Journal of the Nigerian Association of Mathematical Physics
Volume 57(**June – July** *2020 Issue*), **pp29 – 34**

OPTIMIZATION OF A DERIVATIVE-FREE METHOD FOR SOLVING LARGE SCALE SYSTEMS OF NONLINEAR EQUATIONS

M.K. Dauda

Department of Mathematical Sciences, Kaduna State University, Nigeria.

Abstract

Mostly, research from the work of engineers, mathematicians, physicist, computer scientist, Astronomers and other scientist are equations, which are nonlinear in nature. Researchers made significant effort in solving equations of the form $F(x) = 0$, $x \in R^n$, ***yet some of the methods developed do have deficiency. In this article, an improved version of a derivative free method for Solving Nonlinear Equations via a derivative free line search is presented. Interestingly, without computing any derivative, the proposed method never fail to converge throughout the numerical experiments and it satisfied the descent condition. Different initial starting points were used on a benchmark problems and the output is based on number of iterations and CPU time. The approach yield a method of solving systems of nonlinear equations that is capable of significantly reducing the CPU time and number of iteration, as compared to its counterparts. Thus, suitable and achieved the objective. The efficiency of the proposed scheme has been confirmed by the numerical results presented.***

Keywords**:** Optimization, Decent Direction, Derivative Free, Line Search, Nonlinear Equations

1. Introduction

Consider the global optimization problem of the form

$$minf(x), x \in R^n. \quad (1)$$

With condition that$(x_k + \alpha_k d_k) \leq f(x_k)$,where α_k is the line search and d_k is the search direction. The search direction d_k is generally required to satisfy the descent condition $\nabla(x_k)^T d_k < 0$.Supposed$f$ be a norm function defined by a function

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

where, $||\cdot||$ stands for the Euclidian norm. Then the unconstrained optimization problem, (1) is equivalent systems to nonlinear equations problem

$$F(x) = 0, \qquad x \in R^n \quad (3)$$

where $F: R^n \to R^n$is nonlinear mappingassumed to satisfy (i) There exists $x^* \in R^n$such that $F(x^*) = 0$, (ii) $F$is a continuously differentiable mapping in a neighborhood of x^* (iii) the Jacobian matrix of Fat $x$given by $J(x) = F'(x)$is symmetric. There are various methods for solving nonlinear equations (3) [1-6]. Among them is the Newton's method, which plays an important role due to its rapid convergence rate and decreasing the function value of the sequence. However, the methods still suffers deficiency such as computation of the derivative $F'$and solution of some system of linear equations at each iteration. The iterative formula of a Newton method is given by

$x_{k+1} = x_k + s_k, s_k = \alpha_k d_k, k = 0,1,\cdots,$

where$\alpha_k$is a step length to be computed by a line search technique, x_{k+1}represents a new iterative point,x_k is the previous iteration, while $d_k$is the search direction to becalculated by solving thelinear system of equations below,

$$F'(x_k)d_k = -F(x_k), \quad (4)$$

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

The drawback of the technique (4) is the need to compute the Jacobian matrix $F'(x_k)$ at every iteration, whichincreases the difficulty in computation; this is due to the first-order derivative of the system. Sometimes the derivatives are not available or could not be obtained exactly especially for the large-scale problems [7, 8].Hence these requires the need for derivative free methods for solving such problems. There are many scholars [9-13] that discussed derivative-free methods, yet some problems persist. Therefore, motivated by [14] the purpose of this article is to improve the derivative free method with decent direction for solving system of nonlinear equations via derivative free line search[16, 17] and the approximations $F_o(x_k) \approx \gamma_k I$.Where I is an identity matrix and γ_k is the new parameter introduced. The method [14] retained a norm descent property without computing the Jacobian matrix with less number of iterations and CPU time that is globally convergent.

Corresponding Author**:** Dauda M.K., Email: mkdfika@kasu.edu.ng, Tel: +2348038893151, +2348023582223

The next section discussed the proposed method. Section 3reports some numerical results of the proposed method while sections 4 gives the conclusion.

2. Derivation of the Method

This section discussed the Derivation of the proposed method for solving system of nonlinear equations (3). The aim is to improve the computational performance of the existing method [14] by the concept of a derivative free line search method, and introducing the approximations $F_o(x_k) \approx \gamma_k I$, where I is an identity matrix and γ_k is the new parameter.

Recall, from Taylor's expansion of the first order the approximation of $F(x_{k+1})$, if follows that

$$F(x_{k+1}) \approx F(x_k) + F^o(\delta)(x_{k+1} - x_k) \quad (5)$$

where the parameter δ fulfills the conditions $\delta \in [x_k, x_{k+1}]$,and that

$$\delta = x_k + \lambda(x_{k+1} - x_k) \qquad 0 \leq \lambda \leq 1. \quad (6)$$

Note that the distance between x_k, and x_{k+1} is small enough.By taking $\lambda = 1$ in (6), then $\delta = x_{k+1}$. Also, if $\lambda = 0$, then $\delta = x_k$.Therefore we have

$$F^o(\delta) \approx \gamma_{k+1} I. \quad (7)$$

Knowing this, the expression (5) becomes

$$F(x_{k+1}) - F(x_k) = \gamma_{k+1} I(x_{k+1} - x_k) = \gamma_{k+1}(x_{k+1} - x_k). \quad (8)$$

Now, (8) transformed to standard secant condition [3,4].

$$\gamma_{k+1} s_k = y_k, \quad (9)$$

where, $y_k = F(x_{k+1}) - F(x_k)$ and $s_k = x_{k+1} - x_k$,

In [1], Li and Fukushima used the term

$$g_k = \frac{F(x_k + \alpha_k F(x_k)) - F(x_k)}{\alpha_k} \quad (10)$$

With the aid of (10), the scheme will maintain derivative free process and avoids computing exact gradient. It is clear that when$\|F(x_k)\|$is small, then the approximate gradient $\nabla f(x_k)$, is given by

$$g_k \approx \nabla f(x_k).$$

Consequently, g_k^T is obtainable. Thus, pre-multiplying both side of (9) by g_k^T, the relation allows us to compute theparameter γ_{k+1}in the following way:

$$\gamma_{k+1} g_k^T s_k = g_k^T y_k \quad (11)$$

Taking the transpose of both side of (11) gives

$$[\gamma_{k+1} g_k^T s_k]^T = [g_k^T y_k]^T$$

$$\gamma_{k+1} [g_k^T s_k]^T = [g_k^T y_k]^T$$

Where γ_{k+1} is a parameter.

$$\gamma_{k+1} [s_k^T g_k] = [y_k^T g_k] \quad (12)$$

Multiply both side of (12)by$[s_k^T g_k]^{-1}$ yields

$$\gamma_{k+1} [s_k^T g_k]^{-1} [s_k^T g_k] = [s_k^T g_k]^{-1} [y_k^T g_k]$$

$$\gamma_{k+1} = [s_k^T g_k]^{-1} [y_k^T g_k] \quad (13)$$

Thus, direction is given by

$$d_{k+1} = -\gamma_{k+1} F(x_k), \text{for} k = 1,2,3, \cdots \quad (14)$$

Finally, the general scheme is given by

$$x_{k+1} = x_k + \alpha_k d_k. \quad (15)$$

wherex_{k+1} represents a new iterative point,x_k is the previous iteration, while $d_k$is the search direction to be obtain using (14). The basic requirement of the line search is to sufficiently decrease the function values i.e. $\|F(x_k + \alpha_k d_k)\| \leq \|F(x_k)\|$.Therefore, to compute the line search α_k in (15), the line search used in [16, 17]is the best choice, since it is derivative free in nature. This is given by

$$\alpha_k = max\{s, rs, r^2 s, \cdots\} \quad (16)$$

satisfying$-g(x_k + \alpha_k d_k)^T d_k \geq \omega \alpha_k \|g(x_k + \alpha_k d_k)\| \|d_k\|^2$

Remark

It is clear that the line search is well defined.

Algorithm 1.

Step 1: Given $\gamma_0 = 1, \epsilon = 10^{-4}, x_0, (r, r\omega) \in (0,1)$, set $k = 0$ and $d_0 = -g_0$.

Step 2: Compute $F(x_k)$ and check the stopping conditions. If yes, then stop; otherwise continue with Step 3.

Step 3: Compute search direction $d_{k+1} = \gamma_{k+1} F(x_k)$ using (14)

Step 4: Compute step the length α_k(using (16).

Step 5: Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 6: Compute $F(x_{k+1})$.

Step 7: determine the parameter ,γ_{k+1} from (13)

Step 8: Repeat$k = k + 1$, and go to Step 3.

3. Numerical Results

This section gives the computational performance of the proposed method named $M1$and two other methods named $M2$[14] and $M3$[15] respectively. In the presentation, the proposed method [15] which is the improved version of [14], is compared with classical version of [14] and the method which was compared with [14] initially. To ascertain the efficiency of the result, all the three methods were tested using the same Benchmark test problems as indicated in the Appendix. The accuracy of the methods was confirmed using different initial points. The computational codes for all the methods was written in Matlab 7.9.0 (R2009b) [18], and run on a personal computer 2.00 GHz CPU processor and 3 GB RAM memory.

Table 1: Numerical Comparison of $M1, M2$ and $M3$ methods for problem 1 and 2.

			$M1$			$M2$			$M3$		
S/N	**Dim**	**ISP**	**Iter**	**Time**	**Norm**	**Iter**	**Time**	**Norm**	**Iter**	**Time**	**Norm**
1	10	0.2	7	0.029974	5.78E-04	31	0.020822	8.12E-04	10	0.006045	6.84E-05
	100		8	0.019030	7.31E-04	36	0.007647	8.42E-04	11	0.002768	8.49E-05
	1000		9	0.004687	9.24E-04	41	0.020315	8.73E-04	13	0.008038	4.14E-05
	10000		11	0.114284	4.67E-04	46	0.146886	9.04E-04	14	0.061264	5.14E-05
	100000		12	0.307812	5.91E-04	51	1.732412	9.37E-04	15	0.634616	6.39E-05
	10	0.9	5	0.001026	7.42E-04	27	0.005472	8.78E-04	11	0.002501	3.99E-05
	100		6	0.001188	9.37E-04	32	0.007435	9.10E-04	12	0.002964	4.95E-05
	1000		8	0.003464	4.74E-04	37	0.018337	9.44E-04	13	0.006808	6.15E-05
	10000		9	0.023079	6.00E-04	42	0.139060	9.78E-04	14	0.057503	7.64E-05
	100000		10	0.228708	7.58E-04	48	1.593948	8.11E-04	15	0.603471	9.49E-05
2	10	0.2	5	0.006740	3.86E-04	31	0.025203	*	10	0.013997	4.70E-05
	100		6	0.002321	1.47E-04	31	0.026398	*	11	0.003108	5.21E-05
	1000		6	0.003298	4.64E-04	31	0.064822	*	12	0.007429	6.43E-05
	10000		7	0.025496	1.76E-04	36	1.871130	*	13	0.051324	8.75E-05
	100000		7	0.255355	5.56E-04	35	140.6098	*	15	0.594499	4.67E-05
	10	0.9	5	0.001098	4.58E-04	124	0.021183	9.94E-04	9	0.002826	4.95E-05
	100		6	0.001445	1.74E-04	139	0.027641	9.69E-04	10	0.002959	3.36E-05
	1000		6	0.003226	5.50E-04	154	0.057261	9.44E-04	11	0.006268	2.41E-05
	10000		7	0.027749	2.09E-04	168	0.394139	9.95E-04	11	0.040932	7.62E-05
	100000		7	0.257811	6.60E-04	183	5.132788	9.70E-04	12	0.468681	2.62E-05

Table 2: Numerical Comparison of $M1, M2$ and $M3$ methods for problem 3 and 4.

			$M1$			$M2$			$M3$		
S/N	**Dim**	**ISP**	**Iter**	**Time**	**Norm**	**Iter**	**Time**	**Norm**	**Iter**	**Time**	**Norm**
3	10	0.2	5	0.007880	3.51E-04	39	0.260591	*	9	0.007014	9.11E-05
	100		6	0.001555	2.22E-04	37	0.033213	*	10	0.002861	9.56E-05
	1000		6	0.002903	7.03E-04	37	0.086759	*	11	0.006741	7.62E-05
	10000		7	0.023343	4.44E-04	37	2.112099	*	12	0.056222	7.99E-05
	100000		8	0.273506	2.81E-04	37	142.4925	*	13	0.633558	6.37E-05
	10	0.9	4	0.000924	6.76E-04	25	0.004853	8.74E-04	5	0.001823	5.77E-05
	100		5	0.001744	4.27E-04	30	0.007191	9.06E-04	6	0.001918	8.15E-05
	1000		6	0.003145	2.70E-04	35	0.017076	9.39E-04	7	0.004520	2.47E-05
	10000		6	0.019402	8.55E-04	40	0.124208	9.72E-04	7	0.031964	7.82E-05
	100000		7	0.260632	5.41E-04	46	1.999564	8.06E-04	9	0.452263	1.06E-05
4	10	0.2	5	0.006751	3.51E-04	39	0.052702	*	9	0.005365	9.11E-05
	100		6	0.001168	2.22E-04	37	0.028444	*	10	0.003112	9.56E-05
	1000		6	0.002344	7.03E-04	37	0.071258	*	11	0.005913	7.62E-05
	10000		7	0.019958	4.44E-04	37	1.873526	*	12	0.040622	7.99E-05
	100000		8	0.210191	2.81E-04	37	140.1881	*	13	0.467590	6.37E-05
	10	0.9	4	0.000875	6.76E-04	25	0.004130	8.74E-04	5	0.582910	5.77E-05
	100		5	0.001102	4.27E-04	30	0.007141	9.06E-04	6	0.012311	8.15E-05
	1000		6	0.002571	2.70E-04	35	0.014490	9.39E-04	7	0.031729	2.47E-05
	10000		6	0.018040	8.55E-04	40	0.098346	9.72E-04	7	0.112399	7.82E-05
	100000		7	0.194157	5.41E-04	46	1.514717	8.06E-04	9	0.357280	1.06E-05

According to the program, the iteration is terminated if the total number of iterations exceeds 1000 or $\|F(x_k)\| \leq 10^{-4}$. If the method fails, the symbol $"*"$ is used and represents failure due to; (i) Memory requirement (ii) Number of iterations exceed 1000. (iii) If $\|F(x_k)\|$ is not a number.

For the both algorithms in [14,15], the following parameters $\omega_1 = \omega_2 = 10^{-4}, r = 0.2$ and $\eta_k = \frac{1}{(k+1)^2}$are used.

Table 3: Numerical Comparison of $M1, M2$ and $M3$ methods for problem 5 and 6.

			$M1$			$M2$			$M3$		
S/N	**Dim**	**ISP**	**Iter**	**Time**	**Norm**	**Iter**	**Time**	**Norm**	**Iter**	**Time**	**Norm**
5	10	0.2	2000	0.238044	0.1974	112	0.027648	9.94E-04	8	0.013481	7.55E-05
	100		2000	0.277768	0.6243	127	0.035844	9.69E-04	10	0.002987	4.72E-07
	1000		2000	0.729098	1.9741	142	0.064516	9.44E-04	10	0.010842	1.49E-06
	10000		2000	5.708984	6.2426	156	0.469077	9.95E-04	10	0.051874	4.72E-06
	100000		2000	68.898187	19.7408	171	6.504744	9.70E-04	10	0.464678	1.49E-05
	10	0.9	2000	0.226144	0.1216	30	0.006853	9.13E-04	7	0.003896	2.49E-05
	100		2000	0.289621	0.3845	35	0.006963	9.47E-04	7	0.002062	7.86E-05
	1000		2000	0.810016	1.216	40	0.021452	9.81E-04	8	0.006757	8.99E-05
	10000		2000	5.595398	3.8452	46	0.140034	8.13E-04	10	0.046637	6.12E-05
	100000		2000	65.461812	12.1595	51	1.947417	8.43E-04	11	0.527849	1.09E-06
6	10	0.2	8	0.045419	7.53E-04	28	0.016831	8.26E-04	10	0.059030	5.05E-05
	100		10	0.003527	4.06E-04	33	0.007480	8.29E-04	10	0.002714	6.48E-05
	1000		11	0.003839	5.17E-04	38	0.019901	8.56E-04	11	0.007156	5.92E-05
	10000		12	0.037682	6.55E-04	43	0.179768	8.87E-04	12	0.074682	7.03E-05
	100000		13	0.364561	8.28E-04	48	2.121760	9.19E-04	13	0.652322	8.70E-05
	10	0.9	14	0.001905	4.29E-04	*	0.348713	1.1412	12	0.003284	6.58E-05
	100		16	0.002461	6.91E-04	*	0.419275	4.4653	13	0.003611	4.66E-05
	1000		19	0.007444	4.78E-04	*	1.136226	16.4158	14	0.010835	5.15E-05
	10000		21	0.070103	6.88E-04	*	9.293672	52.3308	15	0.071868	6.31E-05
	100000		23	0.829604	8.55E-04	*	96.63993	154.1355	16	0.814324	7.83E-05

Tables 1 gives the numerical results of problems 1 and 2. Tables 2 gives the numerical results of problems 3 and 4 while Tables 3 gives the numerical results of problems 5 and 6. In all the Tables, the Columns for each methods is titled with "S/N", "Dim", "ISP","Iter", "Time" and "Norm" which stands for the serial number of the problem, dimension of the problem, Initial Starting Point, total number of all iterations, the CPU time in solving the problems measured in seconds and residual at the stopping point ($\|F(x_k)\|$)respectively.

From the Table, it is observed that all the three methodsi.e.$M1, M2$ and $M3$attempt to solve the systems of nonlinear equations (3), it is worth noting that the proposed method effectively solved the problems where the other two methods fails. For instance, in problems 2, 3 and 6, $M2$ fails where $M1$ successfully and effectively solved the problems with minimal number of iterations. The method $M1$ considerably outperforms the $M2$ and $M3$ in almost all the tested problems, since it has the least number of iterations and CPU time as compared to the CPU time and the number of iterations for the method$M2$ and $M3$ respectively.This is quite evident that the proposed method $M1$improved the computational performance of $M2$.Theefficiency and accuracy of the proposed algorithm was clear for it solves systems of nonlinear equations (3) where mostly the method $M2$ and $M3$ fails. This achievement is due to the derivative free line search used [16,17].

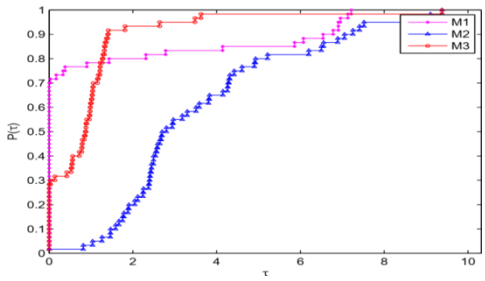

Figure 1: Performance profile of $M1, M2$ and $M3$ methods with respect to the number of iterations.

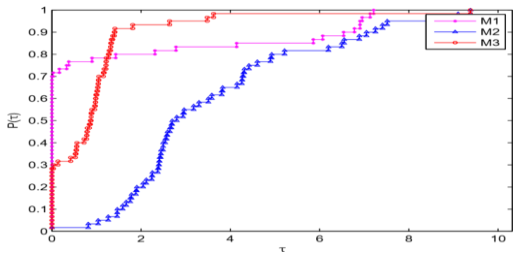

Figure 2: Performance profile of $M1, M2$ and $M3$ methods with respect to the CPU time.

Figures 1, 2, and 3 gives the graphical representation of the numerical results of all the three methods. The analysis was done using the profile by Dolan and More [19]. The proposed method operates with less number of iteration and CPU time denoted by $M1$representedin pink color, while $M2$ and $M3$ performs below $M1$ with line represented in red color and blue color respectively. Obviously, the performance of the proposed method is better in terms of number of iterations, CPU time and residual function than the other two existing methods in comparison. Another merit of the proposed method is its ability to solve problems at shortest possible time. Generally, the results shows that the proposed method is an improvement of [14] with respect to matrix storage and computational time.

4. Conclusion

In this paper, an improved version of derivative-free decent method via acceleration parameter for solving systems of nonlinear equations is given. The aim of this research is to improve the computational performance and convergent rate of existing methods, thereby reducing the computational time at each iteration. The aim of this research has greatly achieved by doing away with a computation of the Jacobian inverse during the iteration process. It is a fully derivative-free iterative method, which retain the global convergence of the classical method under some appropriate conditions. The Numerical results reported using a set of large-scale test problems has shown that the proposed method is practically effective. The method is valid in terms of derivation, reliable in terms of number of iterations and accurate in terms of CPU time. Conclusively, the proposed method is recommended for solving large-scale system of nonlinear equations of the form $F(x) = 0, x \in R^n$.

Appendix.

The following test problems are used for the numerical experiments. The mapping $F$is taking as $(x) = (f_1(x);\ f_2(x);\ \cdots ;f_n(x))^T$; and $x = (x_1; x_2;\ \cdots ;x_n)^T$ .The test problems 1,2,3,4,5 are from [20] while problem 6 is from [21].

Problem 1.

$f_1(x) = (x_1 - x_2^2)(x_1 - \sin(x_2))$,
$f_2(x) = (\cos(x_2) - x_1)(x_2 - \cos(x_1))$.

Problem 2.

$f_1(x) = -x_1 + 0.5x_2^2 - 1.5$,
$f_2(x) = -x_2 + 0.605\, exp(1 - x_1^2) + 0.395$

Problem 3.

$f_1(x) = x_1^2 - 2x_1 + \frac{1}{3}x_2^3 + \frac{2}{3}$
$f_2(x) = x_1^3 - x_1x_2 - 2x_1 + 0.5x_2^2 + 1.5$

Problem 4.

$f_1(x) = x_1$,
$f_2(x) = \frac{10x_1}{x_1 + 0.1} + 2x_2^2$.

Problem 5.

$f_1(x) = -13 + x_1 + \big((-x_2 + 5)x_2 - 2\big)x_2$,
$f_2(x) = -29 + x_1 + \big((x_2 + 1)x_2 - 14\big)x_2$.

Problem 6.

$F_1(x) = x_1(x_1^2 + x_2^2) - 1$,
$F_i(x) = x_i\big(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2\big) - 1$,
$F_n(x) = x_n(x_{n-1}^2 + x_n^2)$.
$i = 2,3,\cdots n - 1$.

References

[1] D. Li and M. Fukushima, (2000). A global and superlinear convergent Gauss-Newton based BFGS method for symmetric nonlinear equation, SIAM Journal on numerical Analysis, 37, 152-172.

[2] Mustafa Mamat, M. K. Dauda, M. Y. Waziri, Fadhilah Ahmad, and FatmaSusilawati 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.

[3] Mustafa Mamat, FatmaSusilawati Mohamad, Abubakar S. Magaji and M.Y. Waziri (2019). Derivative Free Conjugate Gradient Method via Broyden's Update for solving symmetric systems of nonlinear equations. Journal of Physics: Conference Series, 1366. 012099 IOP Publishing doi:10.1088/1742-6596/1366/1/012099

[4] M. K. Dauda, Mustafa Mamat, Mohamad A. Mohamed, NorShamsidah Amir Hamzah. (2019). Hybrid conjugate gradient parameter for solving symmetric systems of nonlinear equations. Indonesian Journal of Electrical Engineering and Computer Science Vol. 16, No. 1, October, pp. 539~543 ISSN: 2502-4752, DOI: 10.11591/ijeecs.v16.i1.pp539-543.

[5] Gongli Yuan*, Xiwen Lu, (2008). A new backtracking inexact BFGS method for symmetric nonlinear equations", Journal of computers and mathematics with applications, 55, 116-129.

[6] Mahammad Kabir Dauda, Mustafa Mamat, Mohamad Afendee bin Mohamed and Mahammad Yusuf Waziri (2019). Improved Quasi-Newton method via SR1 update for solving symmetric systems of nonlinear equations. Malaysian Journal of Fundamental and Applied Sciences Vol. 15, No.1, 117-120.

[7] M.K. Dauda, M. Mamat, M.Y. Waziri, F. Ahmad and F.S. Mohamad, (2016). Inexact CG-Method via SR1 Update for Solving Systems of Nonlinear Equations, Far East Journal of Mathematical Sciences, 100(11), 787-1804.

[8] M. K. Dauda, Mustafa Mamat, Mohamad Afendee Mohamed, FatmaSusilawati Mohamad and M.Y. Waziri, (2017). Derived Conjugate Gradient Parameter for Solving Symmetric Systems of Nonlinear Equations, *Far East Journal of Mathematical Sciences (FJMS)*, 102 (11)2599-2610.

[9] M. Y. Waziri and J. Sabi'u, (2015). A derivative-free conjugate gradient method and its global convergence for solving symmetric nonlinear equations, Int. J. Math. Math. Sci., 8 pp, Article ID 961487.

[10] M. Mamat, M. K. Dauda, M. A. bin Mohamed, M. Y. Waziri and F. S. Mohamad, H. Abdullah, (2018). Derivative free Davidon-Fletcher-Powell (DFP) for solving symmetric systems of nonlinear equations, IOP Conf. Series: Materials Science and Engineering 332, doi:10.1088/1757-899X/332/1/012030.

[11] Li, Q. & Li, D.H., (2011). *A Class of Derivative-free Methods for Large-Scale Nonlinear Monotone Equations*, IMA Journal of Numerical Analysis, 31(4), pp. 1625-1635.

[12] Waziri M.Y, Leong W.J, Mamat M and Moyi, A.U. (2013). Two-Step Derivative-Free Diagonally Newton's Method for Large-Scale Nonlinear Equations. *World Applied Sciences Journal* Vol 21, pp. 86-94.

[13] Gaohang Yu, (2010). A Derivative-Free Method for Solving Large-Scale Nonlinear Systems of Equations. Journal of Industrial and Management Optimization. Volume 6, Number 1,pp. 149–160. Doi:10.3934/Jimo.2010.6.149.

[14] A.S. Halilu1, M.K. Dauda, M.Y. Waziri, M. Mamat (2019). Open Journal of Science and Technology A Derivative-Free Decent Method via Acceleration Parameter for Solving Systems of Nonlinear Equations*2(3); 1-4.*

[15] A.S. Halilu and M. Waziri, (2018). An improved derivative-free method via double direction approach for solving systems of nonlinear equations, Journal of Ramanujan Mathematical Society, 33(1), 7589.

[16] 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.

[17] Jamilu Sabi'u, Abdullah Shah, Mohammed Yusuf Waziri & Muhammad Kabir Dauda (2020). A New Hybrid Approach for Solving Large-scale Monotone Nonlinear Equations. J. Math. Fund. Sci., Vol. 52, No. 1, 17-26. DOI: 10.5614/j.math.fund.sci.2020.52.1.2

[18] Mathews J. H, Fink K. D. (1999).*Numerical method using MATLAB*. Prentice Hull, Upper saddle river, NJ 07458.

[19] Elizabeth D. Dolan, Jorge J. Mor´e(2002). Benchmarking Optimization Software with Performance Profiles. Mathematical Programming 91, 201-213.

[20] M. K. Dauda, Abubakar S Magaji, Habib Abdullah, Jamilu Sabi'u and Abubakar S. Halilu (2019). A New Search Direction via Hybrid Conjugate Gradient Coefficient for Solving Nonlinear System of Equations. Malaysian Journal of Computing and Applied Mathematics, Vol 2(1): 8-15.

[21] Weijun Zhou & Dongmei Shen (2014). An Inexact PRP Conjugate Gradient Method for Symmetric Nonlinear Equations, Numerical Functional Analysis and Optimization, 35:3, 370-388, DOI: 10.1080/01630563.2013.871290.