MJFAS MALAYSIAN JOURNAL OF FUNDAMENTAL AND APPLIED SCIENCES
PRINT ISSN: 2289-5981 | ONLINE ISSN: 2289-599X

RESEARCH ARTICLE

# Improved Quasi-Newton method via SR1 update for solving symmetric systems of nonlinear equations

Muhammad Kabir Dauda [a, b,*], Mustafa Mamat [a], Mohamad Afendee bin Mohamed [a], Mahammad Yusuf Waziri [c]

[a] Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Gong Badak, Terengganu, Malaysia
[b] Department of Mathematical Sciences, Kaduna State University, Kaduna, Nigeria
[c] Department of Mathematical Sciences, Bayero University, Kano, Nigeria

* Corresponding author: mkdfika@gmail.com

**Article history**
Received 27 February 2018
Revised 28 March 2018
Accepted 21 May 2018
Published Online 4 February 2019

**Abstract**

The systems of nonlinear equations ~~emerges~~ from many areas of computing, scientific and engineering research applications. A variety of an iterative methods for solving such systems have been developed, this include the famous Newton method. Unfortunately, the Newton method suffers setback, which includes storing $n \times n$ matrix at each iteration and computing Jacobian matrix, which may be difficult or even impossible to compute. To overcome the drawbacks that bedeviling Newton method, a modification to SR1 update was proposed in this study. With the aid of inexact line search procedure by Li and Fukushima, the modification was achieved by simply approximating the inverse Hessian matrix $B_{k+1}^{-1}$ with an identity matrix without computing the Jacobian. Unlike the classical SR1 method, the modification neither require storing $n \times n$ matrix at each iteration nor needed to compute the Jacobian matrix. In finding the solution to non-linear problems of the form $F(x) = 0, x \in R$, 40 benchmark test problems were solved. A comparison was made with other two methods based on CPU time and number of iterations. In this study, the proposed method solved 37 problems effectively in terms of number of iterations. In terms of CPU time, the proposed method also outperformed the existing methods. The contribution from the methodology yielded a method that is suitable for solving symmetric systems of nonlinear equations. The derivative-free feature of the proposed method gave its advantage to solve relatively large-scale problems (10,000 variables) compared to the existing methods. From the preliminary numerical results, the proposed method turned out to be significantly faster, effective and suitable for solving large scale symmetric nonlinear equations.

***Keywords***: SR1, global convergence, nonlinear equations



## INTRODUCTION

Given the symmetric nonlinear system as following:

$$F(x) = 0, x \in R^n \tag{1}$$

where $F: R^n \to R^n$ is continuously differentiable in an open convex set $S$ and assumed to satisfy the assumptions that (i) Its Jacobian $J(x) \approx F'(x)$ is symmetric, i.e., $J(x) = J(x)^T$.

There exists a solution vector $x^*$ of (1) in $S$ such that $F(x^*) = 0$ and $F'(x^*) \neq 0$. (iii) The Jacobian $F'(x)$ is Lipschitz continuous at $x^*$.

The systems of symmetric nonlinear equation (1) has been discussed by researchers (Li & Shengjie, 2015; Zhang & Maojun, 2015). The Newton method is famous, unfortunately, the method suffers setback which includes storing an $n \times n$ matrix at each iteration and computing Jacobian matrix which may be difficult or even not possible to compute. For more details on Newton method and other numerical methods of solving nonlinear equations see (Dauda, Mamat, Waziri, Ahmad, & Mohamad, 2016; Mamat, Dauda, Waziri, Ahmad, & Mohamad, 2016). The Newton method generates an iterative sequence $x_k$ from a given initial guess vector $x_0$ in the neighborhood of $x^*$ from the following algorithm.

**Algorithm** (Newton's Method)

For $k = 0,1,2,\ldots$ of $F'(x_k)$, the Jacobian matrix of $F$,

**Step 1:** Solve $F'(x_k)s_k = -F(x_k)$
**Step 2:** Update $x_{k+1} = x_k + s_k$, where $s_k$ is the Newton correction in

the Newton system. When the Jacobian matrix $F'(x^*)$, is nonsingular at a solution of (1) the convergence is guaranteed with a quadratic rate from any initial point $x_0$ in the neighborhood of $x^*$. Throughout this article, we always assume that the problem (1) is symmetric and equivalent to the global optimization problem(2)

$$min_{x\in R^n} f(x) \tag{2}$$

with function f in (2) is defined by

$$f(x) = \frac{1}{2}\|F(x)\|_2^2. \tag{3}$$

To approximate the gradient $\nabla f(x_k)$, which avoids computing exact gradient. It is clear that, when $F(x_k)$ is small, then $g(x_k) \approx \nabla f(x_k)$. In (D. L. Fukushima & M, 2000), Li and Fukushima used the term:

$$g_k \approx \frac{F(x_k+\alpha_k F(x_k))}{\alpha_k} \tag{4}$$

The purpose of this article was to overcome the drawbacks that bedeviling Newton method by extending the classical SR1 update method (Dauda *et al.*, 2016) for general problems to nonlinear equations without using exact gradient and Jacobian. The proposed method was capable of reducing the execution time (CPU time) and number of iterations. The modification was achieved by simply approximating the inverse Hessian matrix $B_{k+1}^{-1}$ to $\theta_k I$ without computing the Jacobian. Unlike the classical SR1 method, the modification neither required to store an $n\times n$ matrix at each iteration nor needed to compute the Jacobian matrix. The remaining part of the article was organized by presenting the derivation of the proposed method in section 2. In section 3, some numerical results are presented, while section 4 presents the conclusion and future work.

## The proposed method

The idea of the proposed quasi-Newton method in which $(B_{k+1})^{-1}$ updated from $(B_k)^{-1} = \theta_k I$ was as a result of modification to SR1 update in (Dauda *et al.*, 2016). Applying the idea of symmetric rank-one (SR1) update, the following search direction was obtained. Recall, in (Wright & S.J, 2006) from Sherman-Morrison formula, the inverse of SR1 update was denoted as $(B_{k+1})^{-1}$ and given by

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(s_k - B_k^{-1}y_k)(s_k - B_k^{-1}y_k)^T}{(s_k - B_k^{-1}y_k)^T y_k} \tag{5}$$

where $B_k^{-1}$ is the inverse of $B_k$, which is an approximation to the Jacobian updated at each iteration (Morales, 2008).

The matrix $B_{k+1}$ was chosen so that it satisfied the secant equation

$$B_{k+1}s_k = y_k,\ s_k = x_{k+1} - x_k \ \ and \ \ y_k = F(x_{k+1}) - F(x_k) \tag{6}$$

By approximating $B_k^{-1}$ with the matrix $\theta_k^{-1} I$ where $\theta_k = \frac{y_k^T y_k}{y_k^T s_k}$

(Morales, 2008; Zhou, 2013) and substitute in (5) it became:

$$B_{k+1}^{-1} = \theta_k I + \frac{(s_k - \theta_k I y_k)(s_k - \theta_k I y_k)^T}{(s_k - \theta_k I y_k)^T y_k} \tag{7}$$

$$B_{k+1}^{-1} = \theta_k + \frac{(s_k - \theta_k y_k)(s_k - \theta_k y_k)^T}{(s_k - \theta_k y_k)^T y_k} \tag{8}$$

$Q_{k+1}^{-1} = B_{k+1}^{-1}$ whenever $B_k^{-1} = \theta_k I$. The quasi Newton's direction $d_{k+1} = Q_{k+1}F(x_{k+1})$ in which the (nonsingular) matrix $Q_{k+1} \in R^{nxn}$ was an approximation satisfying the standard secant equation (Li & Shengjie, 2015). Thus,

$$Q_{k+1}F(x_{k+1}) = \theta_k F(x_{k+1}) + \frac{(s_k - \theta_k y_k)(s_k - \theta_k y_k)^T F(x_{k+1})}{(s_k - \theta_k y_k)^T y_k} \tag{9}$$

Hence,

$$d_{k+1} = \begin{cases} -F(x_0) & if\ k=0 \\ -\theta_k F(x_{k+1}) + \frac{(s_k - \theta_k y_k)(s_k - \theta_k y_k)^T F(x_{k+1})}{(s_k - \theta_k y_k)^T y_k}, & if\ k \geq 1 \end{cases} \tag{10}$$

Since $d_k$ given by (10) might not be a descent direction, the standard Wolfe and Armijo line searches could not be used to compute the step-size directly, the non-monotone line search proposed by Li and Fukushima ( Fukushima & Li, 2000; Fukushima & Li, 2000) was used to compute the next step-size $\alpha_k$. It was the most frequently used line search in practice. The inexact line search to be used was sufficiently decreased the function values along the ray $x_k + \alpha_k d_k > 0$, i.e. $\|F(x_k + \alpha_k d_k)\| \leq \|F(x_k)\|$. Motivated by this features, let $\sigma_1 > 0, \sigma_2 > 0, r \in (0,1)$ be constants and $\eta_k$ be a given positive sequence such that:

$$\sum_{k=0}^{\infty} \eta_k < \infty \tag{11}$$

*Let* $\alpha_k = max\ \{1, r_1, r_2, \ldots\}$ *satisfy*

$$f(x_k + \alpha_k d_k) - f(x_k) \leq -\sigma_1 \|\alpha_k F(x_k)\|^2 - \sigma_2 \|\alpha_k d_k\|^2 + \eta_k F(x_k) \tag{12}$$

the SR1 method was an iterative method that generated a sequence of $\{x_k\}_{k\geq 0}$ from a given initial guess $x_0$ via the following

$$x_{k+1} = x_k - \alpha_k B_k^{-1} F(x_k), k = 0,1,2, \ldots \tag{13}$$

where $\alpha_k > 0$ is a step length determined by (12).

## Algorithm (proposed method)

**Step 1**: Given $x_0, \alpha > 0$, $\sigma \epsilon (0,1)$ and $\epsilon > 0$ compute $d_0 = -F(x_0), set\ k = 0$.

**Step 2**: Compute $g_k \approx \frac{F(x_k+\alpha_k F(x_k))}{\alpha_k}$ and test the stopping criterion, i.e. $\|g(x_k)\| \leq \varepsilon$. If yes, then stop. Otherwise continue with step 3

**Step. 3** Compute $\alpha_k$ by using the line search (12)

**Step. 4** Compute $x_{k+1} = x_k + \alpha_k d_k$

**Step. 5** Compute search direction using (10)

**Step. 6** Consider $k = k + 1$ and go to step 2

## Numerical results

In this section, a comparison on the performance of the proposed method for solving nonlinear equations (1) with the following two methods was made, denoting the methods as Alg1, Alg2 and Alg3 respectively. The following benchmark problems were used

P1**:** (System of exponential nonlinear equations) (Wright & S.J, 2006).

$$F(i) = e^{x_i} - 1; i = 1,2,3, \cdots, n.$$
$$x_0 = (0.5, 0.5, \cdots, 0.5)^T$$

P2: (System of trignometric nonlinear equations) (Dauda *et al.*, 2016).

$$F(1) = x_1 - e^{cos(\frac{x_1+x_2}{n+1})};$$
$$F_i(x) = x_i - e^{cos(\frac{x_i+x_{i+1}}{n+1})};$$
$$F_i(n) = x_n - e^{cos(\frac{x_n+x_{n+1}}{n+1})};$$
$$i = 1,2,3, \cdots, n.$$
$$x_0 = (1,1,1, \ldots, 1)^T$$

P3: (System of exponential nonlinear equations)(Dauda *et al.*, 2016).

$$F(i) = i * (1 - x_i^2 - (e^{x_i^2}))$$
$$F(n) = \frac{n}{10} * (1 - e^{-x_i^2})$$
$$\mathrm{i} = 1,2,3, \ldots, \mathrm{n};\ x_0 = (0.5, 0.5, 0.5 \ldots, 0.5)^T$$

P4: (System of exponential nonlinear equations) (Dauda *et al.*, 2016).

$$F_i(x) = \begin{pmatrix} 2 & -1 & \cdots & & \\ 0 & 2 & -1 & & \\ \ddots & \ddots & \ddots & & \\ \ddots & \ddots & -1 & & \\ \cdots & -1 & 2 & & \end{pmatrix} x_i + (e_1^x - 1, e_2^x - 1, \cdots, e_n^x - 1)^T;$$

$$i = 1,2,3,\ldots,n;\ x_0 = (0.5, 0.5, 0.5, \cdots, 0.5)^T$$

P5: (System of nonlinear equations) (Waziri, Leong, & Mamat, 2012).

$$F(1) = \left(\frac{1}{3}\right) * x_1^3 + 0.5 * x_2^2$$

$$F(i) = -\left(0.5 * x_i^2\right) + i\left(\frac{1}{i}\right) * x_1^3 + (0.5 * x_{i+1}^2)$$

$$F(n) = -0.5 * x_n^2 + \left(\frac{1}{3}\right) * n * x_n^2$$

$$i = 1,2,3,\cdots,n;\ x_0 = (0.5, 0.5, 0.5, \cdots, 0.5)^T$$

P6: (System of nonlinear equations)(Waziri *et al.*, 2012).

$$F(i) = x_i^2 - 4$$

$$i = 1,2,3,\cdots,n;\ x_0 = (0.5, 0.5, 0.5, \cdots, 0.5)^T$$

P7: (System of trigonometric nonlinear equations) (Mamat *et al.*, 2016).

$$F(1) = 3 * x_1^3 + 2 * x_2 - 5 + (\sin(x_1 - x_2)) * (\sin(x_1 + x_2))$$

$$F(i) = -x_{i-1} * e^{x_{i-1} - x_i} + x_i * \left(4 + 3 * x_i^2\right) + 2 * x_{i+1} + (\sin(x_i - x_{i+1})) * (\sin(x_i + x_{i+1})) - 8$$

$$F(n) = -x_{n-1} * e^{x_{n-1} - x_n} + 4 * x_n - 3$$

$$i = 1,2,3,\cdots,n;\ x_0 = (0.5, 0.5, 0.5, \cdots, 0.5)^T$$

P8: (System of nonlinear equations)(Mamat *et al.*, 2016).

$$F(1) = x_1 * (x_1^2 + x_2^2) - 1$$

$$F(i) = x_i * \left(x_{i-1}^2 + 2 * x_i^2 + x_{i+1}^2\right) - 1$$

$$F(n) = x_n * (x_{n-1}^2 + x_n^2)$$

$$i = 1,2,3,\cdots,n;$$

$$x_0$$

The above test problems with different given initial points were considered, each problem has been tested using all the methods with different values of n=10,100,500,1000 and 10000, where $n$ is the number of variables of each problem. The search was stopped if the total number of iteration was exceeded 1000 or $\| F(x_k) \| < \epsilon$ with $\epsilon < 10^{-4}$. The experiment was carried out in the MATLAB 7.1, R2009b programming environment and run on a personal computer 1.8GHz, CPU processor and 4GB RAM memory and windows XP operator. The Algorithms were implemented with the following parameters $\sigma = \rho = 0.9$ for all k. ”P” indicates the problem; ”Iter” and ”Time” stand for the total number of iterations and the CPU time in seconds respectively. ” $\| F(x_k) \|$ ” is the norm of the residual at the stopping point. The symbol ” $-$ ” in the tables indicates a failure due to memory shortages or/and when the number of iterations exceeded 1000. Clearly the method alg1 was the best with complete success in comparison with Alg2 method and Alg3. Moreover, from Tables 1-2 it was evident that the Alg1 was the best (in terms of iteration). According to the Tables 1-2, the performance of these three methods were shown in Figures 1 and 2 by using the performance profiles of Dolan and Moré (Mort'e & J, 2002). Figure 1 shows the performance relative to the number of iteration. Similarly, Figure 2 shows the performance of the methods relative to CPU Time. For each method, the fraction $P(\tau)$ was plotted against τ. The top curve was the method that solved the most problems in a time that was within a factor $\tau$ of the best time. The figures indicates that Alg1 was the most efficient for solving the given test problems among the three methods since the top curve was corresponded to Alg1

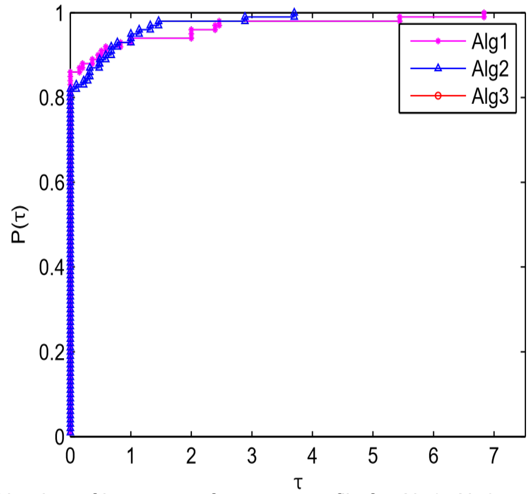


**Fig. 1** Number of iterates performance profile for Alg1, Alg2 and Alg3 of Problem 1-8.

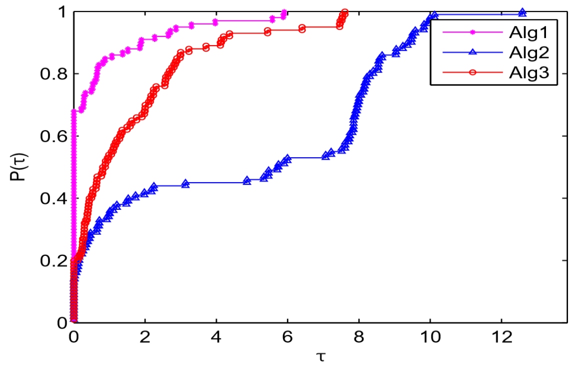


**Fig. 2** CPU time performance profile for Alg1, Alg2 and Alg3 of problem 1-8.

## CONCLUSION AND FUTURE WORK

Method for solving systems of nonlinear equations via memoryless SR1 update was presented. In finding the solution to nonlinear problems of the form $F(x) = 0, x \in R^n$, 40 benchmark test problems were solved. A comparison was made between the proposed methods Alg1 and two other methods Alg2 (Dauda *et al.*, 2016) and Alg3 (Mamat *et al.*, 2016). The contribution yielded a method that suitable for solving symmetric systems of nonlinear equations. Based on number of iterations, 37 problems were solved effectively by the proposed method. In terms of CPU time, the proposed method also outperformed the existing methods. The derivative-free feature of the proposed method gave ~~it~~ advantage to solve relatively large-scale problems (10,000 variables) compared to the existing method. From the preliminary numerical results, the proposed method turned out to be significantly faster, effective and suitable for solving large scale symmetric nonlinear equations. To extend this work futher in future, one could establish the global Convergence of the proposed Algorithm.

## ACKNOWLEDGEMENT

The authors would like to thank the administration of Universiti Sultan Zainal Abidin (UniSZA) for funding this research partially under the fundamental research grant FRGS/1/2015/ICT03/UniSZA/02/1.

**Table 1** Numerical results for Alg1, Alg2 and Alg3 of Problem 1-4.

| P | N | Alg1 Iter | Alg1 CPU | Alg1 NFE | Alg2 Iter | Alg2 CPU | Alg2 NFE | Alg3 Iter | Alg3 CPU | Alg3 NFE |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 4 | 0.002417 | 8.94E-04 | 8 | 0.000659 | 1.40E-04 | 8 | 0.000952 | 3.35E-04 |
| | 100 | 4 | 0.000893 | 4.53E-04 | 10 | 0.001177 | 4.43E-04 | 9 | 0.00223 | 5.24E-06 |
| | 500 | 4 | 0.001985 | 4.05E-04 | 11 | 0.000946 | 9.90E-04 | 9 | 0.001621 | 1.17E-05 |
| | 1000 | 5 | 0.002713 | 5.73E-04 | 11 | 0.001799 | 3.06E-06 | 9 | 0.00238 | 1.66E-05 |
| | 10000 | 7 | 0.161322 | 7.25E-04 | 12 | 0.051037 | 3.74E-05 | 9 | 0.02207 | 5.24E-05 |
| 2 | 10 | 4 | 0.000851 | 4.34E-04 | 4 | 0.000895 | 4.17E-04 | 8 | 0.00092 | 4.17E-04 |
| | 100 | 5 | 0.001075 | 5.49E-04 | 5 | 0.000783 | 8.30E-06 | 9 | 0.000766 | 8.30E-06 |
| | 500 | 5 | 0.001222 | 4.91E-04 | 5 | 0.001009 | 1.86E-05 | 10 | 0.001165 | 1.86E-05 |
| | 1000 | 5 | 0.001851 | 6.95E-04 | 5 | 0.001372 | 2.62E-05 | 10 | 0.00128 | 2.62E-05 |
| | 10000 | 5 | 0.017249 | 8.79E-04 | 7 | 0.043058 | 2.36E-06 | 11 | 0.010917 | 8.30E-05 |
| 3 | 10 | 6 | 0.001129 | 4.40E-04 | 7 | 0.000929 | 8.53E-04 | 10 | 0.001171 | 8.39E-04 |
| | 100 | 2 | 0.001127 | 9.21E-04 | 2 | 0.000872 | 8.75E-04 | 11 | 0.00064 | 8.75E-04 |
| | 500 | 2 | 0.002602 | 8.32E-04 | 2 | 0.000909 | 3.29E-06 | 12 | 0.001114 | 3.29E-06 |
| | 1000 | 2 | 0.004596 | 4.71E-04 | 2 | 0.001231 | 2.93E-07 | 13 | 0.001466 | 2.93E-07 |
| | 10000 | 1 | 0.033528 | 5.95E-04 | 1 | 0.005354 | 1.14E-04 | 14 | 0.007131 | 1.14E-04 |
| 4 | 10 | 7 | 0.000868 | 7.89E-04 | 7 | 0.000966 | 5.18E-04 | 9 | 0.001806 | 6.68E-04 |
| | 100 | 2 | 0.001198 | 7.13E-04 | 2 | 0.000709 | 7.12E-04 | 11 | 0.000652 | 7.12E-04 |
| | 500 | 2 | 0.008648 | 6.44E-04 | 2 | 0.000877 | 2.68E-06 | 12 | 0.00103 | 2.68E-06 |
| | 1000 | 2 | 0.003439 | 9.12E-04 | 2 | 0.001332 | 2.39E-07 | 12 | 0.002071 | 2.39E-07 |
| | 10000 | 1 | 0.040345 | 4.61E-04 | 1 | 0.007386 | 1.20E-04 | 14 | 0.006156 | 1.20E-04 |

**Table 2** Numerical results for Alg1, Alg2 and Alg3 of Problem 5-8.

| P | N | Alg1 Iter | Alg1 CPU | Alg1 NFE | Alg2 Iter | Alg2 CPU | Alg2 NFE | Alg3 Iter | Alg3 CPU | Alg3 NFE |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 10 | 7 | 0.017405 | 8.44E-04 | _ | 4.382775 | 0.30950 | 34 | 0.082004 | 9.93E-04 |
| | 100 | 8 | 0.023140 | 6.61E-04 | _ | 6.178020 | 0.74640 | 48 | 0.156076 | 9.99E-04 |
| | 500 | 9 | 0.066623 | 6.05E-04 | _ | 19.863146 | 0.63060 | 51 | 0.467148 | 9.91E-04 |
| | 1000 | 9 | 0.173882 | 8.78E-04 | _ | 58.794125 | 0.75350 | 53 | 1.394067 | 9.17E-04 |
| | 10000 | 10 | 12.292827 | 7.18E-04 | _ | 3886.7239 | 0.66700 | 51 | 88.942749 | 9.61E-04 |
| 6 | 10 | 11 | 0.025781 | 7.58E-04 | _ | 4.710905 | 0.4872 | 44 | 0.127728 | 9.92E-04 |
| | 100 | 11 | 0.030667 | 5.07E-04 | _ | 22.350121 | 8.5348 | 49 | 0.183563 | 9.28E-04 |
| | 500 | 11 | 0.0742 | 7.83E-04 | _ | 67.351265 | 18.806 | 53 | 0.696541 | 8.12E-04 |
| | 1000 | 12 | 0.221913 | 4.89E-04 | _ | 200.28285 | 26.5462 | 56 | 1.426673 | 9.18E-04 |
| | 10000 | 12 | 14.204232 | 5.95E-04 | _ | _ | _ | 62 | 112.31609 | 8.35E-04 |
| 7 | 10 | 10 | 0.001306 | 7.26E-04 | 130 | 0.037914 | 9.27E-04 | _ | 0.254265 | 4.2519 |
| | 100 | 11 | 0.001644 | 7.62E-04 | _ | 1.17309 | 0.9014 | _ | 0.301353 | 14.1021 |
| | 500 | 13 | 0.002679 | 1.33E-04 | _ | 0.603953 | 0.8794 | _ | 0.470482 | 31.6286 |
| | 1000 | 13 | 0.003755 | 1.89E-04 | _ | 0.927353 | 0.6155 | _ | 0.700859 | 44.7969 |
| | 10000 | 13 | 0.030127 | 5.98E-04 | 96 | 1.921122 | 4.04E-04 | _ | 5.343979 | 141.7241 |
| 8 | 10 | 4 | 0.001818 | 8.61E-04 | 5 | 0.001225 | 7.59E-05 | 8 | 0.001117 | 4.32E-05 |
| | 100 | 4 | 0.001827 | 4.57E-04 | 5 | 0.00147 | 2.52E-04 | 10 | 0.001171 | 1.43E-04 |
| | 500 | 4 | 0.002597 | 4.10E-04 | _ | 1.86652 | 18.1517 | 11 | 0.001673 | 3.21E-04 |
| | 1000 | 4 | 0.003452 | 5.80E-04 | _ | 2.824956 | 25.7091 | 11 | 0.002839 | 4.55E-04 |
| | 10000 | 5 | 0.031191 | 7.34E-04 | _ | 21.935074 | 81.3359 | 12 | 0.021555 | 8.36E-07 |

## REFERENCES


Dauda, M. K., Mamat, M., Waziri, M. Y., Ahmad, F., Mohamad, F. S. 2016. Inexact cg-method via sr1 update for solving systems of nonlinear equations. *Far East Journal of Mathematical Sciences,* 100, 11.

Fukushima, M., Li, D.-H. 2000. A derivative-free line search and global convergence of broyden like methods for nonlinear equations. *Optimization Methods and Software,* 13, 181-201.

Fukushima, M., Li, D.-H. 2000. A globally and superlinearly convergent gauss newton-based bfgs method for symmetric nonlinear equations. *SIAM Journal on Numerical Analysis,* 37, 152-172.

Li, J. L., Shengjie. 2015. Spectral dy type projection method for nonlinear monotone systems of equations. *Journal of Computation of Mathematics,* 4, 341-354.

Mamat, M., Dauda, M., Waziri, M., Ahmad, F., Mohamad, F. S. 2016. Improved quasi-newton method via psb update for solving systems of nonlinear equations. AIP Conference Proceedings 1782, 030009.

Morales, J. L. 2008. *Variational quasi-newton formulas for systems of nonlinear equations and optimization problems*. www.researchgate.net.

Morťe, J. J., Dolan, E. D. 2002. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A,* 91, 201-213.

Waziri, M. Y., Leong, W. J., Mamat, M. 2012. A two-step matrix-free secant method for solving large-scale systems of nonlinear equations. *Journal of Applied Mathematics, 2012,* 348654.

Wright, S. J., Nocedal, J. 2006. *Numerical Optimization* (second ed.). New York: Springer.

Zhang, G. Y., Maojun. 2015. A three-terms polak-ribière-polyak conjugate gradient algorithm for large-scale nonlinear equations. *Journal of Computational and Applied Mathematics, 286*, 186-195.

Zhou, W. 2013. A short note on the global convergence of the unmodified PRP method. *Optimization Letter, 7*, 1367-1372.