On Performance Analysis of Diagonal Variants of Newton's Method for Large -Scale Systems of Nonlinear Equations

M.Y. Waziri
Department of Mathematical science, Bayero University Kano, Nigeria

H. Aisha
Department of Mathematical science, Bayero University Kano, Nigeria

A.I. Gambo
Central Bank of Nigria, Abuja, Nigeria

ABSTRACT
In this paper, we compared and analyzed some newly diagonal variants of Newton methods for solving large -scale systems of nonlinear equations. Due to the fact that, the diagonal updating scheme is computationally less expensive than classical Newton methods and some of its variants. The two diagonal updating were introduced by Waziri et.al. [6] and Waziriet. al.[7] respectively . Reasonable analysis into the efficiency and stability of the two diagonal updating scheme are given by numerical evaluation of some benchmark nonlinear systems with Newton method and some of its variants..

Keywords
Numerical Method, solution, Newton's method.

1. INTRODUCTION
Consider the system of nonlinear equations

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

where $F = (f_1, f_2,..., f_n) : R^n \to R^n$ is continuously differentiable in a open neighborhood E of a solution $x^* \in E$ of the system (1.1). Assume that, there exists a solution x^* where $F(x^*) = 0$ and $F'(x^*) \neq 0$.

The renowned method for solving (1.1), is Newton method. Furthermore, this method generates a sequence of points $\{x_k\}$ from any specified initial guess x_0 in the neighborhood of the solution, via the following form:

Algorithm CN

Given an initial guess x_0 , and for $k = 0,1,2\ldots,$

Solve for s_k

$$F'(x_k)s_k = -F(x_k) \tag{1.2}$$

Set

$$x_{k+1} = x_k + s_k \tag{1.3}$$

where $F'(x_k)$ is the nonsingular Jacobian matrix of F , s_k is a Newton step and (1.9) is Newton system. The most attractive part of this iterative method is the convergence rate, the method has quadratic rate of convergence provided the Jacobian is not singular at a solution x^* [5].

$$\left\|x_{k+1} - x^*\right\| \leq \lambda \left\|x_k - x^*\right\|^2, \tag{1.4}$$

where $k = 0,1,2\ldots$ for some λ .

Despite its good convergence rate and simplicity to implement, Newton method requires computation and storage of the Jacobian matrix as well as solving n linear equations in each iteration, that is why an iteration of algorithm CN turns to be expensive, [5]. This is more visible when handling large-scale systems of nonlinear equations or systems in which their derivatives are quite costly. These critical weaknesses have attracted the interest of some number of scholars. Revised Newton-type and Newton-like methods where introduced, which incudes fixed Newton, quasi-Newton, inexact Newton, Newton-Krylov e.t.c. with the anticipation of diminishing the well known weakness of Newton method. Nevertheless the principal complexity of such methods is the Jacobian matrix storage prerequisite especially when solving large-scale systems. Fixed Newton has been the simplest and easiest variant of classical Newton method, the method avoids computation and storing the Jacobian matrix in each iterations(only at $k = 0$) nevertheless solves systems of n linear equations. The method is an iterative procedure that yields a sequence of points $\{x_k\}$ from an initial point x_0 in the neighborhood of x^* , using the following:

Algorithm FN (Fixed Newton method)

Given an initial guess x_0 , and for $k = 0,1,2\ldots,$

Solve for s_k

$$F'(x_0)s_k = -F(x_k) \tag{1.5}$$

Set

$$x_{k+1} = x_k + s_k \tag{1.6}$$

The method is cheaper than Newton method but it converges very slow, due to less information of the Jacobian in each iteration.

Inexact Newton method is another variant of Newton method, this method determines the approximate Newton step x_k by some iterations [see,[8] for details], via the following:

Algorithm INM(Inexact Newton)

Find some s_k which satisfies

$$F'(x_k)s_k = -F(x_k) + r_k \tag{1.7}$$

where

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

set

$$x_{k+1} = x_k + s_k \tag{1.8}$$

where η_k is a sequence of forcing terms for $0 \le \eta_k \le 1$.

The famous variant of Newton method that replaces Jacobian or its inverse with an approximation which can be updated at each iteration is quasi Newton method [4]. The method generates sequence of points $\{x_k\}$ according to following stages:

Algorithm QN (Quasi-Newton method)

Given an initial guess x_0 and for $k = 0,1,2\ldots,$

Solve for s_k

$$B_k s_k = -F(x_k) \tag{1.9}$$

Update

$$x_{k+1} = x_k + s_k, \tag{1.10}$$

where B_k is an approximation to the Jacobian. Various Jacobian approximation matrices such as the Broyden's method [1] are proposed:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T}{s_k^T s_k}, \tag{1.11}$$

where B_k is an Jacobian approximation, $y_k = F(x_{k+1}) - F(x_k)$ and $s_k = x_{k+1} - x_k$.

Still algorithm QN requires to form and store a full-matrix approximation to the Jacobian in every iteration. The method that do not need to compute neither to store the Jacobian matrix is diagonal variant of Newton method. Diagonal Jacobian update was proposed by [6] and diagonal Jacobian inverse updating formula presented by [7].

In this paper, we compared the numerical performance of the above schemes for solving large scale systems of nonlinear equations. Our foremost concern here, is on matrix storage requirement and execution time(CPU) in seconds for the two newly diagonal variants of Newton methods for solving large-scale systems of nonlinear equations, in other to draw conclusion among the two newly methods.

We organized the rest of this work as follows: the Jacobian inverse approximation is presented in section 2, Jacobian diagonal updating scheme in section 3. Numerical experiments are given in section 4, and finally Discussion and Conclusion are reported in Section 5.

2. DIAGONAL JACOBIAN UPDATING SCHEME

Here we shall consider an approximation of the Jacobian into diagonal matrix which proposed by [6]. They presented the approximation using Taylor series expansion of $F(x)$ i.e

$$F(x) = F(x_k) + F'(x_k)(x - x_k) + O(\|x - x_k\|). \tag{2.1}$$

By imposing a well known conditions on the incomplete Taylor series expansion of $F(x)$ [6] proposed an approximation to Jacobian as

$$F'(x_k) \approx D \tag{2.2}$$

where $D = diag(d^1, d^2, \ldots, d^n)$ and

$$d_{k+1}^{(i)} = \frac{F(x_{k+1})^{(i)} - F(x_k)^{(i)}}{x_{k+1}^{(i)} - x_k^{(i)}} \tag{2.3}$$

hence

$$D_{k+1} = diag(d_k^{(i)}) \tag{2.4}$$

provided $x_{k+1}^{(i)} - x_k^{(i)} \neq 0$ (See [6] for details). The updating scheme and the algorithms for approximate Jacobian matrix into diagonal matrix, is given as [6]:

$$x_{k+1} = x_k - D_k^{-1} F(x_k) \tag{2.5}$$

Algorithm DJUS [6]

Consider $F(x): \Re^n \to \Re^n$ with the same property as (1.1)

Step1 : Given x_0 and $D_0 = I_n$, set $k = 0$

Step2 : Compute $F(x_k)$

Step3 : Compute $x_{k+1} = x_k - D_k^{-1} F(x_k)$ where D_k defined by (3.4), provided

$|\, x_{k+1}^{(i)} - x_k^{(i)} \,| > 10^{-4}$ else set $d_k^{(i)} = d_{k-1}^{(i)}$ for $k = 1,2,\ldots$

Step4 : If $\|x_{k+1} - x_k\| + \|F(x_k)\| \le 10^{-4}$ stop else set $k = k+1$ and go to step 2.

3. DIAGONAL JACOBIAN INVERSE UPDATING SCHEME

We consider Jacobian inverse updating scheme into diagonal matrix proposed by [7]. The improvement of this approach over Jacobian updating approach is that, it do not require storage or computation of the true Jacobian matrix in every iteration. Moreover this guaranteed a possible reduction in execution time (CPU time) and matrix storage requirement. By using Taylor series expansion of $F(x)$ i.e

$$F(x) = F(x_k) + F'(x_k)(x - x_k) + O(\|x - x_k\|) \tag{3.1}$$

and applying the well known conditions on the incomplete Taylor series expansion of $F(x)$ [7] presented the inverse Jacobian scheme as

$$F'(x_k)^{-1} \approx D \tag{3.2}$$

where $D = diag(d^1, d^2, \ldots, d^n)$ and

$$d_{k+1}^{(i)} = \frac{x_{k+1}^{(i)} - x_k^{(i)}}{F(x_{k+1})^{(i)} - F(x_k)^{(i)}} \tag{3.3}$$

hence

$$D_{k+1} = diag(d_k^{(i)}) \tag{3.4}$$

provided $F(x_{k+1})^{(i)} - F(x_k)^{(i)} \neq 0$ (See [7] for details). [7] presented the updating scheme and the algorithms as follows:

$$x_{k+1} = x_k - D_k F(x_k) \quad (3.5)$$

Algorithm DJIUS [7]

Consider $F(x): \Re^n \to \Re^n$ with the same property as (1.1)

Step 1 : Given x_0 and $D_0 = I_n$, set $k = 0$

Step 2 : Compute $F(x_k)$

Step 3 : Compute $x_{k+1} = x_k - D_k F(x_k)$ where D_k defined by (3.4), provided

$| F(x_{k+1})^{(i)} - F(x_k)^{(i)} | > 10^{-4}$ else set $d_k^{(i)} = d_{k-1}^{(i)}$ for $k = 1,2,\ldots$

Step 4 : If $\|x_{k+1} - x_k\| + \|F(x_k)\| \le 10^{-4}$ stop else set $k = k+1$ and go to step 2.

4. NUMERICAL RESULTS

In order to compare the performance of the two diagonal variants of Newton method for solving large scale systems of nonlinear equations which we denote by DJUS and DJIUS. We apply the two algorithms to four benchmark problems and compare their numerical performance with the Newton's method (CN) and fixed Newton method(FN) respectively. The comparison is based on CPU time in seconds and matrix storage requirement. The computations experiments are done in MATLAB 7.0 using double precision computer. We used the following stopping criterion

$$\|x_{k+1} - x_k\| + \|F(x_k)\| \le 10^{-4} \quad (4.1)$$

We introduced the symbol "–" to indicate a failure.

In the following we describe the test problems as

Problem 1 System of n nonlinear equations :

$$f_i(x) = \sin(1 - x_i)\sum_{i=1}^{n} x_i^2 + 2x_{n-1} - 3x_{n-2} - \frac{1}{2}x_n f_3(x) = \alpha\left(n - \sum_{j=1}^{n}\cos x_j + i(1-\cos x_i)\right) - \alpha \sin x_i,$$

$$- x_i \ln(9 + x_i) - \frac{9}{2}\exp(1 - x_n) + 2$$

$i = 1,2,\ldots,n, \qquad x_0 = (3,3,..,3)^T$

Problem 2 Trigonometric System of Byeong [9] :

$$f_i(x) = \cos(x_i) - 1$$

$i = 1,2,\ldots,n, \quad and \quad x_0 = (0.87,0.87,...,0.87)$

Problem 3 System of n nonlinear equations :

$$f_i(x) = \ln(x_i)\cos((1-(1+(x^T x)^2)^{-1}))$$

$$\exp((1-(1+(x^T x)^2)^{-1}))$$

$i = 1,2,\ldots,n, \qquad and \qquad x_0 = (2.5,2.5,..,2.5)$.

Problem 4 Trigonometric system :

$$f_1(x) = \cos x_1 - 9 + 3x_1 + 8\exp x_2,$$

$$f_i(x) = \cos x_i - 9 + 3x_i + 8\exp x_{i-1}$$

$$f_n(x) = \cos x_i - 1$$

$i = 2,\ldots,n-1 \qquad and \qquad (5,5,\ldots,5).$

Problem 5 System of n nonlinear equations :

$$f_i(x) = (1 - x_i^2) + x_i(1 + x_i x_{n-2} x_{n-1} x_n) - 2$$

$i = 1,2,\ldots,n \qquad and \qquad x_0 = (2,2,...,2)$.

Problem6 Spares System of Byeong [9] :

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

$$f_n(x) = x_n x_1 - 1$$

$i = 1,2,\ldots,n-1 \qquad and \qquad x_0 = (0.5,0.5,..,.5)$

.

Problem 7 System of n nonlinear equations :

$$f_i(x) = n(x_i - 3)^2 + \frac{\cos(x_i - 3)}{2} - \frac{x_i - 2}{\exp(x_i - 3) + \log(x_i^2 + 1)}$$

. $i = 1,2,\ldots,n \qquad and \qquad x_0 = (-3,-3,-3,\ldots,-3)$

Problem 8 Generalized Trigonometric function of Spedicator [10]

$i = 1,...,n \qquad , \alpha = 200 \qquad and \qquad x_0 = (\frac{1}{n}, \frac{1}{n}, ... \frac{1}{n})$

Problem9 System of n nonlinear equations :

$$f_i(x) = 3n - \left(\sum_i^n (\cos_i - 2)\right) - \left(\sum_i^n x_i + \sin(x_i - 2)\right)$$

$i = 1,2,\ldots,n, \qquad and \qquad x_0 = (0,0,...0)$.

Table 1, Results of Problem 1-9 (CPU Time in Seconds)

prom	*n*	CN	FN	DJUS	DJIUS
1	25	-	-	0.031	0.030
2	25	0.062	0.047	0.001	0.001
3	25	0.062	-	0.015	0.012
4	25	-	-	0.031	0.016
5	25	0.031	0.031	0.004	0.002
6	25	0.015	-	0.001	0.001
7	25	0.190	-	0.014	0.014
8	25	0.062	-	0.014	0.009
9	25	0.062	-	0.001	0.001
1	50	-	-	0.031	0.030
2	50	0.094	0.140	0.004	0.004
3	50	0.109	-	0.016	0.015
4	50	-	-	0.034	0.031
5	50	0.156	0.062	0.006	0.004
6	50	-	-	0.015	0.015
7	50	0.328	-	0.030	0.030
8	50	0.156	-	0.016	0.012
9	50	0.125	-	0.005	0.004
1	500	-	-	0.062	0.047
2	500	16.988	-	0.024	0.022
3	500	16.1305	-	0.032	0.031
4	500	-	-	0.064	0.039
5	500	15.257	-	0.040	0.031
6	500	-	-	0.023	0.019
7	500	39.578	-	0.078	0.069
8	500	8.097	-	0.033	0.030
9	500	13.104	-	0.033	0.030
1	1000	-	-	0.090	0.064
2	1000	101.471	-	0.040	0.028
3	1000	107.8747	-	0.032	0.031
4	1000	-	-	0.094	0.062
5	1000	108.340	-	0.052	0.038
6	1000	-	-	0.028	0.021
7	1000	280.830	-	0.140	0.094
8	1000	90.309	-	0.038	0.039

9	1000	114.567	-	0.036	0.034
1	10000	-	-	0.374	0.343
2	10000	-	-	0.207	0.141
3	10000	-	-	0.203	0.125
4	10000	-	-	0.562	0.437
5	10000	-	-	0.281	0.250
6	10000	-	-	0.109	0.098
7	10000	-	-	1.1078	1.061
8	10000	-	-	0.516	0.344
9	10000	-	-	0.321	0.289

An inspection of Tables 1, one would observe that DJIUS method outperforms CN, FN and DJUS methods in terms of CPU time consumptions, computational complexities and floating points operations. This is due to low computational cost associated with the building the approximation of the Jacobian inverse into nonsingular diagonal matrix. In addition, we observe that DJUS and DJIUS methods are the best with 100% of successes when compared with CN method having 63.9% and FN method with 44.54 % respectively. It is worth mentioning that the DJIUS has total eliminates the need of Jacobian matrix storage, whereby DJUS method has reduces to vector storage, respectively. The numerical comparison further reveals that the DJUS and DJIUS methods CPU time increases on

5. CONCLUSION

In this paper a comparative analysis of two diagonal methods are presented. The results shows DJIUS method is superior than CN, FN and DJUS methods. The fact that the DJIUS method solves the problems without the cost of computing and storing the Jacobian makes it clear the advantage over CN, FN and DJUS methods. It is also worth mentioning that the method is capable of significantly reducing the execution time (CPU time), as compared to CN, FN and DJUS methods. Another fact that makes the DJIUS and DJUS methods more appealing is that throughout the numerical experiments they never fail to converge. Hence we can claim that diagonal updating scheme is a good alternative to Newton-type methods for solving large-scale systems of nonlinear equations.

6. REFERENCES

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

[2] C.G. Broyden Quasi-Newton methods and their applications to function minimization, Math. Comput., 21 (1967), 368-381.

[3] C.T. Kelley Iterative Methods for Linear and Nonlinear Equations", SIAM, Philadelphia, PA, 1995.

[4] C. T. Kelly and J. I. Northrup A point wise quasi-Newton method for integral equations, SIAM J. Numer. Anal., 25 (1988), 1138-1155.

[5] J. E. Dennis and R.B. Schnabel Numerical methods for unconstrained optimization and nonlin- ear equations", Prince-Hall, Inc., Englewood Cliffs, New Jersey (1983).

[6] Waziri, M.Y., Leong, W.J., Hassan,M.A., Monsi, M., 2010 A New Newton method with diagonal Jacobian approximation for systems of Non-Linear equations. Journal of Mathematics and Statistics Science Publication. 6 :(3)9

[7] M.Y.Waziri, Leong, W.J., Hassan,M.A., Monsi, M., 2010 Jacobian computation-free Newton method for systems of Non-Linear equations. Journal of numerical Mathematics and stochastic.2 :1 : 54-63.

[8] Luksan M., Eisenstat S.C and Steihaug T. 1982, Inexact trust region method for large sparse systems of nonlinear equations, JOTA. 81 ,569-590.

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

[10] Spedicato, E. 1975. Computational experience with quasi-Newton algorithms for minimization problems of moderately large size. Rep. CISE-N-175 3: 10-41.