	4	4	4	4	4	4	4
(9)Nfun	6	6	6	6	6	6	6
(9)R	Inf	Inf	Inf	Inf	Inf	Inf	Inf
(9)Time(s)	0.0500	0.1450	0.5600	1.3000	13.6700	28.0700	84.4600
Dim	12	102	1002	2001	10,002	20,001	50,001
(10)Ite	3	3	3	3	4	4	4
(10)Nfun	5	5	5	5	6	6	6
(10)R	1.3420	1.3420	1.3420	1.3420	2.3852	2.3852	2.3852
(10)Time(s)	0.0600	0.1700	0.9000	1.8100	10.8000	21.5800	64.8000
Dim	12	102	1002	2001	10,002	20,001	50,001
(11)Ite	5	6	6	6	6	6	6
(11)Nfun	7	8	8	8	8	8	8
(11)R	1.0013	1.8209	1.8209	1.8209	1.8209	1.8209	1.8209
(11)Time(s)	0.0900	0.3000	2.1500	3.5300	18.8500	40.5600	109.5900
(12)Ite	4	4	4	4	4	4	4
(12)Nfun	5	5	5	5	5	5	5
(12)R	1.7679	1.7551	1.7540	1.7539	1.7538	1.7538	1.7538
(12)Time(s)	0.0500	0.1500	0.6700	1.2700	6.4500	13.7700	43.1500

When B_0 is chosen as the exact Jacobian matrix $F'(x_0)$, the results are given in Tables 3 and 4, respectively. The two methods solve the 12 problems successfully. The SDBroyden method needs fewer iterations than Schubert's method on seven problems (1, 2, 4, 5, 8, 10, 11), equal iterations with Schubert's method on five problems (3, 6, 7, 9, 12). For the total number of function evaluations, the SDBroyden method is more efficient than Schubert's method on six problems (1, 2, 4, 5, 8, 11) and equivalent to Schubert's method on six problems (3, 6, 7, 9, 10, 12). As for the Broyden's mean convergence rate, SDBroyden has better performance on nine problems (1, 2, 3, 4, 5, 8, 10, 11, 12) and equals Schubert's method on two problems (7, 9). The two methods are competitive on one problem (6). It also can be seen that the SDBroyden method outperforms Schubert's method in terms of number of iterations, number of function evaluations, and Broyden's mean convergence rate. Meanwhile, the CPU time of SDBroyden method is mostly more than that of Schubert's method.

Table 3. Results of Schubert's method with $B_0 = F'(x_0)$.

Pro(Dim)	**10**	**100**	**1000**	**2000**	**10,000**	**20,000**	**50,000**
(1)Ite	6	6	6	6	6	6	6
(1)Nfun	8	7	7	7	8	8	8
(1)R	0.8661	0.9523	0.9693	0.9703	0.9077	0.9077	0.9077
(1)Time(s)	0.0000	0.0000	0.0100	0.0000	0.0300	0.0400	0.0900
(2)Ite	6	6	6	6	6	6	6
(2)Nfun	7	7	7	7	7	7	7
(2)R	1.1690	1.1982	1.2024	1.2026	1.2028	1.2028	1.2028
(2)Time(s)	0.0500	0.0000	0.0000	0.0000	0.0400	0.0400	0.0800
(3)Ite	10	11	11	11	11	11	11
(3)Nfun	11	12	12	12	12	12	12
(3)R	0.6216	0.5988	0.6391	0.6515	0.6806	0.6931	0.7097
(3)Time(s)	0.0400	0.0000	0.0200	0.0000	0.0900	0.1400	0.2000
(4)Ite	13	23	23	29	21	30	27
(4)Nfun	25	44	45	58	44	81	72
(4)R	0.2705	0.1743	0.1820	0.1455	0.2054	0.1256	0.2367
(4)Time(s)	0.0500	0.0000	0.0100	0.0000	0.0600	0.1100	0.1800
(5)Ite	18	24	26	24	23	23	23
(5)Nfun	26	44	68	54	42	42	42
(5)R	0.2543	0.1425	0.1190	0.1420	0.1864	0.1900	0.1947
(5)Time(s)	0.0600	0.0000	0.0000	0.0000	0.2100	0.2400	0.5100
(6)Ite	5	3	2	2	2	2	2
(6)Nfun	6	4	3	3	3	3	3
(6)R	1.1562	1.8540	2.4963	2.8469	3.6620	4.0131	4.4773
(6)Time(s)	0.0400	0.0000	0.0100	0.0000	0.0100	0.0400	0.0700
(7)Ite	12	12	7	4	1	1	1
(7)Nfun	18	21	11	6	2	2	2
(7)R	0.2585	0.2158	0.3382	0.6401	1.7302	1.8807	2.0797
(7)Time(s)	0.0100	0.0000	0.0100	0.0100	0.0100	0.0400	0.0500
(8)Ite	8	8	8	7	7	7	7
(8)Nfun	10	11	10	9	10	10	19
(8)R	0.5898	0.5317	0.5779	0.5637	0.5789	0.5881	0.6023
(8)Time(s)	0.0300	0.0000	0.0000	0.0000	0.4200	1.3500	3.8000
(9)Ite	3	3	3	3	3	3	3
(9)Nfun	4	4	4	4	4	4	4
(9)R	4.0327	4.0327	4.0327	4.0327	4.0327	4.0327	4.0327
(9)Time(s)	0.0000	0.0100	0.0000	0.0100	0.0400	0.0700	0.2300
Dim	12	102	1002	2001	10,002	20,001	50,001
(10)Ite	9	10	10	10	11	11	11
(10)Nfun	10	11	11	11	12	12	12
(10)R	0.5520	0.5886	0.5886	0.5886	0.6701	0.6701	0.6701
(10)Time(s)	0.0200	0.0200	0.2800	0.2600	0.7400	1.1800	2.4700
Dim	12	102	1002	2001	10,002	20,001	50,001
(11)Ite	5	6	6	6	6	6	6
(11)Nfun	6	7	7	7	7	7	7
(11)R	1.1416	1.7664	1.7664	1.7664	1.7664	1.7664	1.7664
(11)Time(s)	0.0300	0.0300	0.2000	0.2200	0.4300	0.6500	1.3500
(12)Ite	8	8	8	8	8	8	8
(12)Nfun	9	9	9	9	9	9	10
(12)R	0.5909	0.6449	0.7003	0.7170	0.7558	0.7725	0.7210
(12)Time(s)	0.0200	0.0000	0.0000	0.0100	0.0600	0.1100	0.1800

Table 4. Results of the SDBroyden method with $B_0 = F'(x_0)$.

Pro(Dim)	**10**	**100**	**1000**	**2000**	**10,000**	**20,000**	**50,000**
(1)Ite	4	5	5	5	5	5	5
(1)Nfun	6	6	6	6	7	7	7
(1)R	0.9211	1.6975	1.7287	1.7304	1.6431	1.6431	1.6431
(1)Time(s)	0.0300	0.2200	1.6600	2.6400	14.2100	29.4400	88.3100
(2)Ite	4	4	4	5	5	5	5
(2)Nfun	5	5	5	6	6	6	6
(2)R	1.2544	1.2872	1.2916	2.1183	2.1187	2.1187	2.1187
(2)Time(s)	0.0400	0.1300	0.6600	1.6900	7.7100	16.1100	48.0500
(3)Ite	11	11	11	11	11	11	11
(3)Nfun	12	12	12	12	12	12	12
(3)R	0.5609	0.5642	0.6045	0.6170	0.6460	0.6586	0.6751
(3)Time(s)	0.1900	0.6100	5.5200	10.8000	55.6500	112.3100	307.9200
(4)Ite	13	17	17	18	20	23	18
(4)Nfun	24	28	28	35	38	59	56
(4)R	0.2867	0.2688	0.2707	0.2218	0.6251	0.5620	0.6294
(4)Time(s)	0.1500	0.8900	5.8100	12.4900	66.7200	142.7600	402.8600
(5)Ite	23	21	22	20	20	20	20
(5)Nfun	46	40	35	34	34	34	34
(5)R	0.1996	0.1651	0.1957	0.2207	0.2309	0.2354	0.2412
(5)Time(s)	0.3500	1.7100	13.8200	26.4100	143.4000	293.5900	341.0000
(6)Ite	4	3	2	2	2	2	2
(6)Nfun	5	4	3	3	3	3	3
(6)R	1.1204	2.0735	2.4895	2.8401	3.6550	4.0062	4.4704
(6)Time(s)	0.1000	0.1800	1.1100	1.6500	8.7100	20.2800	53.0700
(7)Ite	12	12	7	4	1	1	1
(7)Nfun	18	21	11	6	2	2	2
(7)R	0.2585	0.2158	0.3382	0.6401	1.7302	1.8807	2.0797
(7)Time(s)	0.2100	0.8900	3.9500	4.4100	5.9700	12.5000	32.9300
(8)Ite	11	7	6	6	6	6	6
(8)Nfun	14	9	8	8	9	9	9
(8)R	0.4237	0.5767	0.6337	0.8068	0.7790	0.7811	0.7995
(8)Time(s)	0.1900	0.4100	2.8100	6.1500	13.3900	24.0600	48.4500
(9)Ite	3	3	3	3	3	3	3
(9)Nfun	4	4	4	4	4	4	4
(9)R	Inf	Inf	Inf	Inf	Inf	Inf	Inf
(9)Time(s)	0.0300	0.1400	0.5400	1.3100	5.7400	13.2200	37.1600
Dim	12	102	1002	2001	10,002	20,001	50,001
(10)Ite	8	9	9	9	9	9	9
(10)Nfun	10	11	11	11	11	11	11
(10)R	0.5616	0.6068	0.6068	0.6068	0.6309	0.6374	0.6438
(10)Time(s)	0.1300	0.4600	3.5500	6.6100	39.4200	82.5100	242.4400
Dim	12	102	1002	2001	10,002	20,001	50,001
(11)Ite	4	5	5	5	5	5	5
(11)Nfun	5	6	6	6	6	6	6
(11)R	1.3736	2.3204	2.3204	2.3204	2.3204	2.3204	2.3204
(11)Time(s)	0.0600	0.2300	1.8600	2.9500	15.8600	33.6900	92.8300
(12)Ite	8	8	8	8	8	8	7
(12)Nfun	9	9	9	9	9	9	9
(12)R	0.6704	0.7239	0.7793	0.7960	0.8348	0.8515	0.7721
(12)Time(s)	0.1100	0.2900	2.1600	3.7500	19.3700	41.5900	106.5800

Performance ration [25] is used to compare the numerical performance. For given solvers set S and problems set P, let $t_{p,s}$ be the number of iterations, the number of function evaluations or others, required to solve problem p by solver s. Then, define the performance ration as

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,q} : q \in S\}},$$

whose distribution function is defined as

$$\rho_s(t) = \frac{1}{N_p}\text{size}\{p \in P : r_{p,s} \leq t\},$$

where N_p is the number of problems in the set P. Thus, $\rho_s : R \to [0, 1]$ was the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ was within a factor $t \in R$ of the best possible ratio. According to the definition of performance profiles, we can see that the top curve corresponds to the best solver.

In Figure 1, the performance of the two methods: the SDBroyden method and Schubert's method, relative to the number of iterations, and the number of function evaluations are evaluated. Figure 1 indicates that SDBroyden has better performance than Schubert's method on the number of iterations and number of function evaluations.

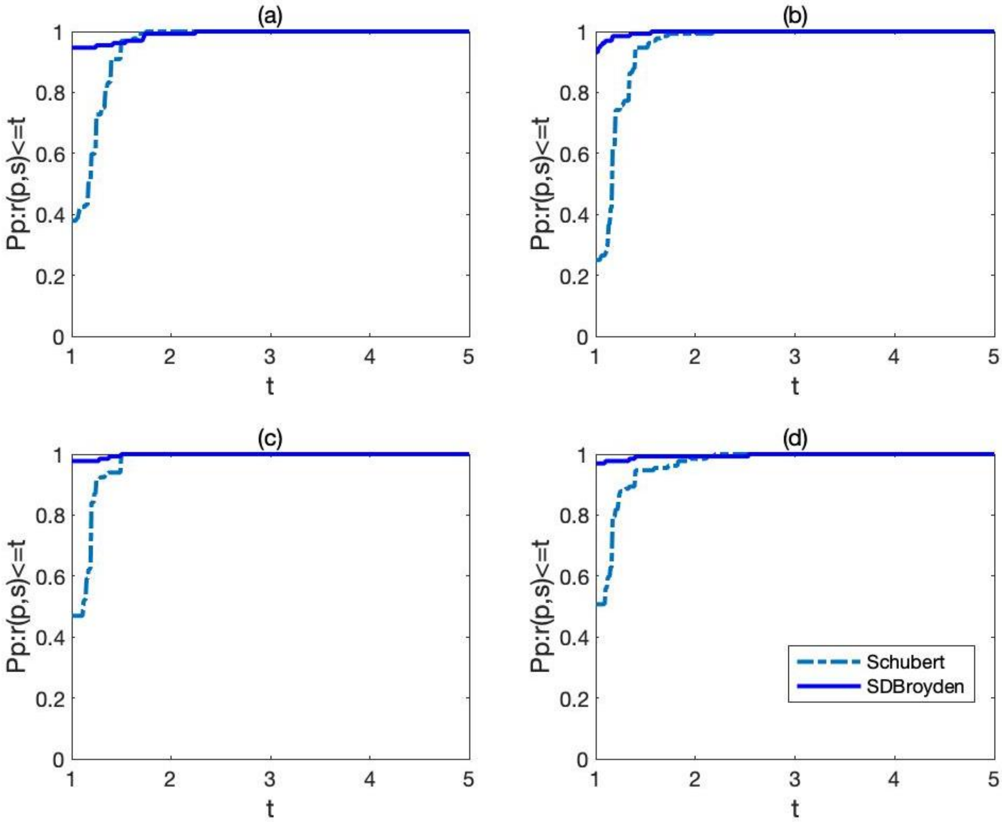

Figure 1. Performance profiles for SDBroyden and Schubert: (**a**) results comparison on the number of iterations with $B_0 = I$; (**b**) results comparison on the number of function evaluations with $B_0 = I$; (**c**) results comparison on the number of iterations with $B_0 = F'(x_0)$; (**d**) results comparison on the number of function evaluations with $B_0 = F'(x_0)$.

(2) In the second set of numerical experiments, we compare the SDBroyden method with the direct Broyden quasi-Newton method (DBQN). We give the results of the DBQN method with $B_0 = I$ in Table 5. The DBQN method fails on four problems (3, 5, 8, 9). For the number of iterations and number of function evaluations, the SDBroyden method needs less iterations on five problems (2, 4, 6, 7, 11) and equals DBQN on three problems (1, 10, 12). For the Broyden's mean convergence rate, the SDBroyden method performs better on

five problems (2, 4, 6, 7, 11), equals DBQN on two problems (1, 10), and works badly on one problem (12).

Table 5. Results of the DBQN method with $B_0 = I$.

Pro(Dim)	10	20	50	100	200	500	1000
(1)Ite	5	4	4	4	4	5	5
(1)Nfun	6	5	5	5	5	6	6
(1)R	1.6865	1.0998	1.1828	1.2113	1.2258	2.1341	2.1390
(1)Time(s)	0.2900	0.1900	0.1500	0.5400	0.5600	3.1000	19.3500
(2)Ite	5	5	5	5	5	6	6
(2)Nfun	6	6	6	6	6	7	7
(2)R	0.9943	1.0107	1.0229	1.0274	1.0297	1.2624	1.2631
(2)Time(s)	0.1000	0.1300	0.1600	0.2100	0.4500	3.1700	21.8400
(4)Ite	24	33	53	59	58	73	95
(4)Nfun	42	68	117	118	120	187	318
(4)R	0.1630	0.0963	0.0606	0.0585	0.0587	0.0383	0.0233
(4)Time(s)	1.0800	1.9000	6.3600	12.6900	24.1200	97.9500	510.2200
(6)Ite	6	5	3	3	2	2	2
(6)Nfun	7	6	4	4	3	3	3
(6)R	0.8516	1.1498	1.6230	2.0975	2.4436	3.0384	3.4893
(6)Time(s)	0.2000	0.2700	0.1500	0.7800	0.4200	1.5500	8.1900
(7)Ite	15	23	20	24	14	12	11
(7)Nfun	22	32	30	27	29	22	20
(7)R	0.2285	0.1356	0.1421	0.1510	0.1401	0.1741	0.1863
(7)Time(s)	0.5700	0.6900	1.3200	3.0500	3.9500	10.9700	47.1200
Dim	12	21	51	102	201	501	1002
(10)Ite	3	3	3	3	3	3	3
(10)Nfun	5	5	5	5	5	5	5
(10)R	1.3420	1.3420	1.3420	1.3420	1.3420	1.3420	1.3420
(10)Time(s)	0.1500	0.1800	0.1300	0.2600	0.6400	2.0000	10.9100
Dim	12	21	51	102	201	501	1002
(11)Ite	7	7	7	7	8	8	8
(11)Nfun	13	13	13	13	14	14	14
(11)R	0.5532	0.5532	0.5532	0.5532	0.6535	0.6224	0.5851
(11)Time(s)	0.2200	0.3400	0.2800	0.5700	1.3100	4.8700	33.8700
(12)Ite	4	4	4	4	4	4	4
(12)Nfun	5	5	5	5	5	5	5
(12)R	1.9415	1.8389	1.7860	1.7696	1.7617	1.7569	1.7554
(12)Time(s)	0.1400	0.1700	0.1500	0.2500	0.5000	2.1700	13.6100

The results of the DBQN method with $B_0 = F'(x_0)$ are listed in Table 6. The DBQN method fails on one problem (5). For the number of iterations, SDBroyden is better than the DBQN method on seven problems (2, 4, 6, 8, 10, 11, 12), equivalent to the DBQN method on three problems (1, 3, 9). At the same time, DBQN performs well on one problem (7). For the number of function evaluations and Broyden's mean convergence rate, SDBroyden is excellent on six problems (2, 4, 6, 8, 11, 12), while the DBQN method works well on one problem (10). The two methods coincide with each other on three problems (3, 9, 10).

Table 6. Results of the DBQN method with $B_0 = F'(x_0)$.

Pro(Dim)	10	20	50	100	200	500	1000
(1)Ite	4	5	5	5	5	5	5
(1)Nfun	6	6	6	6	6	6	6
(1)R	0.9211	1.5530	1.6623	1.6975	1.7149	1.7252	1.7287
(1)Time(s)	0.0900	0.2200	0.4800	0.5100	0.6700	3.1600	20.2400
(2)Ite	11	11	12	12	12	13	13
(2)Nfun	12	12	13	13	13	14	14
(2)R	0.4726	0.4677	0.4819	0.4822	0.4827	0.4807	0.4808
(2)Time(s)	0.2000	0.3300	0.3200	0.5100	1.1300	6.7900	47.0600
(3)Ite	10	11	11	11	11	11	11
(3)Nfun	11	12	12	12	12	12	12
(3)R	0.5629	0.5496	0.5542	0.5641	0.5754	0.5912	0.6035
(3)Time(s)	0.4800	0.6900	0.7200	1.1800	2.2700	9.2200	45.4800
(4)Ite	16	25	32	29	35	34	33
(4)Nfun	23	40	56	58	64	83	67
(4)R	0.3164	0.1641	0.1241	0.1219	0.1142	0.0897	0.1128
(4)Time(s)	0.7800	1.2400	4.0200	6.7600	15.7000	49.1800	180.5200
(6)Ite	5	5	4	4	3	3	3
(6)Nfun	6	6	5	5	4	4	4
(6)R	1.0196	1.1784	1.5102	1.9224	1.8193	2.2563	2.5914
(5)Time(s)	0.1000	0.1700	0.3200	0.4400	0.6200	2.2700	12.2300
(7)Ite	3	3	3	3	3	1	1
(7)Nfun	4	4	4	4	4	2	2
(7)R	1.5674	1.8740	2.2760	2.5987	2.9294	2.0320	2.2588
(7)Time(s)	0.1100	0.1000	0.3100	0.3900	0.7100	0.8600	4.2900
(8)Ite	14	14	18	23	20	23	24
(8)Nfun	20	23	38	59	44	76	80
(8)R	0.2827	0.2437	0.1454	0.0953	0.1199	0.0731	0.0670
(8)Time(s)	0.6300	1.0300	1.6100	2.7100	5.0500	20.8500	102.1800
Dim	12	21	51	102	201	501	1002
(9)Ite	3	3	3	3	3	3	3
(9)Nfun	4	4	4	4	4	4	4
(9)R	3.8134	3.7605	3.7383	3.5184	3.4381	3.6076	3.5910
(9)Time(s)	0.0300	0.0600	0.2200	0.2200	0.3700	1.4300	9.5300
Dim	12	21	51	102	201	501	1002
(10)Ite	9	9	9	9	9	9	9
(10)Nfun	10	10	10	10	10	10	10
(10)R	0.7420	0.7420	0.7420	0.7420	0.7420	0.7420	0.7420
(10)Time(s)	0.3200	0.2500	0.4300	1.4000	1.6200	6.0500	32.8000
Dim	12	21	51	102	201	501	1002
(11)Ite	5	5	5	5	5	5	5
(11)Nfun	6	6	6	6	6	6	6
(11)R	1.3488	1.3488	1.3488	1.3488	1.3488	1.3488	1.3488
(11)Time(s)	0.1300	0.1300	0.1900	0.4200	0.7500	3.0700	20.7800
(12)Ite	11	13	15	15	15	15	15
(12)Nfun	12	14	16	16	16	16	16
(12)R	0.4607	0.3912	0.3475	0.3534	0.3610	0.3720	0.3808
(12)Time(s)	0.1500	0.3900	0.5600	0.9900	1.9300	8.0200	55.5800

In Figure 2, we also give the comparison of the SDBroyden method and DBQN method relative to the number of iterations and number of function evaluations. It can be seen that the top curve corresponds to the SDBroyden method. This means that the SDBroyden method has satisfactory performance in terms of number of iterations and number of function evaluations when compared with its dense version.

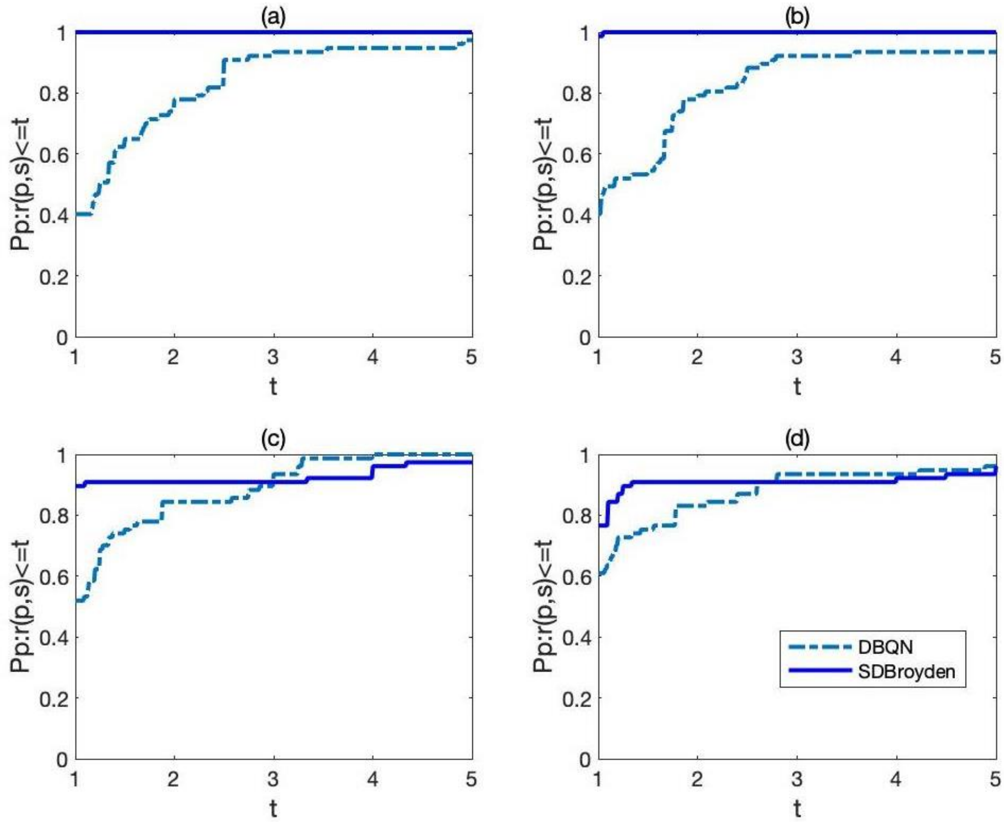

Figure 2. Performance profiles of SDBroyden and DBQN (**a**) results comparison on the number of iterations with $B_0 = I$; (**b**) results comparison on the number of function evaluations with $B_0 = I$; (**c**) results comparison on the number of iterations with $B_0 = F'(x_0)$; (**d**) results comparison on the number of function evaluations with $B_0 = F'(x_0)$.

(3) In the third set of our numerical experiments, we compare the SDBroyden method with Newton's method, where the results are listed in Table 7. Newton's method fails on three problems (5, 8, 10). One can see that the SDBroyden method requires slightly more iterations than Newton's method in most tests and has no significant advantages in the number of iterations, number of function evaluations, and Broyden's mean convergence rate. However, the CPU time for Newton's method is much higher than that of the SDBroyden method. Moreover, the CPU time of Newton's method increases significantly faster than that of the quasi-Newton methods. Thus, the SDBroyden method can be applied to solve large-scale nonlinear equations.

Table 7. Results of the Newton's method.

Pro(Dim)	10	20	50	100	200	500	1000
(1)Ite	4	5	5	5	5	5	5
(1)Nfun	6	6	6	6	6	6	6
(1)R	0.9211	1.5530	1.6623	1.6975	1.7149	1.7252	1.7287
(1)Time(s)	0.0000	0.0000	0.0100	0.0700	0.1900	1.9100	16.8700
(2)Ite	4	4	4	4	4	4	4
(2)Nfun	5	5	5	5	5	5	5
(2)R	1.2544	1.2704	1.2826	1.2872	1.2896	1.2911	1.2916
(2)Time(s)	0.0100	0.0000	0.0100	0.0100	0.1500	2.2600	13.3200
(3)Ite	4	4	4	4	4	4	5
(3)Nfun	5	5	5	5	5	5	6
(3)R	1.2884	1.3061	1.3333	1.3542	1.3729	1.3913	2.2609
(3)Time(s)	0.0000	0.0000	0.0100	0.0100	0.1900	2.1300	17.8300
(4)Ite	16	17	18	18	19	19	20
(4)Nfun	17	18	19	19	20	20	21
(4)R	0.3721	0.3649	0.3625	0.3617	0.3620	0.3618	0.3623
(4)Time(s)	0.0200	0.0100	0.0100	0.1100	0.8500	7.8600	67.7600
(6)Ite	15	12	8	6	5	4	3
(6)Nfun	16	13	9	7	6	5	4
(6)R	0.3600	0.4488	0.6793	0.9078	1.1567	1.4961	1.7423
(6)Time(s)	0.0300	0.0000	0.0100	0.0600	0.2000	1.4900	9.9400
(7)Ite	22	18	12	8	4	1	1
(7)Nfun	23	19	13	9	5	2	2
(7)R	0.1919	0.2323	0.3217	0.4503	0.7825	2.0320	2.2588
(7)Time(s)	0.0300	0.0000	0.0200	0.0600	0.1200	0.3800	3.3100
(9)Ite	2	2	2	2	2	2	2
(9)Nfun	3	3	3	3	3	3	3
(9)R	Inf	Inf	Inf	Inf	Inf	Inf	Inf
(9)Time(s)	0.0000	0.0000	0.0100	0.0100	0.1000	0.7000	5.8800
Dim	12	21	51	102	201	501	1002
(11)Ite	3	3	3	3	3	3	3
(11)Nfun	4	4	4	4	4	4	4
(11)R	2.0137	2.0137	2.0137	2.0137	2.0137	2.0137	2.0137
(11)Time(s)	0.0100	0.0000	0.0100	0.0200	0.1100	1.0600	11.1200
(12)Ite	4	4	4	4	4	4	4
(12)Nfun	5	5	5	5	5	5	5
(12)R	1.4972	1.4718	1.4581	1.4541	1.4523	1.4511	1.4508
(12)Time(s)	0.0100	0.0000	0.0100	0.0300	0.1600	1.2900	12.0100

5. Conclusions

We have developed a sparse direct Broyden quasi-Newton method for solving large-scale nonlinear equations, which is the sparse case of the direct Broyden method and is an extension of Broyden's method. The method approximates the Jacobian matrix by least change updating and satisfies the sparsity condition and direct tangent condition simultaneously. We show that the method is locally and superlinearly convergent. Combined with a nonmonotone line search, we also establish the global and superlinear convergence. In particular, the unit step length is essentially accepted. Our numerical results show that the proposed method is effective and competitive for sparse nonlinear equations.

Author Contributions: Conceptualization, H.C.; methodology, H.C. and J.H.; software, H.C.; formal analysis, H.C.; writing—original draft preparation, H.C. and J.H.; writing—review and editing, H.C. and J.H. All authors have read and agreed to the published version of the manuscript.

Funding: The work is supported by the National Natural Science Foundation of China, Grant No. 11701577; the Natural Science Foundation of Hunan Province, China, Grant No. 2019JJ51002, 2020JJ5960; the Scientific Research Foundation of Hunan Provincial Education Department, China, Grant No. 18C0253; and the Natural Science Foundation of Shaanxi Province, China, Grant No. 2022JQ006.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The date used to support the research plan and all the computer codes used in this study are available from the corresponding author upon request.

Acknowledgments: The authors would like to thank the four anonymous referees for their careful reading of this paper and their comments to improve the quality of this paper. The authors also would like to thank the corresponding editor for providing insightful suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this section, we list the test problems with initial guess x_0

$$F(x) = (f_1(x), f_2(x) \dots, f_n(x))^T,$$

where references [26–35] are cited in the Appendix A.

Problem 1. Logarithmic function [26]

$$\begin{aligned} F_i(x) &= \ln(x_i + 1) - \frac{x_i}{n},\ i = 1, 2, \dots, n. \\ x_0 &= (1, 1, \dots, 1)^T. \end{aligned}$$

Problem 2. Strictly convex function [27]
$F(x)$ is the gradient of $h(x) = \sum_{i=1}^{n}(e^{x_i} - x_i)$.

$$\begin{aligned} F_i(x) &= e^{x_i} - 1,\ i = 1, 2, \dots, n. \\ x_0 &= \left(\frac{1}{n}, \frac{2}{n}, \dots, 1\right)^T. \end{aligned}$$

Problem 3. Broyden Tridiagonal function [28]

$$\begin{aligned} F_1(x) &= (3 - 0.5x_1)x_1 - 2x_2 + 1, \\ F_i(x) &= (3 - 0.5x_i)x_i - x_{i-1} - 2x_{i+1} + 1, i = 2, \dots, n-1, \\ F_n(x) &= (3 - 0.5x_n)x_n - x_{n-1} + 1. \\ x_0 &= (-3, -3, \dots, -3)^T. \end{aligned}$$

Problem 4. Trigexp function [28]

$$\begin{aligned} F_1(x) &= 3x_1^3 + 2x_2 - 5 + \sin(x_1 - x_2)\sin(x_1 + x_2), \\ F_i(x) &= -x_{i-1}e^{(x_{i-1} - x_i)} + x_i(4 + 3x_i^2) + 2x_{i+1} \\ &\quad + \sin(x_i - x_{i+1})\sin(x_i + x_{i+1}) - 8,\ i = 2, \dots, n-1, \\ F_n(x) &= -x_{n-1}e^{(x_{n-1} - x_n)} + 4x_n - 3. \\ x_0 &= (0, 0, \dots, 0)^T. \end{aligned}$$

Problem 5. Tridiagonal system [29]

$$\begin{aligned}
F_1(x) &= 4(x_1 - x_2^2), \\
F_i(x) &= 8x_i(x_i^2 - x_{i-1}) - 2(1 - x_i) + 4(x_i - x_{i+1}^2),\ i = 2, \dots, n-1 \\
F_n(x) &= 8x_n(x_n^2 - x_{n-1}) - 2(1 - x_n). \\
x_0 &= (12, \dots, 12)^T.
\end{aligned}$$

Problem 6. Tridiagonal exponential problem [30]

$$\begin{aligned}
F_1(x) &= x_1 - \exp(\cos(h(x_1 + x_2))), \\
F_i(x) &= x_i - \exp(\cos(h(x_{i-1} + x_i + x_{i+1}))),\ i = 2, \dots, n-1, \\
F_n(x) &= x_n - \exp(\cos(h(x_{n-1} + x_n))), \\
h &= 1/(n+1). \\
x_0 &= (1.5, 1.5, \dots, 1.5)^T.
\end{aligned}$$

Problem 7. Discrete boundary value problem [31]

$$\begin{aligned}
F_1(x) &= 2x_1 + 0.5h^2(x_1 + h)^3 - x_2, \\
F_i(x) &= 2x_i + 0.5h^2(x_i + hi)^3 - x_{i-1} + x_{i+1}, \quad i = 2, \dots, n-1, \\
F_n(x) &= 2x_n + 0.5h^2(x_n + hn)^3 - x_{n-1}, \\
h &= 1/(n+1). \\
x_0 &= (h(h-1), h(2h-1), \dots, h(nh-1))^T.
\end{aligned}$$

Problem 8. Troesch problem [32]

$$\begin{aligned}
F_1(x) &= 2x_1 + \rho h^2 \sinh(\rho x_1) - x_2, \\
F_i(x) &= 2x_i + \rho h^2 \sinh(\rho x_i) - x_{i-1} - x_{i+1}, \quad i = 2, \dots, n-1, \\
F_n(x) &= 2x_n + \rho h^2 \sinh(\rho x_n) - x_{n-1}, \\
\rho &= 10, h = 1/(n+1). \\
x_0 &= (0, 0, \dots, 0)^T.
\end{aligned}$$

Problem 9. Extended Rosenbrock function (n is even) [33]

$$\begin{aligned}
F_{2i-1}(x) &= 10(x_{2i} - x_{2i-1}^2), \\
F_{2i}(x) &= 1 - x_{2i-1},\ i = 1, 2, \dots, n/2. \\
x_0 &= (5, 1, \dots, 5, 1)^T.
\end{aligned}$$

Problem 10. Problem 21 in [26] (n is multiple of 3)

$$\begin{aligned}
F_{3i-2}(x) &= x_{3i-2}x_{3i-1} - x_{3i}^2 - 1, \\
F_{3i-1}(x) &= x_{3i-2}x_{3i-1}x_{3i} - x_{3i-2}^2 + x_{3i-1}^2 - 2, \\
F_{3i}(x) &= e^{-x_{3i-2}} - e^{-x_{3i-1}},\ i = 1, 2, \dots, n/3. \\
x_0 &= (1, 1, \cdots, 1)^T.
\end{aligned}$$

Problem 11. Tridimensional valley function (n is multiple of 3) [34]

$$
\begin{aligned}
F_{3i-2}(x) &= (c_2 x_{3i-2}^3 + c_1 x_{3i-2}) \exp\left(\frac{-x_{3i-2}^2}{100}\right) - 1, \\
F_{3i-1}(x) &= 10(\sin(x_{3i-2}) - x_{3i-1}), \\
F_{3i}(x) &= 10(\cos(x_{3i-2}) - x_{3i}),\ i = 1, 2, \ldots, n/3, \\
c_1 &= 1.003344481605351, \\
c_2 &= -3.344481605351171 \times 10^{-3}. \\
x_0 &= (2, 1, 2, \ldots, 2, 1, 2)^T.
\end{aligned}
$$

Problem 12. [35]

$$
\begin{aligned}
F_1(x) &= x_1, \\
F_i(x) &= \cos(x_{k-1}) + x_k - 1,\ i = 2, \ldots, n. \\
x_0 &= (0.5, 0.5, \ldots, 0.5)^T.
\end{aligned}
$$

References

1. Yuan, Y.X. Recent advances in numerical methods for nonlinear equations and nonlinear least squares. *Numer. Algebra Control Optim.* **2011**, *1*, 15–34. [CrossRef]
2. Sun, W.; Yuan, Y.X. *Optimization Theory and Methods: Nonlinear Programming*; Springer Science & Business Media: New York, NY, USA, 2006.
3. Wright, S.; Nocedal, J. *Numerical Optimization*; Springer Science: New York, NY, USA, 1999; Volume 35, p. 7.
4. Fletcher, R. *Practical Methods of Optimization*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
5. Broyden, C.G. A class of methods for solving nonlinear simultaneous equations. *Math. Comput.* **1965**, *19*, 577–593. [CrossRef]
6. Broyden, C.G.; Dennis, J.E., Jr.; Moré, J.J. On the local and superlinear convergence of quasi-Newton methods. *IMA J. Appl. Math.* **1973**, *12*, 223–245. [CrossRef]
7. Dennis, J.E.; Moré, J.J. Quasi–Newton methods, motivation and theory. *SIAM Rev.* **1977**, *19*, 46–89. [CrossRef]
8. Schubert, L.K. Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian. *Math. Comput.* **1970**, *24*, 27–30. [CrossRef]
9. Toint, P.L. On sparse and symmetric matrix updating subject to a linear equation. *Math. Comput.* **1977**, *31*, 954–961. [CrossRef]
10. Nocedal, J. Updating quasi-Newton matrices with limited storage. *Math. Comput.* **1980**, *35*, 773–782. [CrossRef]
11. van de Rotten, B.; Lunel, S.V. A limited memory Broyden method to solve high-dimensional systems of nonlinear equations. In Proceedings of the International Conference on Differential Equations, Hasselt, Belgium, 22–26 July 2003; pp. 196–201.
12. Griewank, A.; Toint, P.L. Partitioned variable metric updates for large structured optimization problems. *Numer. Math.* **1982**, *39*, 119–137. [CrossRef]
13. Griewank, A.; Toint, P.L. Local convergence analysis for partitioned quasi-Newton updates. *Numer. Math.* **1982**, *39*, 429–448.
14. Cao, H.P.; Li, D.H. Partitioned quasi-Newton methods for sparse nonlinear equations. *Comput. Optim. Appl.* **2017**, *66*, 481–505. [CrossRef]
15. Leong, W.J.; Hassan, M.A.; Yusuf, M.W. A matrix-free quasi-Newton method for solving large-scale nonlinear systems. *Comput. Math. Appl.* **2011**, *62*, 2354–2363. [CrossRef]
16. Li, D.H.; Wang, X.; Huang, J. Diagonal BFGS updates and applications to the limited memory BFGS method. *Comput. Optim. Appl.* **2022**, *81*, 829–856. [CrossRef]
17. Martínez, J.M. A quasi-Newton method with modification of one column per iteration. *Computing* **1984**, *33*, 353–362. [CrossRef]
18. Griewank, A. The "global" convergence of Broyden-like methods with suitable line search. *ANZIAM J.* **1986**, *28*, 75–92. [CrossRef]
19. Li, D.H.; Fukushima, M. A derivative-free line search and global convergence of Broyden-like method for nonlinear equations. *Optim. Methods Softw.* **2000**, *13*, 181–201. [CrossRef]
20. Griewank, A.; Walther, A. On constrained optimization by adjoint based quasi-Newton methods. *Optim. Methods Softw.* **2002**, *17*, 869–889. [CrossRef]
21. Schlenkrich, S.; Griewank, A.; Walther, A. On the local convergence of adjoint Broyden methods. *Math. Program.* **2010**, *121*, 221–247. [CrossRef]
22. Marwil, E. Convergence results for Schubert's method for solving sparse nonlinear equations. *SIAM J. Numer. Anal.* **1979**, *16*, 588–604. [CrossRef]
23. Powell, M.J.D. A hybrid method for nonlinear equations. In *Numerical Methods for Nonlinear Algebraic Equations*; Gordon and Breach: London, UK, 1970.

24. Forth, S.A.; Edvall, M.M. *User Guide for MAD-MATLAB Automatic Differentiation Toolbox TOMLAB/MAD*; Version 1.1 The Forward Mode; TOMLAB Optimisation Inc.: San Diego, CA, USA, 2007; p. 92101.
25. Dolan, E.D.; Moré, J.J. Benchmarking optimization software with performance profiles. *Math. Program.* **2002**, *91*, 201–213. [CrossRef]
26. La Cruz, W.; Martínez, J.; Raydan, M. Spectral residual method without gradient information for solving large-scale nonlinear systems of equations. *Math. Comput.* **2006**, *75*, 1429–1448. [CrossRef]
27. Raydan, M. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM J. Numer. Anal.* **1997**, *7*, 26–33. [CrossRef]
28. Gomes-Ruggiero, M.A.; Martínez, J.M.; Moretti, A.C. Comparing algorithms for solving sparse nonlinear systems of equations. *SIAM J. Sci. Comput.* **1992**, *13*, 459–483. [CrossRef]
29. Li, G. Successive column correction algorithms for solving sparse nonlinear systems of equations. *Math. Program.* **1989**, *43*, 187–207. [CrossRef]
30. Bing, Y.; Lin, G. An efficient implementation of Merrill's method for sparse or partially separable systems of nonlinear equations. *SIAM J. Optim.* **1991**, *1*, 206–221. [CrossRef]
31. Moré, J.J.; Garbow, B.S.; Hillstrom, K.E. Testing unconstrained optimization software. *ACM Trans. Math. Softw. (TOMS)* **1981**, *7*, 17–41. [CrossRef]
32. Roberts, S.M.; Shipman, J.S. On the closed form solution of Troesch's problem. *J. Comput. Phys.* **1976**, *21*, 291–304. [CrossRef]
33. Gasparo, M.G. A nonmonotone hybrid method for nonlinear systems. *Optim. Methods Softw.***2000**, *13*, 79–94. [CrossRef]
34. Friedlander, A.; Gomes-Ruggiero, M.A.; Kozakevich, D.N.; Mario Martínez, J.; Augusta Santos, S. Solving nonlinear systems of equations by means of quasi-neston methods with a nonmonotone stratgy. *Optim. Methods Softw.* **1997**, *8*, 25–51. [CrossRef]
35. Luksan, L.; Matonoha, C.; Vlcek, J. *Problems for Nonlinear Least Squares and Nonlinear Equations*; Technical Report 1259; Institute of Computer Science, Academy of Sciences of the Czech Republic: Prague, Czech Republic, 2018.