Malaysian Journal of Mathematical Sciences 6(1): 59-73 (2012)

Diagonal Broyden-Like Method for Large-Scale Systems of Nonlinear Equations

[1,3]Mohammed Yusuf Waziri, [1,2] Wah June Leong and [1,2]Malik Abu Hassan

[1]Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

[2]Institute for Mathematical Research, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

[3]Department of Mathematical Sciences, Faculty of Science, Bayero University Kano (BUK), Kano, Nigeria,

E-mail: mywaziri@gmail.com

ABSTRACT

The prominent method for solving nonlinear equation is the classical Newton's method. Nevertheless the method is computational expensive, especially when handling large-scale systems, it requires more computation for storage of the Jacobian matrix, as well as solving a Newton's system at each iteration. In this paper, we continue the spirit of Newton method to develop an alternative approximation for the Newton step via diagonal updating. The anticipation behind our approach is to reduce the computational complexity of the classical Newton's method for solving large-scale systems of nonlinear equations. The convergence of the method proposed has been proven under standard assumptions. Numerical investigation into the effectiveness and consistency of the proposed scheme are given by numerical evaluation of some well-known benchmark nonlinear systems with some variants of Newton's method.

Keywords: Nonlinear equations, Diagonal Updating Large scale systems, Broyden's method.

1. INTRODUCTION

Consider the system of nonlinear equations

$$\begin{aligned} &f_1(x_1, x_2, x_3, ..., x_n) = 0 \\ &f_1(x_1, x_2, x_3, ..., x_n) = 0 \\ &\vdots \\ &f_n(x_1, x_2, x_3, ..., x_n) = 0. \end{aligned} \tag{1}$$

The above system can be denoted by $F(x)=0$, where $F=(f_1,f_2,...,f_n)\colon R^n \to R^n$ is continuously differentiable in an open neighborhood $\Delta^* \subset \Delta$ of a solution $x^* \in \Delta$ of the system, where $F(x^*)=0$ and the Jacobian matrix of F at x^* is given by $F'(x^*)$ that is a nonsingular matrix. The famous iterative method for solving (1) is the Newton's method. For any specified initial point x_0 in the neighborhood of x^* , the solution of (1) that is obtained by Newton's method is an iterative procedure given below:

Algorithm NM (Newton's method)

Step 1: Solve $F'(x_k)s_k = -F(x_k)$ for s_k , $k=0,1,2,...$

Step 2: Update $x_{k+1} = x_k + s_k$.

Where $F'(x_k)$ is the Jacobian matrix of F at x_k, s_k is a correction to the previous iteration and the equation in Step 1 is the Newton's system. The convergence of Algorithm NM is guaranteed where the rate is quadratic for an initial point x_0 in the neighborhood of x^* when the Jacobian matrix is nonsingular at the solution x^* (see Dennis (1983)), that is,

$$\left\|x_{k+1}-x^*\right\| \le h\left\|x_k-x^*\right\|, k=0,1,2,\ldots \qquad (2)$$

for step-size h.

The numerical implementation of algorithm NM turns to be expensive due to the computation for storing the Jacobian matrix, as well as solving the Newton's system at each iteration (see Dennis (1983)). These basic shortcomings have attracted the attention of many scholars over time. In addition to these, the computation of the Jacobian matrix which entails first-order derivatives of the systems are quite costly and even not available or could not be done precisely, in these case Newton's method cannot be used directly.

It is imperative to mention that there is quite a number of revised Newton's -type methods that is being introduced which includes fixed Newton's, inexact Newton's and quasi-Newton's to diminish the weakness of Newton's method see for example (Kelly (1995)).

However, the foremost difficulty of these methods is the matrix storage requirements particularly when handling large systems of nonlinear equations. To overcome this difficulty, the simplest and easiest modification is the fixed Newton's method, where the computation for storing the Jacobian matrix at each iterations (except at $k=0$) is avoided but requires solving systems of n linear equations, Natasa *et al.* (2001).

Since the less information of the Jacobian matrix at each iteration, the fixed Newton's method has linear rate of convergence. For any specified initial point x_0 in the neighborhood of x^*, the solution of (1) that is by the fixed Newton's method is an iterative procedure stated below:

Algorithm FN (Fixed Newton method)

Given initial x_0,

Step 1: Solve $F'(x_0)s_k = -F(x_k)$ for s_k, $k=0,1,2,...$

Step 2: Update $x_{k+1} = x_k + s_k$.

The second revised Newton's-type method is the inexact Newton's method, where approximate solution of Newton's system is obtained (step1 of algorithm FN) by some iterations (see Eisenstat *et al.* (1982) for details), as mentioned below:

Algorithm INM (Inexact Newton's method)

Step 1: Find some s_k, which satisfies $F'(x_k)s_k = -F(x_k) + r_k$

where $\|r_k\| \le \eta_k \|F(x_k)\|$.

Step 2: Set $x_{k+1} = x_k + s_k$.

Where η_k is a sequence of forcing terms for $0 \le \eta_k \le 1$, $k=0,1,2,...$

The Quasi-Newton method is another revised Newton's-type method, where it replaces the Jacobian matrix or inverse of the Jacobian matrix with an approximation, which can be updated at each iteration, Drangoslav *et al.* (1996), according to following stages:

Algorithm QN (Quasi-Newton method)

Given initial guess x_0,

Step 1: Solve $B_k s_k = -F(x_k)$, for s_k, $k = 0,1,...,n$

Step 2: Update $x_{k+1} = x_k + s_k$,

where B_k is an approximation to the Jacobian matrix. The basic idea of quasi-Newton's method is to reduce the evaluation cost of the Jacobian matrix. If function evaluations are very expensive, then the cost of finding a solution by quasi-Newton's method could be much smaller than those with some variant of Newton's methods, like the inexact Newton's method Drangoslav *et al.* (1996). Various Jacobian approximation matrices such as the Broyden (1965) updating are given below:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k}, \tag{3}$$

where B_k is a Jacobian matrix approximation, $y_k = F(x_{k+1}) - F(x_k)$ and $s_k = x_{k+1} - x_k$. In fact, the most crucial part of these methods is on forming and storing a full-matrix approximation to the Jacobian matrix at each iteration. Some alternatives methods are proposed to do away with the shortcomings of the Newton's method. These weaknesses, together with some other weaknesses of the Newton's-like methods especially when handling large-scale systems of nonlinear equations, lead us to the inspiration of this paper. It is important to note that, the diagonal updating scheme has been applied in unconstrained optimization problems (see Hassan *et al*. (2009); Leong *et al*. (2009); Leong *et al*. (2010; 2011a; 2011b); Modarres *et al* (2010;2011); Waziri *et al*. (2010a; 2010b;2011)).

In this paper, we design and implement an alternative approximation to the Newton's step via diagonal updating by means of variational techniques. It is worth mentioning that the new updating scheme has been applied to solve (1) without the cost of computing or storing the Jacobian matrix. This can reduce: computational cost, matrix storage requirement, execution time (CPU time) and eliminates the needs of solving n linear equations at each iteration. The proposed diagonal updating method works efficient and the results are very encouraging. In addition, the method proposed can also solve a couple of problems, which cannot be solved by methods involving Jacobian matrix computation. The rest of this paper is organized as follows.

We present the proposed method in Section 2 and discuss the convergence analysis in Section 3. Some numerical results are reported in Section 4 and finally Conclusion is given in Section 5.

2. DIAGONAL UPDATING

In this section we shall derive a new approximation scheme to the Newton's step based on diagonal updating. This scheme does not require function derivatives but it generates a sequence of points $\{x_k\}$ via

$$x_{k+1} = x_k - Q_k F(x_k), \tag{4}$$

where Q_k is a diagonal matrix. Our plan here is to build a square matrix Q using diagonal updating which is an approximation to Jacobian inverse into diagonal matrix. Let us denote $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. Then by the mean value theorem, we have

$$\overline{F}'(x_k)s_k = y_k, \tag{5}$$

where $\overline{F}'(x_k) = \int_0^1 F\left(x_k + \theta s_{k-1}\right) d\theta$ is Jacobian matrix. Equation (5) is considered as the secant equation. On the other hand, if $\overline{F}'(x_k)$ is invertible, according to (5) we have

$$s_k = (\overline{F}'(x_k))^{-1} y_k. \tag{6}$$

We propose to approximate the Newton's step S_k^N through

$$S_k^N = (\overline{F}'(x_k))^{-1} F(x_k) \approx Q_k F(x_k), \tag{7}$$

where we update Q by $Q_{k+1} = Q_k + B_k$ with a correction B that is also a diagonal matrix. In addition, the deviation between Q_{k+1} and Q_k is minimized under some variational technique, hence, in the following theorem, we state the resulting update formula for Q_k.

Theorem 2.1 *Assumed that* Q_k *is a diagonal matrix and diagonal update of* Q_k *to be* Q_{k+1}. *Let us denotes the deviation between* Q_k *and* Q_{k+1} *as* $B_k = Q_{k+1} - Q_k$. *Supposed that* $y \neq 0$. *Consider the following problem:*

$$\min \frac{1}{2}\|B_k\|_F^2$$
$$\text{s.t } y_k^T(Q_k + B_k)y_k = y_k^T s_k, \tag{8}$$

where $\|.\|_F$ *denotes the Frobenuis norm. Hence the optimal solution of (E:ma) is given by*

$$B_k = \frac{(y_k^T s_k - y_k^T Q_k y_k)}{Tr(A_k^2)} A_k, \tag{9}$$

where $A_k = diag(y_k^{(1)^2}, y_k^{(2)^2}, ..., y_k^{(n)^2})$, $\sum_{i=1}^n y_k^{(i)^4} = Tr(A_k^2)$ *and* $Tr(.)$ *is the trace operator.*

Proof. From the fact that, the objective function and the constraint are convex, the unique solution can be obtained by considering its Langrangian function as follows:

$$L(B_k, \alpha) = \frac{1}{2}\|B_k\|_F^2 + \alpha\left(y_k^T B_k y_k - y_k^T s_k + y_k^T Q_k y_k\right). \tag{10}$$

where α is the corresponding Langrangian multiplier. By differentiating L with respect to each $B_k^{(1)}, B_k^{(2)}, ..., B_k^{(n)}$, and set them equal to zero, we obtain

$$B_k^{(i)} = -\alpha(y_k^{(i)})^2 \text{ for all } i = 1, 2, ..., n. \tag{11}$$

Through multiplying both sides of (11) by $(y_k^{(i)})^2$ and sum them to give

$$\sum_{i=1}^n y_k^{(i)^2} B_k^{(i)} = -\alpha \sum_{i=1}^n y_k^{(i)^4} \text{ for every } i = 1, 2, ..., n. \tag{12}$$

We invoking the constraint by differentiating L with respect to α and since $y_k^T B_k y_k = \sum_{i=1}^n (y_k^{(i)^2} B^{(i)})$ then we have

$$\sum_{i=1}^n (y_k^{(i)^2} B_k^{(i)}) = y_k^T s_k - y_k^T Q_k y_k. \tag{13}$$

Equating (12) and (13) yields

$$\alpha = -\frac{y_k^T s_k - y_k^T Q_k y_k}{\sum_{i=1}^{n} y_k^{(i)^4}}. \tag{14}$$

Finally, by substituting (14) into (11) gives

$$B_k^{(i)} = \frac{y_k^T s_k - y_k^T Q_k y_k}{\sum_{i=1}^{n} y_k^{(i)^4}} (y_k^{(i)})^2 \quad \text{for } i = 1, 2, ..., n. \tag{15}$$

this completes the proof .

Hence, from Theorem 2.1 the best possible updating formula for diagonal matrix Q_{k+1} is given by

$$Q_{k+1} = Q_k + \frac{(y_k^T s_k - y_k^T Q_k y_k)}{Tr(A^2)} A_k. \tag{16}$$

To safeguard the possibly very small $\|y_k\|$ and $Tr(A^2)$, we require that $\|y_k\| \geq \varepsilon_1$ for some chosen small $\varepsilon_1 > 0$. Else we set $Q_{k+1} = Q_k$. Hence,

Q_{k+1} is given as: $Q_{k+1} = \begin{cases} Q_k + \frac{(y_k^T s_k - y_k^T Q_k y_k)}{Tr(A^2)} A_k & ; \|y_k\| \geq \varepsilon_1, \\ Q_k & ; \text{otherwise.} \end{cases}$

Finally, we propose the following algorithm.

Algorithm DBLM (Diagonal Broyden-Like method)

Given x_0 and Q_0, set $k = 0$

Step 1 : Compute $F(x_k)$ and $x_{k+1} = x_k - Q_k F(x_k)$.

Step 2 : If $\|s_k\|_2 + \|F(x_k)\|_2 \leq 10^{-4}$ stop. Else go to Step 3.

Step 3 : If $\|y_k\|_2 \geq \varepsilon_1$ where $\varepsilon_1 = 10^{-4}$, compute Q_{k+1} from (16), else, $Q_{k+1} = Q_k$. Set $k = k + 1$ and go to 1.

3. CONVERGENCE ANALYSIS

This section states the convergence analysis of the proposed method. In doing so, we require the following standard assumptions on F.

Assumption 1

(i) F is differentiable in an open convex set Δ in $\Re^n$

(ii) $F'(x)$ is continuous for all x and there exists $x^* \in \Delta$ such that $F(x^*) = 0$,

(iii) There exists constants $t_1 \leq t_2$ such that

$$t_1 \|\Phi\|^2 \leq \Phi^T F'(x)\Phi \leq t_2 \|\Phi\|^2 \tag{17}$$

for all $x \in \Delta$ and $\Phi \in \Re^n$.

(iv) There exists a positive constant ν such that the Jacobian matrix satisfies the Lipshitez condition, that is

$$\|F'(x) - F'(y)\| \leq \nu \|x - y\| \tag{18}$$

for all $x, y \in \Delta$.

To present the convergence of our proposed method, we require the following convergence result which is an exceptional case of a more general theorem that has been proved by Kelly (1995).

Theorem 3.1. *Let Assumption 1 holds. There are* $K_B > 0$, $\delta > 0$ *and* $\delta_1 > 0$, *such that if* $x_0 \in B(\delta)$ *and the matrix-valued function* $B(x)$ *satisfies* $\|I - B(x)F'(x^*)\| = \rho(x) < \delta_1$ *for all* $x \in B(\delta)$ *then the iteration:*

$$x_{k+1} = x_k - B(x_k)F(x_k) \tag{19}$$

converges linearly to x^*.

For the proof see Kelly (1995).

Hence, we state the following result:

Theorem 3.2. *Let* $\{x_k\}$ *be a sequence generated by* $x_{k+1} = x_k - Q_k F(x_k)$, *where* Q_k *is defined by (16). Let the Assumption 1 holds and there exist constants* $\theta > 0$, $\delta > 0$ *and* $\lambda > 0$, *such that if* $x_0 \in \Delta$ *and* Q_0 *satisfies*

$\left\|I - Q_0 F'(x^*)\right\|_F < \delta$ *for all* $x \in \Delta$, *the sequence* $\{x_k\}$ *converges linearly to* x^*.

Proof.

It is enough to show that the updating formula Q_k satisfied $\left\|I - Q_k F'(x^*)\right\|_F < \delta_k$, for some constant $\delta_k > 0$ and all k. It follows from (16), since $B_k = \frac{(y_k^T s_k - y_k^T Q_k y_k)}{Tr(A^2)} A_k$, then we obtain

$$\left\|Q_{k+1}\right\|_F \leq \left\|Q_k\right\|_F + \left\|B_k\right\|_F . \tag{20}$$

Without the lost of generality, by assuming $Q_0 = I$ and for $k = 0$ we have

$$\left\|Q_1\right\|_F \leq \left\|Q_0\right\|_F + \left\|B_0\right\|_F . \tag{21}$$

Since Q_0 is an identity matrix, hence $\left\|Q_0\right\|_F = \sqrt{n}$. From (9) when $k = 0$ we have

$$| B_0^{(i)} | = \left| \frac{y_0^T s_0 - y_0^T Q_0 y_0}{Tr(A^2)} \, y_0^{(i)^2} \right|$$

$$\leq \frac{| y_0^T s_0 - y_0^T Q_0 y_0 |}{y_0^{(\max)^2} \sum_{i=1}^{n} y^{(i)^4}} \, y_0^{(\max)^4} . \tag{22}$$

Since $\frac{y_0^{(\max)^4}}{\sum_{i=1}^{n} y_0^{(i)^4}} \leq 1$, then (22) turns into

$$| B_0^{(i)} | \leq \frac{| y_0^T F'(\hat{x}_0) y_0 - y_0^T Q_0 y_0 |}{y_0^{(\max)^2}} . \tag{23}$$

Using condition (iv) of Assumption 1 and letting $t = \max\{| t_1 |, | t_2 |\}$, (23) becomes

$$| B_0^{(i)} | \leq \frac{| t - 1 | (y_0^T y_0)}{y_0^{(\max)^2}} . \tag{24}$$

For $i = 1,...,n,$ and $y_0^{(i)^2} \leq y_0^{(\max)^2}$, it follows that

$$| B_0^{(i)} | \leq \frac{| t-1 | n y_0^{(\max)^2}}{y_0^{(\max)^2}}, \tag{25}$$

and, thus

$$\left\| B_0 \right\|_F \leq n^{\frac{3}{2}} | t-1 |. \tag{26}$$

Letting $\lambda = n^{\frac{3}{2}} | t-1 |,$ then

$$\left\| B_0 \right\|_F \leq \lambda. \tag{27}$$

Substituting (27) in (21) and let $\theta = \sqrt{n} + \lambda,$ it follows that

$$\left\| Q_1 \right\|_F \leq \theta. \tag{28}$$

Since $Q_1 = Q_0 + B_0$ and it's assumed that at $k = 0,$ $\left\| I - Q_0 F'(x^*) \right\|_F < \delta,$ then

$$\begin{aligned} \left\| I - Q_1 F'(x^*) \right\|_F &\leq \left\| I - Q_0 F'(x^*) \right\|_F + \left\| B_0 F'(x^*) \right\|_F \\ &\leq \left\| I - Q_0 F'(x^*) \right\|_F + \left\| B_0 \right\|_F \left\| F'(x^*) \right\|_F, \end{aligned} \tag{29}$$

hence $\left\| I - Q_1 F'(x^*) \right\|_F < \delta + \lambda\varphi = \delta_1,$ where $\phi = \left\| F(x^*) \right\|_F$. Therefore by induction, $\left\| I - Q_k F'(x^*) \right\|_F < \delta_k$ for all k. Hence, from Theorem 3.1, the sequence $\{x_k\}$ generated by Algorithm DBLM converges linearly to x^*.

4. NUMERICAL RESULTS

In order to evaluate the performance of the proposed method (DBLM), we apply the algorithm to seven popular problems and compared its numerical result with four (4) prominent methods. The comparison was based upon the following criterion: Number of iterations, CPU time in seconds and storage requirement. The methods are Newton method (NM), fixed Newton method (FN), Broyden method (BM) and diagonal Broyden-

like method (DBLM) that is introduced in this paper. The stopping criterion used is $\|s_k\| + \|F(x_k)\| \leq 10^{-4}$. We implemented the four methods (DBLM,NM,FN and BM) using MATLAB 7.0. All the calculations were carried out in a double precision computer. The symbol "−" indicates a failure as a result of:

(1) The CPU time in second and /or number of iteration reaches 300, but no x_k satisfies the stopping criterion;

(2) Insufficient memory to the initial the run.

We give some details of the benchmarks test problems as follow:

Problem 1 Trigonometric System of (Byeong (2010)): $f_i(x) = \cos(x_i) - 1$ $i = 1, 2, \ldots, n,$ and $x_0 = (0.5, 0.5, .., 0.5) \times \frac{\pi}{180} \times 100 = (0.87, 0.87, ..., 0.87)$

Problem 2 Artificial problem:

$f_i(x) = \ln(x_i)\cos((1-(1+(x^T x)^2)^{-1}))\exp((1-(1+(x^T x)^2)^{-1}))$ $i = 1, 2, \ldots, n,$ and $x_0 = (2.5, 2.5, .., 2.5)$

Problem 3 Artificial problem:

$f_1(x) = \cos x_1 - 9 + 3x_1 + 8\exp x_2,$ $f_i(x) = \cos x_i - 9 + 3x_i + 8\exp x_{i-1}$
$f_n(x) = \cos x_i - 1,$ $i = 2, \ldots, n-1$ and $(5, 5, \ldots, 5).$

Problem 4 Trigonometric function (Spedicato (1975)).

$$f_i(x) = n - \sum_{j=1}^{n} \cos x_j + i(1 - \cos x_i) - \sin x_i,$$

$$i = 1, ..., n \text{ and } x_0 = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})$$

Problem 5 System of n nonlinear equations (Rose (1990)). $f_i(x) = x_i - \frac{(\sum_{i=1}^{n} x_i)^2}{n^2} + \left(\sum_{i=1}^{n} x_i\right) - n,$ $i = 1, 2, \ldots, n$ and $x_0 = (10, 10, \ldots, 10).$

TABLE 1: Number of iterations and CPU time (in bracket) of Problems (1)-(5)

Prob	Dim	NM	FN	BM	DBLM
1	25	7 (0.062)	236 (0.047)	12 (0.005)	25 (0.005)
2	25	5(0.062)	— (—)	7 (0.016)	5 (0.013)
3	25	— (—)	— (—)	— (—)	11(0.015)
4	25	6 (0.031)	68 (0.062)	11 (0.016)	12 (0.004)
5	25	5 (0.047)	23 (0.031)	8 (0.014)	8 (0.004)
1	100	7 (0.872)	356 (0.168)	12 (0.018)	27 (0.010)
2	100	6 (0.593)	— (—)	7 (0.312)	5 (0.03)
3	100	— (—)	— (—)	— (—)	11 (0.016)
4	100	7 (0.842)	181 (0.718)	13 (0.312)	12 (0.023)
5	100	4 (0.421)	8 (0.078)	7 (0.046)	12 (0.031)
1	500	7 (16.988)	— (—)	13 (4.092)	29 (0.024)
2	500	6 (16.1305)	— (—)	7 (2.168)	5 (0.032)
3	500	— (—)	— (—)	— (—)	14(0.031)
4	500	7 (18.985)	415 (26.801)	14 (6.099)	11(0.028)
5	500	4 (10.561)	6 (2.824)	6 (1.732)	6 (0.016)
1	1000	7 (101.471)	— (—)	14 (7.167)	31 (0.032)
2	1000	6 (107.8747)	— (—)	7 (8.736)	5 (0.036)
3	1000	— (—)	— (—)	— (—)	15 (0.051)
4	1000	7 (131.743)	— (—)	19 (28.205)	14 (0.036)
5	1000	4 (73.851)	6 (19.126)	6 (7.691)	6 (0.031)
1	10000	— (—)	— (—)	— (—)	33 (0.113)
2	10000	— (—)	— (—)	— (—)	6 (0.172)
3	10000	— (—)	— (—)	— (—)	12 (0.327)
4	10000	— (—)	— (—)	— (—)	15 (0.359)
5	10000	— (—)	— (—)	— (—)	5 (0.160)

TABLE 1 (continued): Number of iterations and CPU time (in bracket) of Problems (1)-(5)

Prob	Dim	NM	FN	BM	DBLM
1	250000	– (–)	– (–)	– (–)	29 (6.699)
2	250000	– (–)	– (–)	– (–)	6 (3.526)
3	250000	– (–)	– (–)	– (–)	12 (7.316)
4	250000	– (–)	– (–)	– (–)	25 (12.917)
5	250000	– (–)	– (–)	– (–)	5 (2.730)

We have tasted our proposed method (DBLM) through solving five benchmark non-linear systems of equations. The results in Table 1 demonstrate very clearly that the use of DBLM method to solve all the benchmarks problems has significantly reduced the possible execution time (CPU time) from $O(n^2)$ to $O(n)$, especially as the dimension of the systems increases. In addition, the DBLM method entails very less computing exertion in building the approximation of the Jacobian inverse, hence, we claim that it is more efficient to do this than to computes the full elements of Jacobian matrix. From matrix storage requirement point of view, it is obvious that our method has significantly reduced the matrix storage from exponential order to linear order. Moreover, we observe that the DBLM method has never failed in solving the benchmark problems. These observations grant further authentication of the advantages of our diagonal updating approach to Newton's methods for solving large scale systems of nonlinear equations. Finally, these results show that the DBLM method is effective.

5. CONCLUSIONS

In this paper, we presented a new diagonal variant of the classical Newton's method for solving a large-scale system of nonlinear equations. The fundamental idea of this method is to approximate the Newton's step using diagonal updating in order to make it less computationally expensive than Newton's method and faster than fixed Newton method. The computational scheme is obtained by approximating the Jacobian inverse into a diagonal matrix by means of variational technique. Numerical testing provides a strong indication that the DBLM method the exhibits enhanced performance in all the tested problems (as measured by the CPU time and matrix storage requirement) by comparing with the other variants of Newton's methods. Lastly we conclude that the use of diagonal updating to

approximate the Newton's step is capable of improving the performance of Newton method to an acceptable level, especially when the function derivatives are relatively expensive or could not be done accurately.

REFERENCES

Broyden, C. G. 1965. A class of methods for solving nonlinear simultaneous equations. *Math. Comput.* **19** : 577-593.

Byeong, C. S., Darvishi, M.T. and Chang, H.K.. 2010. A comparison of the New-ton Krylov method with high order Newton-like methods to solve nonlinear systems. *Applied Math. Comput.* **217**: 3190-3198.

Dennis, J. E. 1983. *Numerical methods for unconstrained optimization and nonlinear equations.* Englewood Cliffs, New Jersey: Prince-Hall, Inc.

Drangoslav, H. and Natasa, K. 1996. Quasi-Newton's Method with corrections. *Novi Sad, J. Math.* **26:** 115-127.

Eisenstat, S. C. and Steihaug, T. 1982. Inexact trust region method for large sparse systems of nonlinear equations. *J. Optm. Theory Appl.* **81**: 569-590.

Hassan, M. A., Leong, W.J. and Farid, M. 2009. A new gradient method via quasi-Cauchy relation which guarantees descent. *J. Comput. Appl. Math.* **230**: 300-305.

Kelly, C.T. 1995. *Iterative Methods for Linear and Nonlinear Equations.* Philadelphia, PA: SIAM.

Leong, W.J. and Hassan, M. A. 2009. A restarting approach for the symmetric rank one update for unconstrained optimization. *Computational Optimization and Applications.* **43**: 327-334.

Leong, W.J. and Hassan, M. A. 2011a. A new gradient method via least change secant update. *International Journal of Computer Mathematics*. **88**: 816-828.

Leong, W. J., Hassan, M. A. and Farid, M. 2010. A monotone gradient method via weak secant equation for unconstrained optimization. *Taiwanese J. Math.* **14**: 413-423.

Leong, W. J., Hassan, M. A and Waziri, M. Y. 2011b. A matrix-free quasi-Newton method for solving large-scale nonlinear systems. *Comput. Math. App*. **62**: 2354-2363.

Modarres, F., Leong, W. J. and Hassan, M. A. 2010. A new two-step gradient-type method for large-scale unconstrained optimization. *Computers and Mathematics with Applications.* **59**: 3301-3307.

Modarres, F., Hassan, M. A. and Leong, W. J. 2011. Improved Hessian approximation with modified secant equations for symmetric rank-one method. *J. Comput. Appl.* **61**: 3312-3318.

Natasa, K. and Zorna. 2001. Newton-like method with modification of the right-hand vector. *Math. Compt.* **71**: 237-250.

Roose, A., Kulla, V., Lomp, M. and Meressoo, T. 1990. Test examples of systems of nonlinear equations. Tallin: *Estonian Software and Computer Service Company*.

Spedicator, E. 1975. *Computational experience with quas-Newton algorithms for min-imization problems of moderately large size Rep.* CISE-N-175, Segrate (Milano).

Waziri, M. Y., Leong, W. J., Hassan, M. A. and Monsi, M. 2010a. A new Newton's method with diagonal Jacobian approximation for systems of nonlinear equations*, J. Mathematics and Statistics.* **6**: 246-252.

Waziri, M. Y., Leong, W. J., Hassan, M. A. and Monsi, M. 2010b. Jacobian computation-free Newton method for systems of Non-Linear equations. *Journal of numerical Mathematics and stochastic*. **1**: 54-63.

Waziri, M. Y., Leong, W. J., Hassan, M. A. and Monsi, M. 2011. A Low memory Solver for integral equations of Chandrasekar Type in the Radiative Transfer Problems. *Mathematical Problems in Engineering* . **2011**: 12.