FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 4 No. 4, December, 2020, pp 382 – 390
DOI: https://doi.org/10.33003/fjs-2020-0404-242

MODIFICATION OF PSB QUASI-NEWTON UPDATE AND ITS GLOBAL CONVERGENCE FOR SOLVING SYSTEMS OF NONLINEAR EQUATIONS

M. K. Dauda

Department of Mathematical Sciences, Kaduna State University, Nigeria.
mkdfika@kasu.edu.ng

Abstract

Nonlinear problems mostly emanate from the work of engineers, physicists, mathematicians and many other scientists. A variety of iterative methods have been developed for solving large scale nonlinear systems of equations. A prominent method for solving such equations is the classical Newton's method, but it has many shortcomings that include computing Jacobian inverse that sometimes fails. To overcome such drawbacks, an approximation with derivative free line is used on an existing method. The method uses PSB (Powell-Symmetric Broyden) update. The efficiency of the proposed method has been improved in terms of number of iteration and CPU time, hence the aim of this research. The preliminary numerical results show that the proposed method is practically efficient when applied on some benchmark problems.

Mathematics Subject Classification: 65H11,65K05,65H12,65H18
Keywords: Conjugate Gradient, Quasi-Newton, *PSB*, Nonlinear Equations

INTRODUCTION

Consider the system of nonlinear equations

$$f_1(x_1,x_2,x_3,\dots,x_n)=0 \; f_2(x_1,x_2,x_3,\dots,x_n)=0 \; f_3(x_1,x_2,x_3,\dots,x_n)=0 \; f_4(x_1,x_2,x_3,\dots,x_n)=0 \; \vdots \; f_n(x_1,x_2,x_3,\dots,x_n)=0$$

The above system can be denoted by

$$F(x)=0, x\in R^n$$
(1).

where the function $F(x_1,x_2,x_3,\dots,x_n): R^n \to R^n$ is a nonlinear mapping assumed to satisfy the following conditions, (i) There exists an $x^* \in R^n$ such that $F(x^*)=0$ (ii) F is a continuously differentiable mapping in a neighborhood of x^* of the system and (iii) The Jacobian matrix of F at x given by $J(x)=F'(x)$ is symmetric. There are many iterative methods for solving (1) which include Newton's method, Quasi Newton's method, Diagonal Broyden-like method etc. but the most prominent method for finding the solution of (1) is the classical Newton's method which generates a sequence of iterates x_k from a given initial point x_0 via

$$x_{k+1}=x_k-(F'(x_k))^{-1}F(x_k), k = 0,1,2,\dots$$
(2)

Where $F'(x_k)$ is the Jacobian matrix of *F* at x_k.

The Newton method has some shortcomings which includes computation of the Jacobian matrix which may be challenging to compute and solving the Newton system in every iteration. Also, the common setback with some quasi-Newton methods is the need to compute and store an $n\times n$ matrix at each iteration; this is computationally costly for large scale problems. However, they are not particularly suitable for solving large scale nonlinear systems of equations. To overcome such deficiencies, a published article [8] have been reviewed and improve it by establishing its global convergence using suitable conditions. In the proposed method, the approximate Jacobian inverse H_k of PSB (Powell-Symmetric-Broyden) is updated and its efficiency has been improved in terms of number of iterations and CPU time, thereby making the method suitable for solving systems of nonlinear equations. Hence the main aim of this paper.

When the Jacobian matrix $F'(x_k)$, 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^*[7]. Throughout this article, we always assume that problem (1) is symmetric and can be converted to an equivalent global optimization problem.

$$min\, f(x), x \in R^n$$

(3)

with function *f* defined by

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

(4)

To approximate the gradient $\nabla f(x_k)$, which avoids computing exact gradient, Li and Fukushima [2], used the term

$$g(x_k) \approx \frac{F(x_k + \alpha_k F(x_k)) - F(x_k)}{\alpha_k},$$

(5)

It is clear that, when $F(x_k)$ is small, then $g(x_k) \approx \nabla f(x_k)$.

In general, CG methods for solving nonlinear systems of equations generate an iterative points x_k from initial given point x_0 using

$$x_{k+1} = x_k + \alpha_k d_k$$

(6)

Where $\alpha_k > 0$ is attained via line search and direction d_k are obtained using

$$d_k = \{F(x_k) \qquad\qquad if\ k = 1\ F(x_{k-1} + \beta_k d_{k-1}) \quad if\ k \geq 1$$

(7)

Where β_k is termed as conjugate gradient parameter [5], [8], [9] and [12].

2 Materials and Methods

2.1 Derivation of The Proposed Method: (PSB) Update

The PSB (Powell-Symmetric-Broyden) update comes from the solution of the following problem [7], [8]:

$$min_B||B - B_k||_F\ , s.t. Bs_k = y_k, (B\ - B_k) = (B - B_k)^T$$

(8)

The solution of (1) gives the Hessian update of PSB

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T + s_k(y_k - B_k s_k)^T}{s_k^T s_k} - \frac{{s_k}^T(y_k - B_k s_k)s_k s_k^T}{(s_k^T s_k)^2}$$

(9)

In general, the PSB method is an iterative method that generates a sequence of $\{x_n\}_k \geq 0$ from a given initial guess x_0 via the following

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

(10)

where $\alpha_k > 0$ is a step length determined by any suitable line search.

Recall using Sherman-Morrison-Woodbury, the formula of the inverse hessian approximation H_k for PSB is given by

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)y_k^T + y_k(s_k - H_k y_k)^T}{y_k^T y_k} - \frac{y_k^T(s_k - H_k y_k)y_k y_k^T}{(y_k^T y_k)^2}$$

(11)

H_k is updated at each iteration for $k = 0,1,2,\cdots$. The updated matrix H_{k+1} is chosen in such a way that it satisfies the secant equation (12).

$$s_k = H_{k+1} y_k, \text{ with } s_k = x_{k+1} - x_k \text{ and } y_k = F(x_{k+1}) - F(x_k) \qquad (12)$$

In this section, we approximate the inverse hessian H_k with identity matrix (i.e.$H_k \approx I$,) and substitute in (11) to get

$$H_{k+1} = I + \frac{(s_k - Iy_k)y_k^T + y_k(s_k - Iy_k)^T}{y_k^T y_k} - \frac{y_k^T(s_k - Iy_k)y_k y_k^T}{(y_k^T y_k)^2}$$

(13)

This is equivalent to

$$H_{k+1} = I + \frac{(s_k - y_k)y_k^T + y_k(s_k - y_k)^T}{y_k^T y_k} - \frac{y_k^T(s_k - y_k)y_k y_k^T}{(y_k^T y_k)^2}$$

(14)

Pre-multiplying (14) by $F(x_{k+1})$ ensure good approximation and yields

$$H_{k+1}F(x_{k+1}) = F(x_{k+1}) + \frac{(s_k - y_k)y_k^T + y_k(s_k - y_k)^T F(x_{k+1})}{y_k^T y_k} - \frac{y_k^T(s_k - y_k)y_k y_k^T F(x_{k+1})}{(y_k^T y_k)^2}$$

(15)

Where $F(x_{k+1})$is the update. By quasi-Newton direction $H_{k+1}F(x_{k+1})$in which the (nonsingular) matrix $H_{k+1} \in R^{n \times n}$ is the approximation satisfying the standard secant equation (12), then

$$d_{k+1} = -H_{k+1} g_{k+1}$$

(16)

Hence from (15) and (16) we have

$$d_{k+1} = -F(x_{k+1}) - \frac{(s_k - y_k)y_k^T + y_k(s_k - y_k)^T}{y_k^T y_k} F(x_{k+1}) - \frac{y_k^T(s_k - y_k)y_k y_k^T}{(y_k^T y_k)^2} F(x_{k+1})$$

(17)

Now the new direction is obtained via the following

$$d_k = \{-F(x_k) \qquad if\ k = 1 \ -F(x_{k-1}) + \theta_k F(x_{k-1}) + \delta_k F(x_{k-1}) \quad if\ k \geq 1$$

(18)

Where $\theta_k = \frac{(s_k - y_k)y_k^T + y_k(s_k - y_k)^T}{y_k^T y_k}$ and $\delta_k = \frac{y_k^T(s_k - y_k)y_k y_k^T}{(y_k^T y_k)^2}$ with iterative update as in (6), where α_k is step size and is obtained using non-monotone line search proposed by Li and Fukushima in [1]. Let $\sigma_1 > 0, \sigma_2 > 0, r \in (0,1)$ be constants and η_k be a given positive sequence such that

$$\sum \eta_k < \infty$$

(19)

let $\alpha_k = Max\{1, r^k\}$ that satisfy

$$f(x_k + \alpha_k d_k) - f(x_k) \leq -\sigma_1 ||\alpha_k F_k||^2 - \sigma_2 ||\alpha_k d_k||^2 + \eta_k F(x_k)$$

(20)

Finally, the following is the iterative scheme and the algorithm for the proposed method:

Algorithm 1

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 $F(x_k)$ and test the stopping criterion, i.e. $||F(x_k)|| \leq \epsilon$, If yes, then stop, otherwise continue with step 3.

Step 3: Compute α_k by using the line search
$f(x_k + \alpha_k d_k) - f(x_k) \leq -\sigma_1 ||\alpha_k F_k||^2 - \sigma_2 ||\alpha_k d_k||^2 + \eta_k F(x_k)$

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

Step 5: Compute search direction using (18)

Step 6: Set *k* = *k* + 1 and go to step 2

Global Convergence of the proposed Algorithm

This section presents the global convergence result of the proposed algorithm using the line search (20) above.
Assumption 2.2.1[15], [17]
In order to get global convergence of algorithm 1, we need the following assumptions.

(i) The level set $S = \{x \in R^n : \| F(x) \| \leq \sqrt{||f(x_0)||^2 + \eta}\}$ is bounded, where x_0 is initial point.

(ii) In a neighborhood N of S, the nonlinear mapping F is continuously differentiable and its gradients is Lipschitz continuous, i.e., there exists a constant $L > 0$ such that

$$\|F(x) - F(y)\| \leq L(\| x - y \|), \forall x, y \in N,$$

This shows that the sequence $\| F(x_k) \|$ is bounded, that is there exists a positive constant μ such that

$$\| F(x_k) \| \leq \mu, \forall x \epsilon S.$$

(21)

the following Lemma is needed in order to obtain the global convergence analysis of the proposed method.

Lemma 2.1

Suppose $\{x_k\}$ be generated by algorithm 1 and that assumption 2.2.1 holds. Then $\{x_k\} \subset S$.

Proof from (20), we have for all k,

$||F(x_k)||^2 \leq ||F(x_k)||^2 + \xi_k \leq \cdots \leq ||F(x_1)||^2 + \sum_{i=1}^{k} \xi_k$

Thus, we have

$$\leq ||F(x_1)||^2 + \eta$$

(22)

using (20), it is obvious that $\{x_k\} \subset S$.

Lemma 2.2

Supposed that assumption 2.2.1 holds and $\{x_k\}$ is generated by algorithm 1. Then we have

$$||\alpha_k d_k|| = 0 \quad (23)$$

$$||\alpha_k F(x_k)|| = 0$$

(24)

Proof

By (20), we have for all $k > 0$,

$\sigma_2 ||\alpha_k d_k||^2 \leq \sigma_1 \left||\alpha_k F(x_k)|\right|^2 + \sigma_2 \left||\alpha_k d_k|\right|^2$

$\leq \|f(x_k)\|^2 - \|F(x_{k+1})\|^2 + \eta_k.$

by summing the following inequality from $i = 1 \ to \ k,$ we obtained

$$\sigma_2 \sum_{i=1}^{k} ||\alpha_k d_k||^2 < \left||F(x_k)|\right|^2 - \left||F(x_{k+1})|\right|^2 + \sum_{i=1}^{k} \eta_i$$

$\leq ||F(x_1) + \eta_i.$

thus from (21) and the fact that η_k satisfy (22), the series $\sum_{i=1}^{k} \|\alpha_i d_i\|^2$ is convergent. Hence implies (23). Following the same fashion, (24) holds.

Theorem 2

Supposed that the assumption 2.2.1 holds and $\{x_k\}$ is generated by algorithm 1. If there exists a constant $\epsilon > 0$ such that

$$||F(x_k)|| \geq \epsilon$$

(25)

Then there exists a constant $\mu_2 > 0$ such that for all k

$$||d_k|| \leq \mu_2$$

and that $inf\left||F(x_k)|\right| = 0$ (27)

Proof

By contradiction, suppose that condition does not hold. Then there exists a constant $\epsilon > 0$, such that for all k, (25) holds. Moreover, from lemma 3.2 we have that (23) holds. Therefore, by equation (24) and the boundedness of $\{||d_k||\}$, we have

$$\alpha_k\left||d_k|\right| = 0$$

So, by combining the equations (21) and (25) and taking their limit, we obtain $\left||d_k|\right| = 0$ which contradict (26), thus (27) holds. This completes proof.

RESULTS AND DISCUSSION

In this section, we present the numerical results of the implementation of the proposed algorithm (denoted as M1). The performance of the *M*1 method is compared with that of *M*2[10] by solving several benchmark problems with their respective initial points using five (5) different dimensions ranging from 10 to 5000. In addition to numerical solution in [8], additional numerical solution is presented to ascertain the effectiveness of the proposed method.

P1:

$F(x) = (2\ \ -1 \quad -1\ 2\ -1 \quad \ddots\ \ddots \quad \ddots \quad \ddots\ -1 \quad -$
$1\ 2\ )x + (e_1^x - 1,\ e_2^x - 1, \cdots, e_n^x - 1,)^T$

(28)

P2: $F(x) = (e_i^x + 1)^2, i = 1,2,3, \cdots, n.$

(29)

P3:

$F(x) = (2\ 1 \quad 0\ 2\ \ -1 \quad \ddots\ \ddots \quad \ddots \quad \ddots\ -1 \quad -$
$1\ 2\ )x + (sinx_1 - 1,\ sinx_2 - 1, \cdots, sinx_n - 1)^T$

(30)

P4:

$F(1) = x_1 * (x_1^2 + x_2^2) - 1\ F(i)\ = x_i * ((x_{i-1})^2 + 2x_i^2 + x_{i+1}^2) - 1\ F(n)\ = x_n * (x_{n-1}^2 + x_n^2)$

(31)

P5:

$F(1) = x_1 - e^{coscos\left(\frac{x_1+x_2}{n+1}\right)}\ F(i)\ = x_i - e^{coscos\left(\frac{x_{i-1}+x_{i+1}}{n+1}\right)}\ F(n)\ = x_n - e^{coscos\left(\frac{x_n+x_n}{n+1}\right)}$

(32)

The comparison of the performance between the methods using the benchmark problems above was based on the performance profile presented by Dolan and More [3]. The performance profile $P: R \rightarrow [0,1]$ is defined as follows: Let P and S be the set of problems and set of solvers respectively. For n_s solvers and n_p problems, and for each

problem $p \in P$ and for each solver $s \in S,$ we define $t_{p,s}:=$(number of iterations required to solve problem p by solver s. The performance ratio is given by
$r_{p,s} := t_{p,s}/min\{t_{p,s}\}$.
Then the performance profile is defined by

$$P(\tau) = \frac{1}{n_p} size\{p \in P: r_{p,s} \leq \tau\}$$

for all $\tau \in R$ where $P(\tau)$ is the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ is within a factor $\tau \in R$ of the best possible ratio. The computational experiment is based on number of iterations and CPU time. The code for the proposed method was done using MATLAB 7.1, R2009b programming environment and run on a personal computer 2.4GHz, Intel (R) Core (TM) i7-5500U CPU processor, 4GB RAM memory and on windows XP operator. Both methods were implemented with the same parameters as $\alpha_1 = 0.01, r = 0.2, \sigma_1 = \sigma_2 = 10^{-4}$, and $\eta_k = \frac{1}{\{k+1\}^2}$. The search is stopped if: (i) $\| F(x_k) \| < \epsilon$ with $\epsilon < 10^{-4}$. (ii) The total number of iterations exceed 1000. The numerical results of the comparison between the proposed method $M1$ and the result in [10] are presented in Tables 1 and 2. The meaning of each column in the tables are stated as follows, $"P"$: stands for Benchmark problem, $"ISP"$: stands for initial starting points, $"n"$: stands for dimension of the test problems, $"Iter"$: the total number of iterations and "CPU" : the CPU time in seconds. In particular problem (i), M1 performs better than the performance of M2 if the number of iterations (iter) or the CPU time in seconds (Time) of M1 is less than the number of iterations or the CPU time corresponding to the M2 method respectively.

Figure 1 and 2 present the graphical results of problems 1-5 relative to number of iterations and CPU time respectively. That is, for each method, we plot the fraction $P(\tau)$ of problems for which the method is within a factor τ of the best time. The top curve is the method that performs better in a time that was within a factor τ of the best time. From Figure 1, the proposed $M1$ method performs relatively better better in terms of number of iterations. Figure 2 gives the performance of $M1$ methods relative to CPU time which outperforms $M2$, this indicates that $M1$ method achieved the objectives of this article, thus yields the best result.

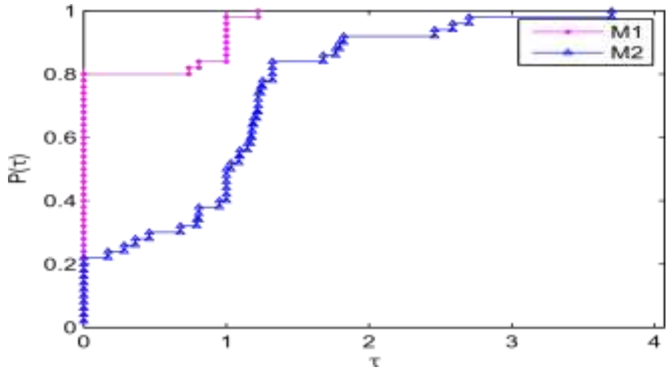

Figure 1: Performance profile of M1 and M2 methods with respect to the number of iterations for problems 1-5

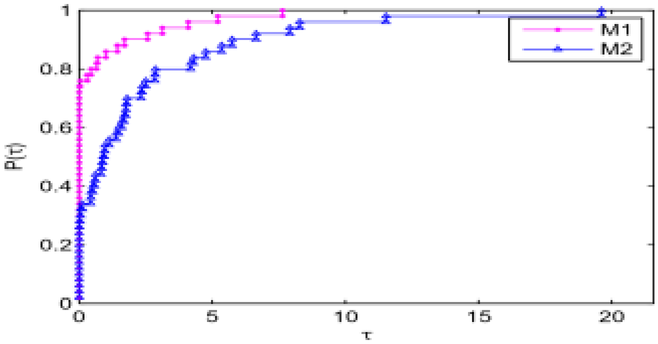

Figure 2: Performance profile of M1 and M2 methods with respect to the CPU Time for the problems 1-5.

Table 1: The Numerical Results for M1, and M2 on problems 1.

			M1		M2	
P	**ISP**	**N**	**Iter**	**CPU**	**Iter**	**CPU**
1	0.2	10	10	0.02005	16	0.03521
		100	11	0.02756	26	0.05994
		500	14	0.2472	31	0.33271
		1000	15	0.86169	29	0.88634
		5000	13	17.0559	31	17.7146
	0.5	10	19	0.00113	43	0.00564
		100	24	0.00602	33	0.0061
		500	22	0.1733	47	0.01983
		1000	26	0.84242	45	0.02279
		5000	21	15.1654	42	0.076617

Table 2: The Numerical Results for M1, and M2 on problems 2.

			M1		M2	
P	**ISP**	**N**	**Iter**	**CPU**	**Iter**	**CPU**
2	0.5	10	15	0.0009	34	0.0476
		100	15	0.0276	48	0.0926
		500	15	0.1984	51	0.3558
		1000	15	0.7385	53	1.3594
		5000	15	16.374	52	24.259
	0.8	10	19	0.0057	44	0.0018
		100	23	0.0033	28	0.0063
		500	23	0.0113	47	0.2229
		1000	24	0.0221	51	0.9047
		5000	21	0.0764	48	18.287

Table 3: The Numerical Results for M1, and M2 on problems 3.

			M1		M2	
P	**ISP**	**n**	**Iter**	**CPU**	**Iter**	**CPU**
3	0.8	10	9	0.0037	7	0.0006

		100	11	0.0005	2	0.0025
		500	12	0.0008	2	0.0842
		1000	13	0.0015	2	0.4692
		5000	13	0.0032	1	9.3477
	0.2	10	2	0.3185	4	0.1189
		100	2	0.3493	4	0.2241
		500	2	0.0536	5	0.1711
		1000	2	0.0578	5	0.1506
		5000	2	0.3901	5	0.1964

Table 4: The Numerical Results for M1, and M2 on problems 4.

			M1		**M2**	
P	**ISP**	**n**	**Iter**	**CPU**	**Iter**	**CPU**
4	0.5	10	5	0.1149	3	0.0067
		100	6	0.0347	3	0.1953
		500	6	0.176	3	0.11
		1000	6	0.1866	3	0.1809
		5000	7	0.3684	3	0.3004
	1	10	3	0.2526	7	0.3386
		100	3	0.1983	7	0.1463
		500	3	0.1635	7	0.3171
		1000	4	0.1548	7	0.4677
		5000	4	0.2757	7	0.7671

Table 5: The Numerical Results for M1, and M2 on problems 5.

			M1		**M2**	
P	**ISP**	**n**	**Iter**	**CPU**	**Iter**	**CPU**
5	0.6	10	7	0.1932	4	0.2734
		100	8	0.0142	4	0.1032
		500	8	0.2772	4	0.3002
		1000	8	0.2332	4	0.3529
		5000	8	0.3032	4	0.3386
	0.2	10	4	0.0094	8	0.1719
		100	4	0.0091	8	0.0305
		500	4	0.0017	9	0.0448
		1000	8	0.0186	9	0.1329
		5000	9	0.1535	9	0.5286

CONCLUSION

In this article, a method for solving nonlinear systems of equations (1) via modification of Powell-Symmetric-Broyden (PSB) update is presented. It is worth noting that $M1$ solves problems effectively, thus the proposed method is particularly suitable for symmetric equations. The global convergence of the given algorithm is established under suitable conditions. We have compared the $M1$ method with $M2$ by [10] and found that the proposed method is effective in practical computation and superior in many situations and the preliminary numerical results show that the proposed method is substantial and efficient for solving symmetric systems of non-linear equations (1).

ACKNOWLEDGEMENTS.

The author would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] D. H. Li and M. Fukushima, (2000). A derivative-free line search and global convergence of Broyden-like methods for nonlinear equations, Optimization Methods and Software, 13, 181-201.

[2] E. Dolan and J. More, (2002). Benchmarking optimization software with performance profiles, Math. Program. Ser. A, 91, 201-213.

[3]. Gonglin Yuana and Maojun Zhang, (2015). A three-terms Polak-Ribire-Polyak conjugate gradient algorithm for large-scale nonlinear equations, Journal of Computational and Applied Mathematics 286, 186-195.

[4] Jinkui Liu and Shengjie Li, (2015). Spectral DY Type Projection Method for Nonlinear Monotone Systems of Equations, Journal of Computation of Mathematics, 4, 341-354.

[5] J. L. Morales, (2008). Variational Quasi-Newton Formulas for Systems of Nonlinear Equations and Optimization problems.

[6] J. Nocedal and S.J. Wright, (2006). Numerical Optimization, 2nd ed., Springer, New York.

[7] Mustafa Mamat, M. K. Dauda, M. Y. Waziri, Fadhilah Ahmad, and Fatma Susilawati Mohamad (2016). Improved Quasi-Newton method via PSB update for solving systems of nonlinear equations, AIP Conference Proceedings 1782, 030009; doi:10.1063/1.4966066

[8]. M. K. Dauda, Mustafa Mamat, M. Y. Waziri, Fadhila Ahmad and Fatma Susilawati Mohamad, (2016). Inexact CG-Method via SR1 Update for Solving Systems of Nonlinear Equations, Far East Journal of Mathematical Sciences(FJMS) Volume 100, Issue 11, Pages 1787-1804.

[9] M. Y. Waziri and S. Jamilu, (2015). A Derivative-Free Conjugate Gradient Method and Its Global Convergence for Solving Symmetric Nonlinear Equations, International Journal of Mathematics and Mathematical Sciences, 39: Article ID 961487,10 pages.

[10] Mohd Rivaie, Mustafa Mamat and Abdelrhaman Abashar, (2015). A new class of nonlinear conjugate gradient coefficients with exact and inexact line searches, Applied Mathematics and Computation, 268, 1152-1163.

[11] Saman Babaie-Kafaki and Reza Ghanbaric, (2012). Two hybrid nonlinear conjugate gradient methods based on a modified secant equation, A Journal of Mathematical Programming and Operations Research, Vol. 63, 1027-1042.

[12]. Neculai Andrei, (2016). An adaptive conjugate gradient algorithm for large-scale unconstrained optimization, Journal of Computational and Applied Mathematics 292, 83-91.

[13]. W. Zhou and D. Shen, (2014). An inexact PRP conjugate gradient method for symmetric nonlinear equations, Numerical Functional Analysis and Optimization, vol. 35, no. 3, pp. 370-388.