Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 875935, 6 pages
http://dx.doi.org/10.1155/2013/875935

Research Article

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

Mohammed Yusuf Waziri[1,2] **and Zanariah Abdul Majid**[1,3]

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

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 relation 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

$$F(x) = 0, \tag{1}$$

where $F : R^n \rightarrow R^n$ is a nonlinear mapping. Often, the mapping F is assumed to be satisfying the following assumptions:

(A1) there exists an $x^* \in R^n$ 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

$$x_{k+1} = x_k - \left(F'(x_k)\right)^{-1} F(x_k), \tag{2}$$

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 this paper. The anticipation has been to further improve the performance of diagonal updating, by modifying the quasi-Cauchy relation so as to carry some additional information from the functions. We organized the paper as follows. In the next section, we present the details of the proposed method. Convergence results are present in Section 3. Some numerical results are reported in Section 4. Finally, conclusions are made in Section 5.

2. Derivation Process

This section presents a new diagonal quasi-Newton-like method for solving large-scale systems of nonlinear equations. The quasi-Newton method is an iterative method that generates a sequence of points $\{x_k\}$ from a given initial guess x_0 via the following form:

$$x_{k+1} = x_k - \alpha_k B_k F(x_k) \quad k = 0, 1, 2 \ldots, \tag{3}$$

where α_k is a step length and B_k is an approximation to the Jacobian inverse which can be updated at each iteration for $k = 0, 1, 2 \ldots$; the updated matrix B_{k+1} is chosen in such a way that it satisfies the secant equation, that is,

$$B_{k+1} s_k = y_k. \tag{4}$$

It is clear that the only Jacobian information we have is y, and this is only approximation information. To this end, we incorporate more information from s_k and F_k to y in order to present a better approximation to the Jacobian matrix. We consider the modification on y presented by Li and Fukushima [4]:

$$\check{y} = y_k + v_k \|F_k\| s_k, \tag{5}$$

where $v_k = 1 + \max\{-s_k^T y_k / \|s_k\|^2, 0\}$.

Our aim here is to build a square matrix, say B, using diagonal updating scheme which is an approximation to the Jacobian inverse, and we let B_{k+1} satisfy the quasi-Cauchy equation, that is,

$$\check{y}_k^T s_k = \check{y}_k^T B_{k+1} \check{y}_k. \tag{6}$$

In addition, the deviation between B_{k+1} and B_k is minimized under some norms; hence, in the following theorem, we state the resulting update formula for B_k.

Theorem 1. *Assume that B_{k+1} be the diagonal update of a diagonal matrix B_k. Let us denote the deviation between B_k and B_{k+1} as $\Psi_k = B_{k+1} - B_k$. Suppose that $\check{y}_k \neq 0$ which is defined by* (5). *Consider the following problem:*

$$\begin{aligned} &\min \frac{1}{2} \|\Psi_k\|_F^2 \\ &s.t \quad \check{y}_k^T (B_k + \Psi_k) \check{y}_k = \check{y}_k^T s_k, \end{aligned} \tag{7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Hence, the optimal solution of (7) *is given by*

$$\Psi_k = \frac{(\check{y}_k^T s_k - \check{y}_k^T B_k \check{y}_k)}{\operatorname{tr}(V_k^2)} V_k, \tag{8}$$

where $V_k = \operatorname{diag}((\check{y}_k^{(1)})^2, (\check{y}_k^{(2)})^2, \ldots, (\check{y}_k^{(n)})^2)$, $\sum_{i=1}^n (\check{y}_k^{(i)})^4 = \operatorname{tr}(V_k^2)$, and Tr *is the trace operation.*

Proof. Consider the Lagrangian function of (7):

$$L(\Psi_k, \alpha) = \frac{1}{2} \|\Psi_k\|_F^2 + \alpha \left(\check{y}_k^T \Psi_k \check{y}_k - \check{y}_k^T s_k + \check{y}_k^T B_k \check{y}_k \right), \tag{9}$$

where α is the corresponding Lagrangian multiplier. By differentiating L with respect to each $\Psi_k^{(1)}, \Psi_k^{(2)}, \ldots, \Psi_k^{(n)}$, and setting them all equal to zero, we obtain

$$\Psi_k^{(i)} = -\alpha \left(\check{y}_k^{(i)} \right)^2 \quad \forall i = 1, 2, \ldots, n. \tag{10}$$

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

$$\sum_{i=1}^n \left(\check{y}_k^{(i)} \right)^2 \Psi_k^{(i)} = -\alpha \sum_{i=1}^n \left(\check{y}_k^{(i)} \right)^4 \quad \text{for every } i = 1, 2, \ldots, n. \tag{11}$$

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

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

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

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

Since $B_k^{(i)}$ is a diagonal component of B_k, $\check{y}_k^{(i)}$ is the ith component of vector $\check{y}_k$, then $V_k = \operatorname{diag}((\check{y}_k^{(1)})^2, (\check{y}_k^{(2)})^2, \ldots, (\check{y}_k^{(n)})^2)$ and $\sum_{i=1}^n (y_k^{(i)})^4 = \operatorname{tr}(V_k^2)$. We further rewrite (13) as

$$\Psi_k = \frac{(\check{y}_k^T s_k - \check{y}_k^T B_k \check{y}_k)}{\operatorname{tr}(V_k^2)} V_k, \tag{14}$$

which completes the proof. □

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

$$B_{k+1} = B_k + \frac{(\check{y}_k^T s_k - \check{y}_k^T B_k \check{y}_k)}{\operatorname{tr}(V_k^2)} V_k. \tag{15}$$

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 $k := 0$.

Table 1: Numerical results of NM, CN, BM, DQNM, and IDJA methods.

prob	Dim	NM		CN		BM		DQNM		IDJA	
		NI	CPU	NI	CPU	NI	CPU	NI	CPU	NI	CPU
1	50	7	0.046	55	0.031	15	0.031	14	0.016	2	0.011
2	50	9	0.078	344	0.062	15	0.031	15	0.031	13	0.031
3	50	10	0.062	—	—	—	—	20	0.016	10	0.016
4	50	—	—	—	—	—	—	19	0.031	9	0.031
5	50	12	0.078	—	—	42	0.031	16	0.016	8	0.015
6	50	8	0.064	—	—	16	0.032	14	0.031	7	0.014
7	50	8	0.094	—	—	—	—	25	0.031	14	0.010
8	50	11	0.064	—	—	11	0.0312	11	0.016	9	0.016

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

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

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

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 $\|\Delta F_k\|_2 \geq \epsilon_1$ where $\epsilon_1 = 10^{-4}$, 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 $\mathfrak{R}^n$.

(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 is there exists a positive constant μ such that

$$\left\|F'(x) - F'(y)\right\| \leq \mu \left\|x - y\right\|, \tag{16}$$

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

(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 $\omega \in \mathfrak{R}^n$.

We can state the following result on the boundedness of $\{\|\Psi_k\|_F\}$ 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 $\|I - B_0F'(x^*)\|_F < \delta$ for all $x_k \in E$ then*

$$\left\|I - B_k F'\left(x^*\right)\right\|_F < \delta_k, \tag{17}$$

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

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

$$\left\|B_{k+1}\right\|_F \leq \left\|B_k\right\|_F + \left\|\Psi_k\right\|_F. \tag{18}$$

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

$$\begin{aligned}\left|\Psi_0^{(i)}\right| &= \left|\frac{\breve{y}_0{}^T s_0 - \breve{y}_0{}^T B_0 \breve{y}_0}{\operatorname{tr}\left(V_0^2\right)}\left(\breve{y}_0{}^{(i)}\right)^2\right| \\ &\leq \frac{\left|\breve{y}_0{}^T s_0 - \breve{y}_0{}^T B_0 \breve{y}_0\right|}{\operatorname{tr}\left(V_0^2\right)}\left(\breve{y}_0{}^{(\max)}\right)^2,\end{aligned} \tag{19}$$

where $(\breve{y}_0{}^{(\max)})^2$ is the largest element among $(\breve{y}_0{}^{(i)})^2$, $i = 1, 2, \ldots, n$.

After multiplying (19) by $(\breve{y}_0{}^{(\max)})^2/(\breve{y}_0{}^{(\max)})^2$ and substituting $\operatorname{tr}(V_0^2) = \sum_{i=1}^n (\breve{y}_0{}^{(i)})^4$, we have

$$\left|\Psi_0^{(i)}\right| \leq \frac{\left|\breve{y}_0{}^T s_0 - \breve{y}_0{}^T B_0 \breve{y}_0\right|}{\left(\breve{y}_0{}^{(\max)}\right)^2 \sum_{i=1}^n \left(\breve{y}_0{}^{(i)}\right)^4}\left(\breve{y}_0{}^{(\max)}\right)^4. \tag{20}$$

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

$$\left|\Psi_0^{(i)}\right| \leq \frac{\left|\breve{y}_0{}^T F'(x)\, \breve{y}_0 - \breve{y}_0{}^T B_0 \breve{y}_0\right|}{\left(\breve{y}_0{}^{(\max)}\right)^2}. \tag{21}$$

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

$$\left|\Psi_0^{(i)}\right| \leq \frac{|c - 1|\left(\breve{y}_0{}^T \breve{y}_0\right)}{\left(\breve{y}_0{}^{(\max)}\right)^2}, \tag{22}$$

where $c = \max\{|c_1|, |c_2|\}$.

Since $(\breve{y}_0{}^{(i)})^2 \leq (\breve{y}_0{}^{(\max)})^2$ for $i = 1, \ldots, n$, it follows that

$$\left|\Psi_0^{(i)}\right| \leq \frac{n|c - 1|\left(\breve{y}_0{}^{(\max)}\right)^2}{\left(\breve{y}_0{}^{(\max)}\right)^2}. \tag{23}$$

Hence, we obtain

$$\|\Psi_0\|_F \le n^{3/2}\,|c-1|. \tag{24}$$

Suppose $\alpha = n^{3/2}|c-1|$, then

$$\|\Psi_0\|_F \le \alpha. \tag{25}$$

From the fact that $\|B_0\|_F = \sqrt{n}$, it follows that

$$\|B_1\|_F \le \beta, \tag{26}$$

where $\beta = \sqrt{n} + \alpha > 0$.

Therefore, if we assume that $\|I - B_0F'(x^*)\|_F < \delta$, then

$$\begin{aligned}\left\|I - B_1F'\left(x^*\right)\right\|_F &= \left\|I - \left(B_0 + \Psi_0\right)F'\left(x^*\right)\right\|_F \\ &\le \left\|I - B_0F'\left(x^*\right)\right\|_F + \left\|\Psi_0F'\left(x^*\right)\right\|_F \\ &\le \left\|I - B_0F'\left(x^*\right)\right\|_F + \left\|\Psi_0\right\|_F\left\|F'\left(x^*\right)\right\|_F;\end{aligned} \tag{27}$$

therefore, $\|I - B_1F'(x^*)\|_F < \delta + \alpha\phi = \delta_1$.

Hence, by induction, $\|I - B_kF'(x^*)\|_F < \delta_k$ 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

$$\|s_k\| + \|F(x_k)\| \le 10^{-8}. \tag{28}$$

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]:

$$f_i(x) = x_i^2 - 1 \quad i = 1,2,\ldots,n,\ x_0 = (5,5,\ldots,5). \tag{29}$$

Problem 2. Trigonometric function of Spedicato [6]

$$\begin{aligned}f_i(x) &= n - \sum_{j=1}^{n}\cos x_j + i\left(1 - \cos x_i\right) - \sin x_i, \\ i &= 1,\ldots,n,\ x_0 = \left(\frac{1}{n},\frac{1}{n},\ldots,\frac{1}{n}\right).\end{aligned} \tag{30}$$

Problem 3. System of n nonlinear equations

$$\begin{aligned}f_i(x) &= \sin\left(1 - x_i\right) \\ &\quad\times\sum_{i=1}^{n}x_i^2 + 2x_{n-1} \\ &\quad - 3x_{n-2} - \frac{1}{2}x_{n-4} + \frac{1}{2}x_{n-5} - x_i\ln\left(9 + x_i\right) \\ &\quad - \frac{9}{2}\exp\left(1 - x_n\right) + 2 \\ &\quad i = 1,2,\ldots,n,\ x_0 = (0,0,\ldots,0)^T.\end{aligned} \tag{31}$$

Problem 4. System of n nonlinear equations

$$\begin{aligned}f_i(x) &= x_i^2 - 4\exp\left(\sin\left(4 - x_i^2\right)\right) \\ &\quad + \sin\left(4 - x_i\right)^2 + i\left(x_n - x_i\right)^2 \\ &\quad + \frac{2n - \sum_{i=1}^{n}x_i}{\cos x_i} \\ i &= 1,\ldots,n,\ x_0 = (2.8,2.8,2.8,\ldots,2.8).\end{aligned} \tag{32}$$

Problem 5. System of n nonlinear equations

$$\begin{aligned}f_i(x) &= \left(\sum_{i=1}^{n}x_i\right)\left(x_i - 2\right) + \left(\cos x_i - 2\right) - 1 \\ i &= 1,\ldots,n,\ x_0 = (1,1,1,\ldots,1).\end{aligned} \tag{33}$$

Problem 6. System of n nonlinear equations

$$\begin{aligned}f_i(x) &= \sum_{i=1}^{n}x_i^2 - \left(\sin\left(x_i\right) - x_i^4 + \sin x_i^2\right) \\ i &= 1,\ldots,n,\ x_0 = (.5,.5,.5,\ldots,.5).\end{aligned} \tag{34}$$

Problem 7. System of n nonlinear equations

$$\begin{aligned}f_j(x) &= \left(\sum_{i=1}^{n}x_i^2 - 1\right)\left(x_j - 1\right) \\ &\quad + x_j\left(\sum_{i=1}^{n}\left(x_i - 1\right)\right) - n + 1 \\ f_n(x) &= \left(\sum_{i=1}^{n}x_i^2 - 1\right)\left(x_n - 1\right) + \left(\cos x_n - 1\right) - 1 \\ j &= 2,\ldots,n-1,\ x_0 = (.5,.5,.5,\ldots).\end{aligned} \tag{35}$$

Problem 8. System of n nonlinear equations

$$\begin{aligned}f_i(x) &= \left(1 - x_i^2\right) + x_i + x_i^2x_{n-2}x_{n-1}x_n - 2 \\ i &= 1,2,\ldots,n,\ x_0 = (.5,.5,\ldots,.5).\end{aligned} \tag{36}$$

Table 2: Numerical Results of NM, CN, BM, DQNM, and IDJA methods.

prob	Dim	NM		CN		BM		DQNM		IDJA	
		NI	CPU	NI	CPU	NI	CPU	NI	CPU	NI	CPU
1	100	7	0.156	98	0.094	15	0.043	14	0.016	2	0.011
2	100	10	0.187	—	—	18	0.062	16	0.032	13	0.032
3	100	7	0.203	—	—	24	0.140	15	0.031	7	0.015
4	100	—	—	—	—	—	—	13	0.031	10	0.030
5	100	13	0.265	—	—	53	0.109	17	0.031	12	0.031
6	100	8	0.203	—	—	16	0.047	14	0.031	7	0.017
7	100	8	0.185	—	—	—	—	26	0.031	16	0.030
8	100	11	0.234	—	—	11	0.094	11	0.032	10	0.016

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

prob	Dim	NM		CN		BM		DQNM		IDJA	
		NI	CPU	NI	CPU	NI	CPU	NI	CPU	NI	CPU
1	250	7	0.359	100	0.109	15	0.101	14	0.034	2	0.032
2	250	11	0.640	—	—	21	0.218	18	0.032	8	0.031
3	250	8	0.499	—	—	29	0.250	16	0.016	9	0.016
4	250	—	—	—	—	—	—	15	0.031	10	0.032
5	250	14	0.827	—	—	—	—	19	0.031	8	0.016
6	250	8	0.686	—	—	24	0.250	14	0.031	10	0.031
7	250	8	0.499	—	—	—	—	27	0.031	14	0.031
8	250	11	0.484	—	—	11	0.125	11	0.031	10	0.016

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

prob	Dim	NM		CN		BM		DQNM		IDJA	
		NI	CPU	NI	CPU	NI	CPU	NI	CPU	NI	CPU
1	500	7	0.796	101	0.702	15	0.671	14	0.016	2	0.011
2	500	13	1.997	—	—	23	0.972	19	0.031	9	0.032
3	500	7	1.4352	—	—	—	—	17	0.031	9	0.031
4	500	—	—	—	—	—	—	12	0.030	10	0.031
5	500	15	2.449	—	—	—	—	21	0.031	9	0.031
6	500	8	2.184	—	—	23	0.998	14	0.032	10	0.045
7	500	8	1.498	—	—	—	—	32	0.047	15	0.047
8	500	11	1.451	—	—	11	0.515	11	0.031	9	0.031

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

prob	Dim	NM		CN		BM		DQNM		IDJA	
		NI	CPU	NI	CPU	NI	CPU	NI	CPU	NI	CPU
1	1000	7	2.730	103	3.167	38	9.438	14	0.016	2	0.011
2	1000	—	—	—	—	31	7.722	20	0.032	8	0.043
3	1000	9	5.819	—	—	—	—	17	0.031	9	0.031
4	1000	—	—	—	—	—	—	11	0.064	10	0.064
5	1000	16	8.705	—	—	—	—	22	0.031	10	0.031
6	1000	8	6.474	—	—	—	—	14	0.062	11	0.061
7	1000	8	4.321	—	—	—	—	38	0.062	31	0.047
8	1000	11	4.882	—	—	11	2.418	11	0.032	10	0.031

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

[1] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, NJ, USA, 1983.

[2] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, vol. 16, SIAM, Philadelphia, Pa, USA, 1995.

[3] W. J. Leong, M. A. Hassan, and M. Waziri Yusuf, "A matrix-free quasi-Newton method for solving large-scale nonlinear systems," *Computers & Mathematics with Applications*, vol. 62, no. 5, pp. 2354–2363, 2011.

[4] D.-H. Li and M. Fukushima, "A modified BFGS method and its global convergence in nonconvex minimization," *Journal of Computational and Applied Mathematics*, vol. 129, no. 1-2, pp. 15–35, 2001.

[5] B.-C. Shin, M. T. Darvishi, and C.-H. Kim, "A comparison of the Newton-Krylov method with high order Newton-like methods to solve nonlinear systems," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3190–3198, 2010.

[6] E. Spedicato, "Cumputational experience with quasi-Newton algorithms for minimization problems of moderatetly large size," Tech. Rep. CISE-N-175 3, pp. 10–41, 1975.