Journal of Novel Applied Sciences
Available online at www.jnasci.org
©2017 JNAS Journal-2017-6-2/39-51
ISSN 2322-5149 ©2017 JNAS

A Multistep Broyden's-Type Method for Solving System of Nonlinear Equations

M.Y.Waziri[1*], M.A. Aliyu[2] and A.Wakili[3]

1- Department of Mathematical Sciences, Faculty of Sciences, Bayero University, Kano, Kano State, Nigeria
2- Department of Mathematical, Faculty of Scienves, Gombe State, Gombe State
3- Department of Mathematical Sciences, Faculty of Sciences, Federal University, Lokoja, Koji State, Nigeria

Corresponding author: M.Y.Waziri

ABSTRACT: The paper proposes an approach to improve the performance of Broyden's method for solving systems of nonlinear equations. In this work, we consider the information from two preceding iterates rather than a single preceding iterate to update the Broyden's matrix that will provide sufficient information in approximating of the Jacobian matrix in each iteration. Under some suitable assumption, the convergence analysis is established. The numerical results verify that the proposed method has clearly enhanced the numerical performance of Broyden's Method.

Keywords: Multi-Step Broyden, nonlinear systems of equations, Performance profile.

INTRODUCTION

Solving systems of nonlinear equations is becoming more essential in the analysis of complex problems in many research areas. The problem considered is to find the solution of nonlinear equations

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

Where $F : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ has continuous first partial derivatives. The assumption here is there exist a vector $x^* = \left(x_1^*, x_2^*, \cdots, x_n^*\right)$ with $F\left(x^*\right) = 0$ and $F'\left(x^*\right) \neq 0,$ where $F'\left(x_k\right)$ is the Jacobian matrix of F at x_k is assumed to be locally Lipschitz continuous at $x^*.$

Newton's method is a well know method for solving $(1).$ The method generates an iterative sequence $\{x_k\}$ from a given initial guess x_0 via

$$x_{k+1} = x_k - F'(x_k)^{-1} F(x_k) \quad (2)$$

where $k = 0,1,2\cdots.$

However, despite the fact that the Newton's method is simple to implement and has a quadratic rate of convergence, still it requires the computation and storage of $n \times n$ Jacobian matrix and its inverse, and also requires solving systems of linear equations at each iteration. In practice, computations of some functions derivatives are quiet costly and sometimes they are not available or could not be done precisely. In this case, Newton's method cannot be used directly.

It is imperative to mention that some efforts have already been carried out in order to reduce or eliminate the well-known shortcomings of Newton's method for solving systems of nonlinear equations. Such methods include Chord's Newton's method, inexact Newton's method, quasi-Newton's method [13,12,11,7]. One of the widely used quasi-Newton method for solving system of nonlinear equations is Broyden's method [1]. The method approximates the

Jacobian matrix and its inverse by a derivative free matrix (known as Broyden's matrix) and therefore calculating the derivative in every iteration is avoided.
Broyden's method uses the information from the last iterate to compute the current point which makes it has lesser information of the previous iterates. It will be nice if we can incorporate the information of the last two or more iterates and update the Broyden's matrix. By doing this, the Broyden matrix will have sufficient information from the previous iterates and hence improve the accuracy of the Jacobian approximation. This motivated the paper.
Waziri *et al* [10] incorporated the concept of two-step into diagonal updating and solve system of nonlinear equations and they succeeded in enhancing the method developed by Leong *et al* [15]. The main reason for developing this approach is to improve the accuracy Broyden's matrix via multi-step in which we use the information from two preceding iterates to compute the current point. This paper is organized as follows: the next section describes our new approach; section 3 gives the convergence analysis of our proposed method. The numerical results are presented in section 4 and the conclusion is given in section 5.

1. MULTI-STEP BROYDEN-LIKE APPROACH

This section describes our new multi-step Broyden-like approach which generates a sequence of vectors $\{x_k\}$ via

$$x_{k+1} = x_k - B_k^{-1}F(x_k), \qquad (3)$$

Where B_k is the Broyden approximation of the Jacobian matrix updated in each iteration via multi-step approach.
Our target is to come up with a matrix B_k through a Broyden updating scheme. To achieve this, we make use of an interpolating curve in the variable space to develop a modified secant equation which was derived initially by Dennis and Wolkowicz [3]. This is made possible by considering some of the most successful two-step methods [5,4,9,6] for more detail. Through integrating this two-step information, we can present an improve secant equation as follows:

$$B_{k+1}(s_k - \alpha_k s_{k-1}) = y_k - \alpha_k y_{k-1} \qquad (3)$$

By letting $\rho_k = s_k - \alpha_k s_{k-1}$ and $\mu_k = y_k - \alpha_k y_{k-1}$ in (4), we have

$$B_k \rho_k = \mu_{k,} \qquad (4)$$

Since we incorporate the information from the last two preceding iterations instead of a preceding iteration in (4) and (5), we require to build an interpolating quadratic curves $x(v)$ and $y(v),$ where $x(v)$ interpolates the last two preceding iterates x_{k-1} and $x_k,$ and $y(v)$ interpolates the last two preceding function evaluation F_{k-1} and $F_k.$
Using the approach introduced in [5], the value of α_k in (4) can be determine by computing the values of v_0, v_1 and $v_2.$ By letting $v_2 = 0,$ $\{v_j\}_{j=0}^{2}$ can be computed as follows:

$$\begin{aligned}-v_1 &= v_2 - v_1\\ &= \|x(v_2) - x(v_1)\|_{\mathbf{B}_k}\\ &= \|x_{k+1} - x_k\|_{\mathbf{B}_k}\\ &= \|s_k\|_{\mathbf{B}_k}\\ &= \left(s_k^T B_k s_k\right)^{\frac{1}{2}} \qquad (5)\end{aligned}$$

$$\begin{aligned}-v_0 = v_2 - v_0 \\ &= \|x(v_2) - x(v_0)\|_{\mathbf{B}_k} \\ &= \|x_{k+1} - x_{k-1}\|_{\mathbf{B}_k} \\ &= \|s_k + s_{k-1}\|_{\mathbf{B}_k} \\ &= \left((s_k + s_{k-1})^T B_k (s_k + s_{k-1})\right)^{\frac{1}{2}} \qquad (6)\end{aligned}$$

Let us define β_k by

$$\beta_k = \frac{v_2 - v_0}{v_1 - v_0}, \qquad (7)$$

Therefore ρ_k and μ_k can be computed by the following relations

$$\rho_k = s_k - \frac{\beta_k^2}{1+2\beta_k} s_{k-1}, \qquad (8)$$

$$\mu_k = y_k - \frac{\beta_k^2}{1+2\beta_k} y_{k-1}, \qquad (9)$$

That is, α_k in (4) is given by $\alpha_k = \beta_k^2 / (1+2\beta_k).$ Using the same approach as in [10] and considering the Broyden's update as in [8], the Multi-Step Broyden's matrix is obtained as

$$B_{k+1} = B_k + \frac{(\mu_k - B_k\rho_k)\rho_k^T}{\rho_k^T \rho_k} \qquad (10)$$

Now, the algorithm for our method is as follows;

Algorithm 2.1 (Multi-Step Broyden's method (MSBM))

```
$Step\ 1:$ $Choose\ an\ initial\ guess\ set\ B_0 = I_n\ and\ let\ k := 0$
$Step\ 2:$ $Compute\ F(x_k).\ If\ \|F(x_k)\| \le \varepsilon\ stop,\ where\ \varepsilon = 10^{-4}.$
$Step\ 3:$ $If\ k := 0\ define\ x_1 = x_0 - B_0^{-1}F(x_0).\ Else\ if\ k := 1\ set\ \rho_k = s_k\ and\ \mu_k = y_k\ and\ go\ to\ 5.$
$Step\ 4:$ $If\ k \ge 2\ compute\ v_1, v_0\ and\ \beta_k\ via\ (6)-(8)\ respectively\ and\ find\ \rho_k\ and\ \mu_k\ using\ (9)\ and\ (10)$
        $respectively.\ If\ \rho_k^T\mu_k \le 10^{-4}\|\rho_k\|_2\|\mu_k\|_2\ set\ \rho_k = s_k\ and\ \mu_k = y_k.$
$Step\ 5:$ $Let\ x_{k+1} = x_k - B_k^{-1}F(x_k)\ and\ update\ B_{k+1}\ as\ defined\ by\ (11).$
$Stepv6:$ $Check\ if\ \|\rho_k\|_2 \ge \varepsilon,\ if\ yes\ retain\ B_{k+1},\ that\ is\ computed\ by\ step\ 5.\ Else\ set,\ B_{k+1} = B_k.$
$Step\ 7:$ $Set\ k := k+1\ and\ go\ to\ 2.$
```

2. CONVERGENCE ANALYSIS

This section discusses the convergence analysis of our method. The following theorem is stated without proof as it will be useful in proving the superlinear convergence of the Multi-step Broyden update.

Theorem 3.1 [14]. *Let* $F : \mathbb{R}^n \to \mathbb{R}^n$ *be continuously differentiable in an open, convex set D in* $\mathbb{R}^n$, *and assume that for some* x^* *in D,* F' *is continuous at* x^* *and* $F'(x^*)$ *is nonsingular. Let* B_k *in* $L(\mathbb{R}^n)$ *be a sequence of* *nonsingular matrices and suppose for some* x_0 *in D the iterates generated by* $x_{k+1} = x_k - B_k^{-1}F(x_k),\ x_k$ *remains in D and converges to* x^* *at a superlinear rate if and only if* $\lim_{k\to\infty} \frac{\left\| \left[B_k - F^{'}(x^*) \right]\left(x_{k+1} - x_k \right) \right\|}{x_{k+1} - x_k} = 0.$

Now, we show the superlinear convergence of our method

Theorem3.2 *Let* $F : \mathbb{R}^n \to \mathbb{R}^n$ *be continuously differentiable function in an open convex set* $C.$ *Assume that there exist positive constants* α *and* β *such that* $\left\| x_0 - x^* \right\| < \alpha$ *and* $\left\| B_0 - F^{'}(x^*) \right\| < \beta.$ *Then the sequence* $\{x_k\}$ *generated by Multi-step Broyden's update*

$$x_{k+1} = x_k - B_k^{-1}F(x_k) \tag{11}$$

$$B_{k+1} = B_k + \frac{\left(\mu_k - B_k\rho_k \right)\rho_k^T}{\rho_k^T \rho_k} \tag{12}$$

is well define and convergence to x^* superlinearly.

Proof: By theorem (3.1), it is enough to show under the conditions of the theorem that

$$\lim_{k\to\infty} \frac{\left\| \left[B_k - F^{'}(x^*) \right]\left(x_{k+1} - x_k \right) \right\|}{x_{k+1} - x_k} = 0 \tag{13}$$

From (12) and (13),

$$B_{k+1} - F'(x^*) = B_k - F'(x^*) + \frac{\left(\mu_k - B_k\rho_k \right)\rho_k^T}{\rho_k^T \rho_k}$$

$$= B_k - F'(x^*) + \frac{\left(F'(x^*)\rho_k - B_k\rho_k \right)\rho_k^T}{\rho_k^T \rho_k} + \frac{\left(\mu_k - F'(x^*)\rho_k \right)\rho_k^T}{\rho_k^T \rho_k}$$

$$= \left(B_k - F'(x^*) \right)\left[I - \frac{\rho_k\rho_k^T}{\rho_k^T \rho_k} \right] + \frac{\left(\mu_k - F'(x^*)\rho_k \right)\rho_k^T}{\rho_k^T \rho_k} \tag{14}$$

Taking norms on both sides gives

$$\left\| B_{k+1} - F'(x^*) \right\| \le \left\| B_k - F'(x^*) \right\| \left\| I - \frac{\rho_k\rho_k^T}{\rho_k^T \rho_k} \right\| + \frac{\left\| \mu_k - F'(x^*)\rho_k \right\|}{\left\| \rho_k \right\|} \tag{15}$$

Since

$$\left\| I - \frac{\rho_k\rho_k^T}{\rho_k^T \rho_k} \right\| = 1 \tag{16}$$

And by the fact that $\left\| F(u) - F(v) - F'(x)(u-v) \right\| \le \frac{\gamma}{2}\left(\left\| u - x \right\| + \left\| v - x \right\| \right)\left\| u - v \right\|$

We have

$$\left\| \mu_k - F'(x^*)\rho_k \right\| = \left\| F(x_{k+1}) - F(x_k) - F'(x^*)\rho_k \right\| \le \frac{\gamma}{2}\left(\left\| x_{k+1} - x^* \right\| + \left\| x_k - x^* \right\| \right)\left\| \rho_k \right\| \tag{17}$$

Hence $\left\| B_{k+1} - F'(x^*) \right\| \le \left\| B_k - F'(x^*) \right\| + \frac{\gamma}{2}\left(\left\| x_{k+1} - x^* \right\| + \left\| x_k - x^* \right\| \right).$

Now let $E_k = B_k - F'(x^*).$ From (16), we have

$$\|E_{k+1}\|_F \le \left\| E_k \left(I - \frac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right) \right\|_F + \frac{\left\| \left(\mu_k - F'(x^*)\rho_k \right) \rho_k^T \right\|_F}{\rho_k^T \rho_k} \tag{18}$$

Since

$$\left\| E_k \frac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right\|_F^2 = tr\left[\left(\frac{E_k \rho_k \rho_k^T}{\rho_k^T \rho_k} \right)^T \frac{E_k \rho_k \rho_k^T}{\rho_k^T \rho_k} \right]$$

$$= tr\left[\frac{\rho_k \left(E_k \rho_k \right)^T \left(E_k \rho_k \right) \rho_k^T}{\|\rho_k\|^4} \right]$$

$$= \frac{\|E_k \rho_k\|^2}{\|\rho_k\|^4} tr\left(\rho_k \rho_k^T \right) = \frac{\|E_k \rho_k\|^2}{\|\rho_k\|^4} \|\rho_k\|^2 = \frac{\|E_k \rho_k\|^2}{\|\rho_k\|^2}$$

And $E_k = E_k \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} + E_k - E_k \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} = E_k \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} + E_k \left(I - \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right)$

We get $tr\left(E_k \right) = tr\left[E_k \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} + E_k \left(I - \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right) \right],$ that is, $\|E_k\|_F^2 = \left\| E_k \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right\|_F^2 + \left\| E_k \left(I - \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right) \right\|_F^2$

Therefore $\|E_k\|_F^2 = \dfrac{\|E_k \rho_k\|^2}{\|\rho_k\|^2} + \left\| E_k \left(I - \dfrac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right) \right\|_F^2$

Hence

$$\left\| E_k \left(I - \frac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right) \right\|_F = \left(\|E_k\|_F^2 - \frac{\|E_k \rho_k\|^2}{\|\rho_k\|^2} \right)^{1/2} \tag{19}$$

Since $\left(\alpha^2 - \beta^2 \right) \le \alpha - \dfrac{\beta^2}{2\alpha}$ for any $\alpha \ge |\beta| \ge 0,$ (20) implies that

$$\left\| E_k \left(I - \frac{\rho_k \rho_k^T}{\rho_k^T \rho_k} \right) \right\|_F \le \|E_k\|_F - \frac{\|E_k \rho_k\|^2}{2\|E_k\|_F \|\rho_k\|^2}. \tag{20}$$

Now, by using (21), (18), and the fact that

$$\|x_{k+1} - x^*\| \le \frac{1}{2} \|x_k - x^*\| \quad \forall k \ge 0. \tag{21}$$

We can write (19) as $\|E_{k+1}\|_F \le \|E_k\|_F - \dfrac{\|E_k \rho_k\|^2}{2\|E_k\|_F \|\rho_k\|^2} + \dfrac{3}{4}\gamma \|x_k - x^*\|,$ which is

$$\frac{\|E_k \rho_k\|^2}{\|\rho_k\|^2} \le 2\|E_k\|_F \left[\|E_k\|_F - \|E_{k+1}\|_F + \frac{3}{4}\gamma \|x_k - x^*\| \right] \tag{22}$$

But from $\left\|B_k - F'(x^*)\right\| \le \left(2-2^{-k}\right)\delta$ and (22), we have that $\|E_k\|_F \le 2\delta,\ \forall k \ge 0$ and $\sum_{k=0}^{\infty}\left\|x_k - x^*\right\| \le 2\varepsilon.$ Thus, (23) can be written as

$$\frac{\|E_k\rho_k\|^2}{\|\rho_k\|^2} \le 4\delta\left[\|E_k\|_F - \|E_{k+1}\|_F + \frac{3}{4}\gamma\left\|x_k - x^*\right\|\right]. \tag{23}$$

By summing both sides, we obtain

$$\sum_{k=0}^{i}\frac{\|E_k\rho_k\|^2}{\|\rho_k\|^2} \le 4\delta\left[\|E_0\|_F - \|E_{i+1}\|_F + \frac{3}{4}\gamma\sum_{k=0}^{i}\left\|x_k - x^*\right\|\right]$$

$$\le 4\delta\left[\|E_0\|_F + \frac{3}{2}\gamma\varepsilon\right]$$

$$\le 4\delta\left[\|E_0\|_F + \frac{3}{2}\gamma\varepsilon\right],$$

Which hold $\forall i \ge 0.$ Therefore $\sum_{k=0}^{\infty}\frac{\|E_k\rho_k\|^2}{\|\rho_k\|^2} < +\infty \Rightarrow \frac{\|E_k\rho_k\|}{\|\rho_k\|} \to 0$ as $k \to \infty.$ Hence the proof is complete.

3. NUMERICAL RESULTS

In this section, we analyze and compare the performance of **MSBM** with that of Newton's method **NM**, Fixed Newton's method **FNM** and Broyden's method **BM** for solving systems of nonlinear equations. The algorithms are written in MATLAB7.10.0 (R2010a) and are tested for some classical benchmark problems. All the problems were run on a PC with AMD E1-1200APU with Radeon(tm) CPU with 2.00Ghz speed. To describe the results of these experiments we give the dimension of problem (N), the number of iterations performed(NI) and the CPU time (seconds).

We declare a termination of the methods whenever,

$$\|F(x_k)\| \le 10^{-4} \tag{24}$$

The identity matrix has been chosen as an initial approximate Jacobian.

The symbol $"-"$ is used to indicate a failure due to:

(1) The number of iteration is at least 500 but no point of x_k that satisfies (12) is obtained;
(2) CPU time in second reaches 500;
(3) Insufficient memory to initiate the run.

Dolan and More´[2] gave a new tool of analyzing the efficiency of Algorithms. They introduced the notion of a performance profile as a means to evaluate and compare the performance of set of solvers S on a set P. Assuming there exist *ns* solvers and *np* problems, for each problem p and solvers s, they defined $t_{p,s} =$ computing time (the number of function evaluations or others) required to solve problem p by solvers s.

Requiring a base line for comparisons, they compared the performance on problem p but solver s with the best performance by any solver on this problem; using the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min t_{p,s} : s \in S} \tag{25}$$

Suppose that a parameter $r_M \ge r_{p,s}$ for all p, s is chosen, and $r_{p,s} = r_M$ if and only if s does not solve problem p.

The performance of solver s on any given problem might be of interest, but we would like to obtain an overall assessment of the performance of the solver, then they defined

$$\rho_s(t) = \frac{1}{n_p} size p \in P : r_{p,s} \le t \tag{26}$$

Thus $\rho_s(t)$ is the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ is within a factor $t \in R$ of the best possible ration. Then function ρ_s is the (cumulative) distribution function for the performance ration. The performance profile $\rho_s : R \to [0,1]$ for a solver is a nondecreasing, piecewise constant function, continuous from the right at each breakpoint. The value of $\rho_s(1)$ is the probability that the solver win over the rest of the solvers.
According to the above rules, we know that one solver whose performance profile plot is on the top right will win over the rest of the solvers.
Below are the benchmarks problems used to test the proposed methods in this research.

problem 1(Artificial Function)

$$f_i(x) = \cos\left(x_i^2 - 1\right) - 1, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(-0.5, -0.5, \cdots, -0.5\right)^T$$

problem 2(Trigonometric system of Beyong, 2010)

$$f_i(x) = \cos x_i - 1, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(-0.5, -0.5, \cdots, -0.5\right)^T$$

problem 3(Beyong *et al*, 2010)

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

$$f_n(x) = x_n x_1 + 1, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 4(Artificial Function)

$$f_i(x) = \left(\sin x_i \cos x_i\right) x_i - \left(\cos x_i - x_i - 1\right) x_i, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 5(Beyongetal, 2010)

$$f_i(x) = x_i^2 - 1, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 6

$$f_i(x) = \sin\left(x_1 - x_2\right) - 4\exp\left(2 - x_2\right) + 2x_1,$$

$$f_n(x) = \sin\left(2 - x_i\right) - 4\exp\left(x_i - 2\right) + 2x_1 + \cos\left(2 - x_i\right) - \exp\left(2 - x_i\right), \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 7(Artificial Function)

$$f_i(x) = \left(1 - x_i^2\right) + x_i\left(1 + x_i x_n - 2x_{n-1} x_n\right) - 2, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 8(Darvishi and Shin, 2011)

$$f_i(x) = e^{x_i} - 1, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 9(Artificial Function)

$$f_1(x) = x_1,$$

$$f_i(x) = \cos x_{i-1} + x_i - 1, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(2, 2, \cdots, 2\right)^T$$

problem 10(Beyong, 2011)

$$f_i(x) = x_i^2 - 4, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(2.5, 2.5, \cdots, 2.5\right)^T$$

problem 11(Beyong, 2011)

$$f_i(x) = x_i^2 + x_i - 2, \quad i = 1, 2, \cdots, n \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 12(Artificial Function)

$$f_i(x) = 4x_1 + \left(x_{i+1} - 2x_i\right) - \left(\frac{x_{i+1}^2}{3}\right),$$

$$f_n(x) = 4x_n + \left(x_{n-1} - 2x_n\right) - \left(\frac{x_{n-1}^2}{3}\right), \quad i = 1, 2, \cdots, n-1 \text{ and } x_0 = \left(0.5, 0.5, \cdots, 0.5\right)^T$$

problem 13(Artificial Function)

$$f_i(x) = x_i^2 + (x_i - 3)\log(x_i + 3) - 9, \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.9, 0.9, \cdots, 0.9)^T$$

problem 14(Artificial Function)

$$f_1(x) = \cos x_1 - 9 + 3x_1 + 8e^{x_2},$$

$$f_i(x) = \cos x_i - 9 + 3x_i + 8e^{x_{i-1}}, \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

problem 15(Artificial Function)

$$f_i(x) = (0.5 - x_i)^2 + x_{n+1-i}^2 - 0.25x_i - 1, \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

problem 16(Hafizand Baghat, 2012)

$$f_i(x) = x_i^2 - \cos(x_i - 1), \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

problem 17(Artificial Function)

$$f_i(x) = x_i - 3x_i\left(\frac{\sin x_i}{3} - 0.66\right) + 2, \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

problem 18(Artificial Function)

$$f_i(x) = \exp(x_i^2 - 1) - \cos(1 - x_i), \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

problem 19(Artificial Function)

$$f_i(x) = (x_i^2 - 1)^2 - 2, \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

Problem20(Rooseetal.,1990)

$$f_i(x) = x_i - \frac{1}{n^2}\left(\sum_{i=1}^{n} x_i\right)^2 + \left(\sum_{i=1}^{n} x_i\right) - n, \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

Problem21(System of *n* nonlinear equations)

$$f_i(x) = 3n - \left(\sum_{i=1}^{n}(\cos x_i - 2)\right) + \left(\sum_{i=1}^{n} x_i + \sin(x_i - 2)\right), \quad i = 1, 2, \cdots, n \text{ and } x_0 = (2, 2, \cdots, 2)^T$$

Problem22(Trigonometric System)

$$f_1(x) = x_1^2 - 3x_1 + 1 + \cos(x_1 - x_2),$$

$$f_i(x) = x_i^2 - 3x_i + 1 + \cos(x_i - x_{i-1}), \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

problem 23(System of *n* nonlinear equations)

$$f_i(x) = x_i^2 - 0.1x_{i+1}^2,$$

$$f_n(x) = x_i^2 - 0.1x_1^2, \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

Problem24(System of *n* nonlinear equations)

$$f_i(x) = x_i\left(\cos x_i - \frac{1}{n}\right) - x_n\left[\sin x_i - 1 - (x_i - 1)^2 - \frac{1}{n}\sum_{i=1}^{n} x_i\right], \quad i = 1, 2, \cdots, n \text{ and } x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

Problem25(System of *n* nonlinear equations)

$$f_1(x) = 2x_1 - x_2 + uh^2\log\left[\cosh(x_1 - 1)\right] - 1,$$

$$f_i(x) = 2x_i - x_{i-1} + uh^2\log\left[\cosh(x_i - 1)\right] - 1,$$

$$f_n(x) = 2x_n - x_{n-1} + uh^2\log\left[\cosh(x_n - 1)\right] - 1,$$

$$i = 1, 2, \cdots, n \quad x_0 = (0.5, 0.5, \cdots, 0.5)^T \quad u = 10 \text{ and } h = \frac{1}{n+1}$$

Below we give the tables and graphs that show the performance of Multi-step Broyden method in comparison with Newton method (NM), Broyden's method (BM) and Fixed Newton method (FNM). We denote by MSBM the method define in algorithm (2.1).

Table1. Numerical Results of the methods when solving problems 1-6

		NM		FNM		BM		MSBM	
Problems	N	NI	CPU Time	NI	CPU Time	NI	CPU Time	NI	CPU Time
P1	25	14	0.0547	-	-	6	0.0001	4	0.0001
	50	14	0.1388	-	-	6	0.0312	4	0.0312
	100	15	0.362	-	-	6	0.1872	4	0.0468
	500	16	17.9348	-	-	7	1.2168	6	1.4196
	1000	16	133.2068	-	-	7	6.2088	6	7.1604
P2	25	15	0.0564	-	-	10	0.0312	8	0.0312
	50	16	0.1226	-	-	10	0.0468	8	0.0312
	100	16	0.3777	-	-	11	0.156	8	0.1572
	500	-	-	-	-	11	1.8408	8	1.638
	1000	-	-	-	-	12	10.2337	8	8.5489
P3	25	5	0.0194	-	-	5	0.0312	3	0.0001
	50	-	-	-	-	5	0.0648	3	0.0312
	100	-	-	-	-	5	0.0936	4	0.0314
	500	-	-	-	-	5	0.7644	4	0.9828
	1000	-	-	-	-	5	3.6348	4	4.524
P4	25	-	-	15	0.0639	3	0.0312	2	0.0001
	50	-	-	16	0.1475	3	0.0312	2	0.0013
	100	-	-	16	0.5388	3	0.0312	2	0.0312
	500	-	-	17	18.5478	4	0.5928	3	0.7176
	1000	-	-	18	137.7307	5	3.4788	3	3.5568
P5	25	5	0.017	-	-	5	0.0156	3	0.0312
	50	5	0.0558	-	-	5	0.0312	3	0.0312
	100	5	0.1403	-	-	5	0.0936	4	0.0468
	500	5	5.601	-	-	5	0.8112	4	0.8736
	1000	5	40.1397	-	-	5	3.5412	4	4.6956
P6	25	5	0.023	-	-	7	0.0468	6	0.0312
	50	5	0.0684	-	-	7	0.156	6	0.0312
	100	5	0.1726	-	-	6	0.1092	5	0.156
	500	5	5.5923	-	-	7	1.17	5	1.1544
	1000	5	38.6409	-	-	6	4.7112	5	5.772

Table 2. Numerical Results of the methods when solving problems 7-13

		NM		FNM		BM		MSBM	
Problems	N	NI	CPU Time	NI	CPU Time	NI	CPU Time	NI	CPU Time
P7	25	10	0.0425	-	-	9	0.0159	9	0.0312
	50	10	0.1325	-	-	9	0.0312	9	0.0468
	100	10	0.2773	-	-	9	0.0624	9	0.0624
	500	10	11.4329	-	-	9	1.3572	9	1.872
	1000	10	83.1567	-	-	9	7.3008	9	9.8125
P8	25	4	0.0185	11	0.011	5	0.0312	4	0.0312
	50	-	-	12	0.0167	5	0.0312	4	0.0312
	100	5	0.1294	12	0.0278	5	0.0312	4	0.0312
	500	5	5.3991	13	1.2127	5	0.7958	4	0.9516
	1000	5	40.5085	13	8.4702	5	3.5256	4	4.6332
P9	25	5	0.0179	23	0.0064	11	0.0312	12	0.0312
	50	5	0.056	23	0.0195	11	0.0312	12	0.0312
	100	5	0.1683	23	0.0361	11	0.1872	11	0.078
	500	5	5.5744	24	1.3178	13	2.2308	12	2.3712
	1000	5	40.8677	24	9.1042	14	12.5581	-	-
P10	25	4	0.0143	8	0.0053	7	0.0312	7	0.0156
	50	4	0.0435	8	0.0091	7	0.0312	7	0.0312
	100	4	0.1807	8	0.035	7	0.1248	7	0.0312
	500	4	4.3721	8	1.1657	7	1.1856	7	1.482
	1000	4	31.5299	8	8.141	7	5.4756	7	7.8157
P11	25	4	0.0148	16	0.0563	6	0.0312	4	0.0156
	50	4	0.0301	17	0.0134	6	0.0312	4	0.0312

	100	4	0.1523	17	0.0946	6	0.1092	4	0.0468
	500	5	5.4596	18	1.1769	6	0.9984	4	0.9516
	1000	5	36.5684	19	7.981	6	4.5708	4	4.6488
P12	25	4	0.0178	6	0.0067	5	0.0624	3	0.0001
	50	4	0.0397	6	0.0106	5	0.1404	3	0.0156
	100	4	0.1547	7	0.0266	5	0.1404	3	0.0312
	500	4	4.2022	7	1.0869	5	0.7488	3	0.6864
	1000	4	30.5035	7	7.6241	5	3.4164	3	3.6036
P13	25	4	0.0298	8	0.0098	6	0.0021	5	0.0012
	50	4	0.0379	8	0.0326	6	0.0936	5	0.0312
	100	4	0.1263	8	0.0379	6	0.1248	5	0.0468
	500	4	4.7032	8	1.2449	6	0.936	5	1.0764
	1000	4	32.4366	8	8.7337	6	4.7892	5	5.7408

Table 3. Numerical Results of the methods when solving problems 14-20

		NM		FNM		BM		MSBM	
Problems	N	NI	CPU Time	NI	CPU Time	NI	CPU Time	NI	CPU Time
P14	25	-	-	-	-	8	0.0312	7	0.0312
	50	-	-	-	-	9	0.0624	7	0.0624
	100	-	-	-	-	9	0.078	7	0.0624
	500	-	-	-	-	9	1.5756	7	1.56
	1000	-	-	-	-	-	-	7	7.8625
P15	25	6	0.0357	-	-	6	0.0312	-	-
	50	6	0.0739	-	-	6	0.0312	-	-
	100	6	0.1593	-	-	6	0.0468	-	-
	500	6	6.7031	-	-	6	0.9672	-	-
	1000	6	48.1997	-	-	6	4.5552	-	-
P16	25	6	0.0221	-	-	5	0.0468	3	0.0156
	50	6	0.0464	-	-	5	0.0468	3	0.0312
	100	6	0.1459	-	-	5	0.0468	4	0.0312
	500	6	0.0793	-	-	5	0.6864	4	0.936
	1000	6	48.0269	-	-	5	3.6348	4	4.6956
P17	25	5	0.0439	-	-	5	0.0156	4	0.0156
	50	5	0.0572	-	-	5	0.0312	4	0.0312
	100	5	0.1859	-	-	5	0.0312	4	0.0936
	500	5	5.5123	-	-	5	0.7332	4	0.9984
	1000	5	41.1653	-	-	5	3.5412	4	4.9296
P18	25	-	-	-	-	6	0.0468	5	0.0021
	50	-	-	-	-	6	0.156	6	0.0312
	100	-	-	-	-	6	0.1872	6	0.0624
	500	-	-	-	-	7	1.0764	6	1.2948
	1000	-	-	-	-	7	5.46	6	6.8172
P19	25	-	-	-	-	-	-	6	0.0021
	50	-	-	-	-	-	-	6	0.0156
	100	-	-	-	-	-	-	6	0.0624
	500	-	-	-	-	-	-	6	1.2636
	1000	-	-	-	-	-	-	6	6.63
P20	25	4	0.0231	5	0.0728	6	0.0156	6	0.0156
	50	3	0.0263	5	0.0145	6	0.0312	5	0.0312
	100	3	0.1607	4	0.0292	6	0.0624	5	0.0312
	500	3	3.3761	4	1.18837	6	1.0452	4	0.936
	1000	3	25.6012	4	8.3199	6	4.7736	4	4.9296

Table 4. Numerical Results of the methods when solving problems 21-25

		NM		FNM		BM		MSBM	
Problems	N	NI	CPU Time	NI	CPU Time	NI	CPU Time	NI	CPU Time
P21	25	8	4.8507	-	-	6	0.0468	4	0.0156
	50	-	-	-	-	6	0.0624	4	0.0312
	100	-	-	-	-	6	0.0624	4	0.078
	500	-	-	-	-	7	1.1544	5	1.1544
	1000	-	-	-	-	7	5.6472	5	5.772
P22	25	5	0.0222	14	0.0065	7	0.0312	6	0.0156
	50	5	0.0495	15	0.013	7	0.0312	6	0.0312
	100	5	0.1478	15	0.0875	7	0.0468	-	-
	500	5	5.6137	16	1.2321	7	1.1856	-	-
	1000	5	40.3162	17	8.5024	7	5.9592	-	-
P23	25	4	0.0887	6	0.0145	3	0.0312	2	0.0056
	50	4	0.0325	6	0.0167	3	0.0312	2	0.0156
	100	4	0.1196	6	0.0624	3	0.156	2	0.0312
	500	4	4.577	7	1.1505	3	0.5304	2	0.5148
	1000	4	31.7271	7	8.1876	4	2.8236	2	2.73
P24	25	5	0.0261	-	-	6	0.0003	5	0.0002
	50	5	0.0689	-	-	6	0.0312	5	0.0156
	100	5	0.1929	-	-	6	0.1872	5	0.0468
	500	5	5.9576	-	-	6	0.9828	5	1.092
	1000	5	41.2697	-	-	7	5.4756	5	5.5692
P25	25	4	0.0187	5	0.006	4	0.0156	3	0.0156
	50	3	0.0308	4	0.0113	4	0.0312	2	0.0312
	100	3	0.1082	4	0.0598	3	0.0312	2	0.0312
	500	3	3.4229	3	1.199	3	0.4994	2	0.5928
	1000	3	24.9551	3	8.3628	3	2.0904	2	2.9016

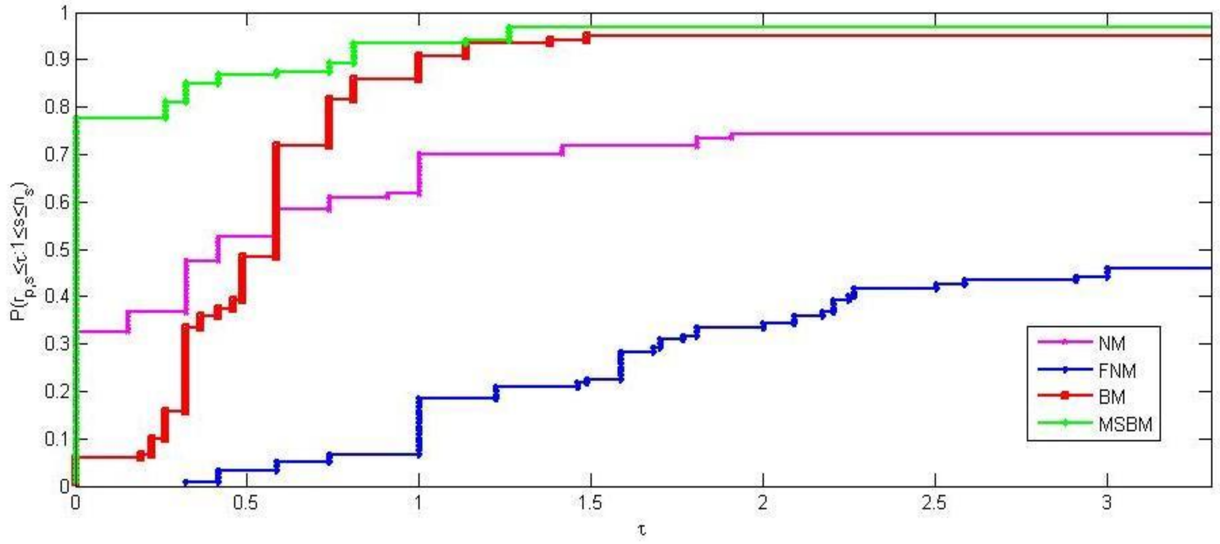

Figure 1. Performance profile of NM, FNM, BM and MSBM methods in term of Number of Iteration

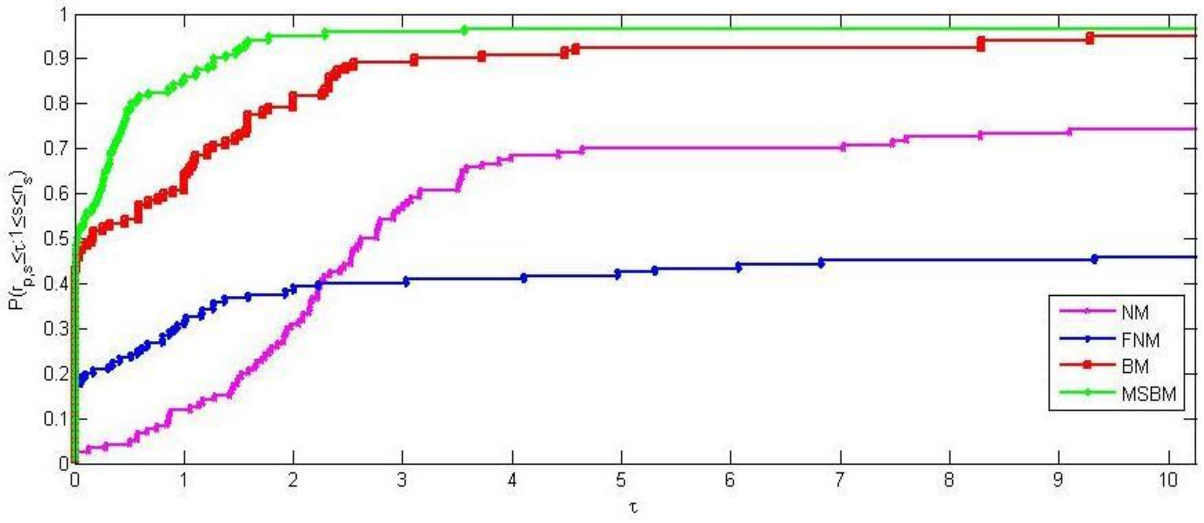

Figure 2. Performance profile of NM, FNM, BM and MSBM methods in term of CPU time

In the above figures, the left axis of the plot represents the percentage of the test problems for which a method is the best, while the right side corresponds to the percentage of the test problems that were successfully solved by these methods. Figure1 and 2 represent the performances profile in terms of the number of iteration and CPU time (seconds) respectively.

Tables 1−3 and figures 1 and 2 have shown that using the two-step approach in building the Broyden's updating scheme has significantly enhanced the performance of the classical Broyden method. This observation is glaring when considering CPU time and number of iterations (NI). In addition, it is worth mentioning that the result of MSBM in solving problem7, when the dimension increases, shows that our new approach becomes a better candidate.

5. CONCLUSION

In this paper, we have presented a new method (MSBM) for solving system of nonlinear equations. Unlike the single step, the method employs a two-step to update the non-singular Broyden's matrix in approximating the Jacobian matrix. Numerical experiments shown strong indication that our new approach requires less computational cost and number of iterations as compared to the NM, FNM and BM methods. Hence, we can wind up that our method (MSBM) is a better candidate when compared with NM, FNM and BM methods in solving system of nonlinear equations.

REFERENCES

[1] C.G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Math. Compute.* 19:577–593, 1965.
[2] E.D. Dolan and J.J. More. Benchmarking optimization software with performance profiles. *Math. Program. Ser, 91:201*–213, 2002.
[3] Jr.J.E. Dennis and H. Wolkowicz. Sizingand least-change secant methods. *SIAM Journal on Numerical Analysis*, 30(5):1291–1314, 1993.
[4] I.A. Moghrabi J.A.Ford. Multi-step quasi-newton methods for optimization. *J. Compute. Appl. Math.*, 50:305–323, 1994.
[5] I.A. Moghrabi J.A. Ford. Alternating multi-step quasi-newton methods for unconstrained optimization. *J. Compute. Appl. Math.*, 82:105–116, 1997.
[6] S. Tharmlikit J.A. Ford. New implicite updates in multi-step quasi-newton methods for unconstrained optimization. *J.Comput. Appl. Math.*, 152:133–146, 2003.
[7] M. Mamat K. Muhammad and M.Y. Waziri. Abroyden'slike method for solving systems of nonlinear equations. *World Applied Aciences Journal*, 21:168–173, 2013.
[8] C.T. Kelly. *Solving nonlinear equations with Newton's method*. SIAM, 2003.
[9] W.J. Leong M. Faridand M.A.Hassan. Anew two-step gradient-type method for large- scale unconstrained optimization, computers and mathematic switch applications. *Computers and Mathematics with Applications*, 59(10):3301–3307, 2010.
[10] W.J. Leong M.Y. Waziri and M. Mamat. A two-step matrix-free secant method for solving large-scale systems of nonlinear equations. *Journal of Applied Mathematics*, doi:10.1155/2012/348654 (Article ID: 348654):9pages, 2012.
[11] K.Natasa and L.Zorna. Newton-like method with modification of the right-hand vector. *Journal of Computational Mathematics*, (17):237–250, 2011.

[12] C.S.Eisenstat R.S.Demboand T Steihaug. Inexact newton method. *SIAM JNumer.Anal.*, 19(2):400–408,1982.
[13] J.E. Dennis R.B.Jr. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.
[14] S.WENYU and Y.YUAN. *Optimization Theory and Methods*. Springer Optimization and Its Applications, 2006.
[15] M.Y. Waziri W.J.Leong and M.A. Hassan. A matrix-freequasi-newton method for solving nonlinear systems. *Comput. Math. Appl.*, 62:2354–2363, 2011.