Research Article

An Improved Diagonal Jacobian Approximation via a New Quasi-Cauchy Condition for Solving Large-Scale Systems of Nonlinear Equations

Mohammed Yusuf Waziri¹,² and Zanariah Abdul Majid¹,³

¹Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia
²Department of Mathematical Sciences, Faculty of Science, Bayero University, Kano PMB 3011, Nigeria
³Institute for Mathematical Research, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

Correspondence should be addressed to Mohammed Yusuf Waziri; mywaziri@gmail.com

Received 8 August 2012; Revised 14 December 2012; Accepted 15 December 2012

Academic Editor: Turgut Özış

Copyright © 2013 M. Y. Waziri and Z. Abdul Majid. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We present a new diagonal quasi-Newton update with an improved diagonal Jacobian approximation for solving large-scale systems of nonlinear equations. In this approach, the Jacobian approximation is derived based on the quasi-Cauchy condition. The anticipation has been to further improve the performance of diagonal updating, by modifying the quasi-Cauchy relations so as to carry some additional information from the functions. The effectiveness of our proposed scheme is appraised through numerical comparison with some well-known Newton-like methods.

1. Introduction

Let us consider the systems of nonlinear equations

\[ F(x) = 0, \]

where \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a nonlinear mapping. Often, the mapping \( F \) is assumed to be satisfying the following assumptions:

(A1) there exists an \( x^* \in \mathbb{R}^n \) s.t \( F(x^*) = 0 \);

(A2) \( F \) is a continuously differentiable mapping in a neighborhood of \( x^* \);

(A3) \( F'(x^*) \) is invertible.

The well-known method for finding the solution to (1) is the classical Newton's method which generates a sequence of iterates \( \{x_k\} \) from a given initial point \( x_0 \) via

\[ x_{k+1} = x_k - \left( F'(x_k) \right)^{-1} F(x_k), \]

where \( k = 0, 1, 2 \ldots \) The attractive features of this method are rapid convergence and being easy to implement. Nevertheless, Newton's method requires the computation of the matrix entails the first-order derivatives of the systems. In practice, computations of some functions derivatives are quite costly, and sometimes they are not available or could not be done precisely. In this case, Newton's method cannot be applied directly.

Moreover, some substantial efforts have been made by numerous researchers in order to eliminate the well-known shortcomings of Newton's method for solving systems of nonlinear equations, particularly large-scale systems (see, e.g., [1, 2]). Notwithstanding, most of these modifications of Newton's method still have some shortfalls as Newton's counterpart. For example, Broyden’s method and Chord Newton’s method need to store an \( n \times n \) matrix, and their floating points operations, are \( O(n^2) \), respectively.

To tackle these disadvantages, a diagonally Newton’s method has been suggested by Leong et al. [3] and showed that their updating formula is significantly cheaper than Newton’s method and some of its variants. Based on this fact, it is pleasing to present an approach which will improve further the diagonal Jacobian approximation, as well as reducing the computational cost, floating points operations and number of iterations. This is what leads to the idea of
where \( V_k = \text{diag}((\tilde{y}_k^{(1)})^2, (\tilde{y}_k^{(2)})^2, \ldots, (\tilde{y}_k^{(n)})^2), \sum_{i=1}^{n} (\tilde{y}_k^{(i)})^4 = \text{tr}(V_k^2), \) and \( \text{Tr} \) is the trace operation.

**Proof.** Consider the Lagrangian function of (7):

\[
L(\Psi_k, \alpha) = \frac{1}{2} \| \Psi_k \|_F^2 + \alpha \left( \tilde{y}_k^T \Psi_k \tilde{y}_k - \tilde{y}_k^T s_k + \tilde{y}_k^T B_k \tilde{y}_k \right),
\]

where \( \alpha \) is the corresponding Lagrangian multiplier. By differentiating \( L \) with respect to each \( \Psi_k^{(i)}, \Psi_k^{(2)}, \ldots, \Psi_k^{(n)} \), and setting them all equal to zero, we obtain

\[
\Psi_k^{(i)} = -\alpha (\tilde{y}_k^{(i)})^2 \quad \forall i = 1, 2, \ldots, n. \tag{10}
\]

Multiplying both sides of (10) by \((\tilde{y}_k^{(i)})^2\) and summing them all give

\[
\sum_{i=1}^{n} (\tilde{y}_k^{(i)})^4 \Psi_k^{(i)} = -\alpha \sum_{i=1}^{n} (\tilde{y}_k^{(i)})^4 \quad \text{for every} \ i = 1, 2, \ldots, n. \tag{11}
\]

Differentiating \( L \) with respect to \( \alpha \), and since \( \tilde{y}_k^T \Psi_k \tilde{y}_k = \sum_{i=1}^{n} (\tilde{y}_k^{(i)})^2 \Psi_k^{(i)} \), then we have

\[
\sum_{i=1}^{n} \left((\tilde{y}_k^{(i)})^2 \Psi_k^{(i)}\right) = \tilde{y}_k^T s_k - \tilde{y}_k^T B_k \tilde{y}_k. \tag{12}
\]

Equating (11) and (12) and substituting the relation into (10), finally we have

\[
\Psi_k^{(i)} = \frac{\tilde{y}_k^T s_k - \tilde{y}_k^T B_k \tilde{y}_k}{\sum_{i=1}^{n} (\tilde{y}_k^{(i)})^4} \quad \forall i = 1, 2, \ldots, n. \tag{13}
\]

Since \( B_k^{(0)} \) is a diagonal component of \( B_k \), \( \tilde{y}_k^{(i)} \) is the \( i \)-th component of \( \tilde{y}_k \), then \( V_k = \text{diag}((\tilde{y}_k^{(1)})^2, (\tilde{y}_k^{(2)})^2, \ldots, (\tilde{y}_k^{(n)})^2) \) and \( \sum_{i=1}^{n} (\tilde{y}_k^{(i)})^4 = \text{tr}(V_k^2) \). We further rewrite (13) as

\[
\Psi_k = \frac{(\tilde{y}_k^T s_k - \tilde{y}_k^T B_k \tilde{y}_k)}{\text{tr}(V_k^2)} V_k, \tag{14}
\]

which completes the proof. \( \square \)

Hence, the best possible updating formula for diagonal matrix \( B_{k+1} \) is given by

\[
B_{k+1} = B_k + \frac{(\tilde{y}_k^T s_k - \tilde{y}_k^T B_k \tilde{y}_k)}{\text{tr}(V_k^2)} V_k. \tag{15}
\]

Now, we can describe the algorithm for our proposed method as follows.

**Algorithm IDJA**

**Step 1.** Choose an initial guess \( x_0, \sigma \in (0, 1), \gamma > 1, B_0 = I_n, \alpha_0 > 0 \), and let \( k := 0 \).
Step 2. Compute $F(x_k)$, and if $\|F(x_k)\| \leq 10^{-8}$ stop.

Step 3. Compute $d = -F(x_k)B_k$.

Step 4. If $\|F(x_k + \alpha_d)\| \leq \sigma \|F(x_k)\|$, retain $\alpha_k$ and go to Step 5. Otherwise set $\alpha_{k+1} = \alpha_k/2$ and repeat Step 4.

Step 5. If $\|F(x_k + \alpha_d) - F(x_k)\| \geq \|F(x_k + \alpha_d)\| - \|F(x_k)\|$, retain $\alpha_k$ and go to Step 6. Otherwise set $\alpha_{k+1} := \alpha_k \times \gamma$ and repeat Step 5.

Step 6. Let $x_{k+1} = x_k + \alpha_d$.

Step 7. If $\|x_{k+1} - x_k\|_2 + \|F(x_k)\|_2 \leq 10^{-8}$ stop. Also go to Step 8.

Step 8. If $\|\Delta F_k\|_2 \geq \varepsilon_1$ where $\varepsilon_1 = 10^{-4}$, compute $B_{k+1}$ as defined by (15); if not, $B_{k+1} = B_k$.

Step 9. Set $k := k + 1$ and go to Step 2.

### 3. Convergence Result

This section presents local convergence results of the IDJA methods. To analyze the convergence of these methods, we will make the following assumptions on nonlinear systems $F$.

**Assumption 2.** (i) $F$ is differentiable in an open convex set $E$ in $\mathbb{R}^n$.

(ii) There exists $x^* \in E$ such that $F(x^*) = 0$; $F'(x)$ is continuous for all $x$.

(iii) $F'(x)$ satisfies the Lipschitz condition of order one that there exists a positive constant $\mu$ such that

$$\|F'(x) - F'(y)\| \leq \mu \|x - y\|, \quad (16)$$

for all $x, y \in \mathbb{R}^n$.

(iv) There exist constants $c_1 \leq c_2$ such that $c_1 \|\omega\|^2 \leq \omega^T F'(x)\omega \leq c_2 \|\omega\|^2$ for all $x \in E$ and $\omega \in \mathbb{R}^n$.

We can state the following result on the boundedness of $\|\Psi_0\|_F$ by assuming that, without loss of generality, the updating matrix (15) is always used, then we have the following.

**Theorem 3.** Suppose that $\{x_k\}$ is generated by Algorithm IDJA where $B_k$ is defined by (15). Assume that Assumption 2 holds. There exists $\beta > 0$, $\delta > 0$, $\alpha > 0$ and $\gamma > 0$, such that if $x_0 \in E$ and $B_0$ satisfies $\|I - B_0F'(x_0)\|_F < \delta$ for all $x_0 \in E$ then

$$\|I - B_kF'(x_k)\|_F < \delta_k,$$  \hspace{1cm} (17)

for some constant $\delta_k > 0$, $k \geq 0$.

**Proof.** Since $\|B_{k+1}\|_F = \|B_k + \Psi_k\|_F$, it follows that

$$\|B_{k+1}\|_F \leq \|B_k\|_F + \|\Psi_k\|_F.$$  \hspace{1cm} (18)

For $k = 0$ and assuming $B_0 = I$, we have

$$\|\Psi_0^{(i)}\| = \frac{\|\tilde{y}_0^T S_0 - \tilde{y}_0^T B_0 \tilde{y}_0 \|_2^2}{\text{tr}(V_0^2)} \leq \frac{\|\tilde{y}_0^T S_0 - \tilde{y}_0^T B_0 \tilde{y}_0 \|_2^2}{\text{tr}(V_0^2)},$$

where $(\tilde{y}_0^{(i)})^2$ is the largest element among $(\tilde{y}_0^{(i)})^2$, $i = 1, 2, \ldots, n$. After multiplying (19) by $(\tilde{y}_0^{(max)})^4 / (\tilde{y}_0^{(max)})^2$ and substituting $\text{tr}(V_0^2) = \sum_{i=1}^n (\tilde{y}_0^{(i)})^4$, we have

$$\|\Psi_0^{(i)}\| \leq \frac{\|\tilde{y}_0^T S_0 - \tilde{y}_0^T B_0 \tilde{y}_0 \|_2^2}{(\tilde{y}_0^{(max)})^2} (\tilde{y}_0^{(max)})^4.$$  \hspace{1cm} (20)

Since $(\tilde{y}_0^{(max)})^4 / \sum_{i=1}^n (\tilde{y}_0^{(i)})^4 \leq 1$, then (20) turns into

$$\|\Psi_0^{(i)}\| \leq \frac{\|\tilde{y}_0^T F'(x) \tilde{y}_0 - \tilde{y}_0^T B_0 \tilde{y}_0 \|_2^2}{(\tilde{y}_0^{(max)})^2}.$$  \hspace{1cm} (21)

From Assumption 2 and $B_0 = I$, (21) becomes

$$\|\Psi_0^{(i)}\| \leq \frac{|c - 1| (\tilde{y}_0^T \tilde{y}_0)}{(\tilde{y}_0^{(max)})^2},$$

where $c = \max\{|c_1|, |c_2|\}$. Since $(\tilde{y}_0^{(i)})^2 \leq (\tilde{y}_0^{(max)})^2$ for $i = 1, \ldots, n$, it follows that

$$\|\Psi_0^{(i)}\| \leq \frac{n |c - 1| (\tilde{y}_0^{(max)})^2}{(\tilde{y}_0^{(max)})^2}.$$  \hspace{1cm} (23)
Hence, we obtain
\[ \| \Psi_0 \|_F \leq n^{3/2} |c - 1| . \] (24)
Suppose \( \alpha = n^{3/2} |c - 1| \), then
\[ \| \Psi_0 \|_F \leq \alpha . \] (25)
From the fact that \( \| B_0 \|_F = \sqrt{n} \), it follows that
\[ \| B_1 \|_F \leq \beta , \] (26)
where \( \beta = \sqrt{n} + \alpha > 0 . \)
Therefore, if we assume that \( \| I - B_k F'(x^*) \|_F < \delta \), then
\[ \| I - B_k F'(x^*) \|_F = \| I - (B_0 + \Psi_0) F'(x^*) \|_F \]
\[ \leq \| I - B_0 F'(x^*) \|_F + \| \Psi_0 F'(x^*) \|_F \]
\[ \leq \| I - B_0 F'(x^*) \|_F + \| \Psi_0 \|_F \| F'(x^*) \|_F ; \] (27)
therefore, \( \| I - B_k F'(x^*) \|_F < \delta + \alpha \phi = \delta_1 . \)
Hence, by induction, \( \| I - B_k F'(x^*) \|_F < \delta_k \) for all \( k . \)

4. Numerical Results
In this section, the performance of IDJA method has been presented, when compared with Broyden’s method (BM), Chord Newton’s method (CN), Newton’s method (NM), and (DQNM) method proposed by [3], respectively. The codes are written in MATLAB 7.4 with a double precision computer; the stopping condition used is
\[ \| s_k \| + \| F(x_k) \| \leq 10^{-8} . \] (28)
The identity matrix has been chosen as an initial approximate Jacobian inverse.
We further design the codes to terminates whenever one of the following happens:
(i) the number of iteration is at least 200 but no point of \( x_k \) that satisfies (28) is obtained;
(ii) CPU time in seconds reaches 200;
(iii) Insufficient memory to initial the run.
The performance of these methods are compared in terms of number of iterations and CPU time in seconds. In the following, some details on the benchmarks test problems are presented.

Problem 1. Spares 1 function of Shin et al. [5]:
\[ f_i (x) = x_i^2 - 1 \quad i = 1, 2, \ldots, n, \ x_0 = (5, 5, \ldots, 5) . \] (29)

Problem 2. Trigonometric function of Spedicato [6]
\[ f_i (x) = n - \sum_{j=1}^{n} \cos x_j + i (1 - \cos x_i) - \sin x_i, \]
\[ i = 1, \ldots, n, \ x_0 = \left( \frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right) . \] (30)

Problem 3. System of \( n \) nonlinear equations
\[ f_i (x) = \sin (1 - x_i) \times \sum_{j=1}^{n} x_j^2 + 2x_{n+1} - 3x_{n-2} - \frac{1}{2} x_{n-4} + \frac{1}{2} x_{n-5} - x_i \ln (9 + x_i) \]
\[ - \frac{9}{2} \exp (1 - x_n) + 2 \]
\[ i = 1, 2, \ldots, n, \ x_0 = (0, 0, \ldots, 0)^T . \] (31)

Problem 4. System of \( n \) nonlinear equations
\[ f_i (x) = x_i^2 - 4 \exp \left( \sin \left( \frac{4 - x_i^2}{2} \right) \right) \]
\[ + \sin \left( \frac{4 - x_i^2}{2} \right) + i(x_n - x_i)^2 \]
\[ + \frac{2n - \sum_{i=1}^{n} x_i}{\cos x_i} \]
\[ i = 1, \ldots, n, \ x_0 = (2.8, 2.8, 2.8, \ldots, 2.8) . \]

Problem 5. System of \( n \) nonlinear equations
\[ f_i (x) = \left( \sum_{i=1}^{n} x_i \right) (x_i - 2) + (\cos x_i - 2) - 1 \]
\[ i = 1, \ldots, n, \ x_0 = (1, 1, 1, \ldots, 1) . \] (33)

Problem 6. System of \( n \) nonlinear equations
\[ f_i (x) = \sum_{i=1}^{n} x_i^2 - (\sin (x_i) - x_i^4 + \sin x_i^2) \]
\[ i = 1, \ldots, n, \ x_0 = (5, 5, 5, 5, \ldots, 5) . \] (34)

Problem 7. System of \( n \) nonlinear equations
\[ f_j (x) = \left( \sum_{i=1}^{n} x_i^2 - 1 \right) (x_j - 1) \]
\[ + x_j \left( \sum_{i=1}^{n} (x_i - 1) \right) - n + 1 \]
\[ j = 2, \ldots, n - 1, \ x_0 = (5, 5, 5, \ldots, 5) . \] (35)

Problem 8. System of \( n \) nonlinear equations
\[ f_i (x) = (1 - x_i^2) + x_i + x_i^2 x_{n-2} x_{n-1} x_n - 2 \]
\[ i = 1, 2, \ldots, n, \ x_0 = (5, 5, 5, \ldots, 5) . \] (36)
Table 2: Numerical Results of NM, CN, BM, DQNM, and IDJA methods.

<table>
<thead>
<tr>
<th>prob</th>
<th>Dim</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>7</td>
<td>0.156</td>
<td>98</td>
<td>0.094</td>
<td>15</td>
<td>0.043</td>
<td>14</td>
<td>0.016</td>
<td>2</td>
<td>0.011</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>10</td>
<td>0.187</td>
<td>—</td>
<td>—</td>
<td>18</td>
<td>0.062</td>
<td>16</td>
<td>0.032</td>
<td>13</td>
<td>0.032</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>7</td>
<td>0.203</td>
<td>—</td>
<td>—</td>
<td>24</td>
<td>0.140</td>
<td>15</td>
<td>0.031</td>
<td>7</td>
<td>0.015</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>13</td>
<td>0.031</td>
<td>10</td>
<td>0.030</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>13</td>
<td>0.265</td>
<td>—</td>
<td>—</td>
<td>53</td>
<td>0.109</td>
<td>17</td>
<td>0.031</td>
<td>12</td>
<td>0.031</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>8</td>
<td>0.203</td>
<td>—</td>
<td>—</td>
<td>16</td>
<td>0.047</td>
<td>14</td>
<td>0.031</td>
<td>7</td>
<td>0.017</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>8</td>
<td>0.185</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>26</td>
<td>0.031</td>
<td>16</td>
<td>0.030</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>11</td>
<td>0.234</td>
<td>—</td>
<td>—</td>
<td>11</td>
<td>0.094</td>
<td>11</td>
<td>0.032</td>
<td>10</td>
<td>0.016</td>
</tr>
</tbody>
</table>

Table 3: Numerical Results of NM, CN, BM, DQNM, and IDJA methods.

<table>
<thead>
<tr>
<th>prob</th>
<th>Dim</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>250</td>
<td>7</td>
<td>0.359</td>
<td>100</td>
<td>0.109</td>
<td>15</td>
<td>0.101</td>
<td>14</td>
<td>0.034</td>
<td>2</td>
<td>0.032</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>11</td>
<td>0.640</td>
<td>—</td>
<td>—</td>
<td>21</td>
<td>0.218</td>
<td>18</td>
<td>0.032</td>
<td>8</td>
<td>0.031</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>8</td>
<td>0.499</td>
<td>—</td>
<td>—</td>
<td>29</td>
<td>0.250</td>
<td>16</td>
<td>0.016</td>
<td>9</td>
<td>0.016</td>
</tr>
<tr>
<td>4</td>
<td>250</td>
<td>14</td>
<td>0.827</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>15</td>
<td>0.031</td>
<td>10</td>
<td>0.032</td>
</tr>
<tr>
<td>5</td>
<td>250</td>
<td>8</td>
<td>0.686</td>
<td>—</td>
<td>—</td>
<td>24</td>
<td>0.250</td>
<td>14</td>
<td>0.031</td>
<td>10</td>
<td>0.031</td>
</tr>
<tr>
<td>6</td>
<td>250</td>
<td>8</td>
<td>0.499</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>27</td>
<td>0.031</td>
<td>14</td>
<td>0.031</td>
</tr>
<tr>
<td>7</td>
<td>250</td>
<td>11</td>
<td>0.484</td>
<td>—</td>
<td>—</td>
<td>11</td>
<td>0.125</td>
<td>11</td>
<td>0.031</td>
<td>10</td>
<td>0.016</td>
</tr>
</tbody>
</table>

Table 4: Numerical results of NM, CN, BM, DQNM, and IDJA methods.

<table>
<thead>
<tr>
<th>prob</th>
<th>Dim</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500</td>
<td>7</td>
<td>0.796</td>
<td>101</td>
<td>0.702</td>
<td>15</td>
<td>0.671</td>
<td>14</td>
<td>0.016</td>
<td>2</td>
<td>0.011</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>13</td>
<td>1.997</td>
<td>—</td>
<td>—</td>
<td>23</td>
<td>0.972</td>
<td>19</td>
<td>0.031</td>
<td>9</td>
<td>0.032</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>7</td>
<td>1.4352</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>17</td>
<td>0.031</td>
<td>9</td>
<td>0.031</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
<td>8</td>
<td>1.484</td>
<td>—</td>
<td>—</td>
<td>23</td>
<td>0.998</td>
<td>14</td>
<td>0.032</td>
<td>10</td>
<td>0.045</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>8</td>
<td>1.498</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>32</td>
<td>0.047</td>
<td>15</td>
<td>0.047</td>
</tr>
<tr>
<td>6</td>
<td>500</td>
<td>11</td>
<td>1.451</td>
<td>—</td>
<td>—</td>
<td>11</td>
<td>0.515</td>
<td>11</td>
<td>0.031</td>
<td>9</td>
<td>0.031</td>
</tr>
</tbody>
</table>

Table 5: Numerical results of NM, CN, BM, DQNM, and IDJA methods.

<table>
<thead>
<tr>
<th>prob</th>
<th>Dim</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
<th>NI</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>7</td>
<td>2.730</td>
<td>103</td>
<td>3.167</td>
<td>38</td>
<td>9.438</td>
<td>14</td>
<td>0.016</td>
<td>2</td>
<td>0.011</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>31</td>
<td>7.722</td>
<td>20</td>
<td>0.032</td>
<td>8</td>
<td>0.043</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>9</td>
<td>5.819</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>17</td>
<td>0.031</td>
<td>9</td>
<td>0.031</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>11</td>
<td>0.064</td>
<td>10</td>
<td>0.064</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>16</td>
<td>8.705</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>22</td>
<td>0.031</td>
<td>10</td>
<td>0.031</td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>8</td>
<td>6.474</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>14</td>
<td>0.062</td>
<td>11</td>
<td>0.061</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>8</td>
<td>4.321</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>38</td>
<td>0.062</td>
<td>31</td>
<td>0.047</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>11</td>
<td>4.882</td>
<td>—</td>
<td>—</td>
<td>11</td>
<td>2.418</td>
<td>11</td>
<td>0.032</td>
<td>10</td>
<td>0.031</td>
</tr>
</tbody>
</table>
The numerical results presented in Tables 1, 2, 3, 4, and 5 demonstrate clearly the proposed method (IDJA) shows good improvements, when compared with NM, CN, BM, and DQNM, respectively. In addition, it is worth mentioning, the IDJA method does not require more storage locations than classic diagonal quasi-Newton’s methods. One can observe from the tables that the proposed method (IDJA) is faster than DQNM methods and required little time to solve the problems when compared to the other Newton-like methods and still keeping memory requirement and CPU time in seconds to only $O(n)$.

5. Conclusions

In this paper, we present an improved diagonal quasi-Newton update via new quasi-Cauchy condition for solving large-scale Systems of nonlinear equations (IDJA). The Jacobian inverse approximation is derived based on the quasi-Cauchy condition. The anticipation has been to further improve the diagonal Jacobian, by modifying the quasi-Cauchy relation so as to carry some additional information from the functions. It is also worth mentioning that the method is capable of significantly reducing the execution time (CPU time), as compared to NM, CN, BM, and DQNM methods while maintaining good accuracy of the numerical solution to some extent. Another fact that makes the IDJA method appealing is that throughout the numerical experiments it never fails to converge. Hence, we can claim that our method (IDJA) is a good alternative to Newton-type methods for solving large-scale systems of nonlinear equations.

References
