Applied Mathematics and Computational Intelligence
Volume 10, No.1, Dec 2021 [45-56]

Modified Non Classical Conjugate Gradient Parameter for Solving Nonlinear Equations

M. K. Dauda[1]

[1]Department of Mathematical Sciences, Kaduna State University, Nigeria

* Corresponding author: mkdfika@kasu.edu.ng

Received: 16 September 2020; Accepted: 10 March 2021; Available online: 22 March 2021

ABSTRACT

In this work, a derived derivative-free conjugate gradient method for large-scale symmetric nonlinear equations is proposed. The basic idea of the method combines the Newton's direction of conjugate gradient method and Quasi Newton from the work of Andrei using a standard secant equation. The aim is to reduce the number of iterations, the CPU time and function evaluation of a given functions. The search direction is obtained using the normal frame of the conjugate gradient method via a new non-monotone line-search procedure. The proposed scheme was implemented using MATLAB and the computational results for the set of problems show that the algorithm substantially outperforms the known conjugate gradient methods by Andrei. The derivative-free nature of the proposed method gives it advantage to solve relatively large-scale problems by avoiding the computation of Jacobian inverse. The computed parameter β_k improved the efficiency of the algorithm by reducing the function values significantly. As compared to some existing methods, the numerical results on the given benchmark test problems show that the proposed algorithms are practically effective. It is accurate in terms of time of computation and valid in terms of number of iterations. Thus, suitable for solving nonlinear equations problems.

Keywords: Conjugate gradient, derivative free, line-search, nonlinear equations, non-monotone.

1 INTRODUCTION

Consider nonlinear equations of the form

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

where $F(X) = (f_1(X), f_2(X), \cdots, f_n(X))^T$, $X = (x_1, x_2, \cdots, x_n)$ is continuous on R [1].

Many problems of the form (1) arises from various applications in mathematics such as differential equations, optimization, operations research and so on. Sometimes, effective variational inequalities methods for solving unconstrained optimization problems of the form

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

is also effective in solving problems of the form (1) by converting (2) into nonlinear system of equations using a penalty function $f(x) = \|F(x_k)\|_2^2$. The unconstrained problem (2) can be viewed as the nonlinear systems (1) as the first-order optimality condition of the problem (2), where $F(x)$ is the gradient of $f: R^n \to R$ if it exists and f is assumed to be continuous and bounded below. Fermat's extremum theorem suggests that if a point x^* is the local minimizer of the unconstrained optimization problem (2) then problem (1) holds [2]. Thus, studying the iterative algorithms for solving nonlinear equations is of great importance. The Newton method is one of the most effective method for solving such equations due to its rapid convergence and easy implementation. Unfortunately, The Newton method requires the computation of the Jacobian matrix, sometimes it fails if the derivatives functions are zero [3,4,5]. The Quasi-Newton method was developed to overcome the major shortcomings associated with the famous Newton method. However, it inherits the problem of storing $n \times n$ matrices throughout the iteration process, which makes it unsuitable for large-scale problems [6]. Many researchers [2, 4, 6, 7], developed crucial approaches to overcome the storage problem associated with Quasi-Newton method by developing the matrix-free method. The following is the most widely use the search direction

$$d_k = -B_k^{-1}F(x_k), \tag{3}$$

with

$$x_{k+1} = x_k + \alpha_k d_k. \tag{4}$$

In equation (3), the term B_k represents the exact Hessian matrix $\nabla^2 f(x_k)$ in the case of Newton's method while in Quasi-Newton method represents the approximation of the Hessian matrix. The approximation of the Hessian matrix B_k, must satisfy the Secant equation as follows:

$$B_k s_{k+1} = y_k \tag{5}$$

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

Given an initial starting point $x_0 \in R^n$, popular iterative methods, such as Newton method, Quasi-Newton method or conjugate gradient method, for solving (2) uses an updating rule defined as in equation (4), which is used to generate the updated iterate with different line search, where α_k and d_k denote step size and search direction, respectively.

In this research, the current work [9] is considered to improve in solving equation (1). The algorithm is built on the strategy of derivative free line search technique [10]. However, their algorithm requires descent directions with respect to the squared norm of the residual. This means the computation of a directional derivative, or its good approximation is required at every iteration.

In this paper, the sections are organized as follows: section 1 gives the brief introduction of the existing problem, section 2 presents the new method and its algorithm based on the modified secant equation while in section 3 reports a detailed explanation on a numerical result. Finally, section 4 reports some concluding remarks and recommendation.

2 THE PROPOSED METHOD: MODIFIED NON CLASSICAL CONJUGATE GRADIENT PARAMETER (NCCG)

Consider equation (4), to compute the next iteration $x_{k+1} = x_k + \alpha_k d_k$, the choice of linesearch procedure, which selects the step size $\alpha_k > 0$ is required. Also, the selection of the parameter β_{k+1} which determines the next search direction is also required. The parameter β_{k+1} can be chosen in a variety of ways, [11, 12, 13]. This section presents the non classical conjugate gradient parameter, resulted from the following classical direction [9, 14]. Notice that Newton direction for solving (1) is given by

$$d_{k+1} = -\nabla^2 f(x_{k+1})^{-1} g(x_{k+1}). \tag{6}$$

Quasi-Newton method use

$$d_{k+1} = -\theta_{k+1} g(x_{k+1}) + \beta_k s_k, \tag{7}$$

where $g(x_k) = \nabla f(x_k)$. The line search α_k is selected along search direction d_k. The scalar β_k is called Conjugate Gradient parameter. The θ_{k+1} is a parameter in $0 < \theta_k < 1$. Observe that, if $\theta_k = 1$ then (7) becomes steepest decent algorithm and if $\theta_k = 0$, the (7) becomes classical conjugate algorithm based on the choice of the scalar parameter β_k. The iterative process is initialized with x_0 and $d_0 = -g_0$. To develop an algorithm for solving (1), we choose parameter β_k in equation (7) in such a way that d_{k+1} satisfy the newton direction.

The need to determine the more effective β_k in (7) is motivated by the work of [9,14].

Now, equating (6) and (7) gives

$$-\nabla^2 f(x_{k+1})^{-1} g(x_{k+1}) = -\theta_{k+1} g(x_{k+1}) + \beta_k s_k, \tag{8}$$

where $\nabla^2 f(x_{k+1}) = J(x_k)$ is the Jacobian of f at x_k

$$-J(x_k)^{-1} g(x_{k+1}) = -\theta_{k+1} g(x_{k+1}) + \beta_k s_k. \tag{9}$$

By multiplying both side of (9) with $J(x_k)$ gives

$$-J(x_k)J(x_k)^{-1} g(x_{k+1}) = -J(x_k)\theta_{k+1} g(x_{k+1}) + \beta_k J(x_k) s_k. \tag{10}$$

This can be simplified as

$$-g(x_{k+1}) = -J(x_k)\theta_{k+1} g(x_{k+1}) + \beta_k J(x_k) s_k. \tag{11}$$

Again, by multiplying both side of (11) with s_k^T yields

$$-s_k^T g(x_{k+1}) = -J(x_k)\theta_{k+1} g(x_{k+1}) s_k^T + \beta_k s_k^T J(x_k) s_k. \tag{12}$$

From the Secant condition

$$J(x_k) s_k = y_k. \tag{13}$$

Then, from equation (13)

$$J(x_k)s_k = y_k \rightarrow (J(x_k)s_k)^T = (y_k)^T = s_k^T\big(J(x_k)\big)^T, \tag{14}$$

also,

$$s_k = J(x_k)^{-1}y_k \rightarrow s_k^T = (J(x_k)y_k)^T = y_k^T(J(x_k)^{-1})^T, \tag{15}$$

since it is symmetric, then $J(x_k)^T = J(x_k)$, therefore equation (12) can be re written as $-s_k^T g(x_{k+1}) = -\theta_{k+1}g(x_{k+1})y_k^T + \beta_k y_k^T s_k$.

This implies,

$$\beta_k y_k^T s_k = \theta_{k+1}g(x_{k+1})y_k^T - s_k^T g(x_{k+1}), \tag{16}$$

hence,

$$\beta_k = \frac{\theta_{k+1}g(x_{k+1})y_k^T - s_k^T g(x_{k+1})}{y_k^T s_k}, \tag{17}$$

with $\theta_{k+1} = \frac{s_k^T s_k}{s_k^T s_k} \in (0,1)\forall, \quad k \geq 0,\ s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$.

Thus,

$$d_{k+1} = -\theta_{k+1}g(x_{k+1}) + \beta_k^* s_k, \quad \theta_{k+1} \in (0,1). \tag{18}$$

The scheme will become $x_{k+1} = x_k + \alpha_k d_k$, where α_k is determined using

$$\alpha_k = max\{s, rs, r^2 s, \cdots\}, \tag{19}$$

satisfying $-g(x_k + \alpha_k d_k)^T d_k \geq \omega\alpha_k \|g(x_k + \alpha_k d_k)\| \|d_k\|^2$ with $r, \omega \in (0,1)$.

It is clear that the line search is well defined.

Thus,

$$d_{k+1} = \begin{cases} -F(x_{k+1}), & k = 0, \\ -\theta_{k+1}F(x_{k+1}) + \beta_k^* d_k, & k \geq 1, \end{cases} \tag{20}$$

where $\beta_k^* = \frac{\theta_{k+1}g(x_{k+1})y_k^T - s_k^T g(x_{k+1})}{y_k^T s_k}$ as defined in equation (17).

In order to ensure convergence properties and to improve the overall efficiency for new CG algorithm. The non monotone line search procedure in [10] is used. They easily guarantee that the curvature condition $s_k^T y_k > 0$ is fulfilled for all k, where $s_k = \alpha_k d_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. Thus, well-suited for our purposes.

2.1 Algorithm of Modified Non Classical Conjugate Gradient Parameter (NCCG)

Step 1. Initializations. Choose the initial point $x_0 \in R^n$, $d_0 = -g_0$ and set $k = 0$.
Step 2. Compute $g(x_k)$ and the search direction d_k.
Step 3. Determine the step size α_k using the non-monotone line search (19).
Step 4. Check the stopping condition. If yes, stop, otherwise go to next step.
Step 5. Update $x_{k+1} = x_k + \alpha_k d_k$.
Step 6 Update search direction d_k using (20).
Step 7. Set $k = k + 1$ and go to Step 2.

The non-monotone line search used in the algorithm greatly improve the efficiency of the new scheme.

3 RESULTS AND DISCUSSION

This section presents the numerical experiment of the current research. The research used number of iterations (NI), CPU time to complete the process (CPUT) and number of function evaluations ($\|F(x_k)\|$) as metrics for the comparison, where the notation, $\|.\|$ denotes the Euclidean norm. The experiment gives the role of the parameter β_k^* (17) in reducing the number of iterations (NI), the number of function evaluations and the CPU time (CPUT) required to solve the problem. The performance analysis given by Dolan and Moore [15] is adopted to analyze our findings. A set of benchmark problems is represented by P be and S denotes the set of algorithms. Dolan and Moore [15] defined $t_{p,s}$ to be the number of iterations, the number of function evaluations or the CPU time in seconds (as the case may be) required to solve the problem $p \in P$ by algorithm $s \in S$. Comparison of each of the three measures is based on the performance ratio defined by

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

The performance profile is given by

$$\rho_s(\tau) = \frac{\left|\{p\in P:\log_2(r_{p,s})\leq\tau\}\right|}{|P|}.$$

The scheme solved some benchmark test problems using different initial starting points with different dimensions as contained in Table 1-6. Both algorithms were coded in MATLAB R2017a [16] and run on a PC with intel Core (TM) i5-8250u processor with 4 GB of RAM and CPU 1.60 GHZ. The dimensions whose ranging from $n = 10$ and $n = 10{,}000$ are used on a test Problems as in the appendix.

A particular algorithm is said to outperform its counterpart if the number of iterations (NI), CPU time (CPUT) to complete the process and/or number of function evaluations is less than that of its counterpart. In each of the following Table, the S/N means the serial number of a particular problem, Dim represents the dimension, ISP indicates the initial starting point, NI means number of iterations, while time to compute a particular problem is denoted as CPUT and the value of function evaluation is given by $\|F(x_k)\|$. The proposed algorithm is denoted as NCCG which means non classical conjugate gradient and CCG means the classical conjugate gradient.

Table 1: The numerical comparison of NCCG and CCG methods for problem 1

			NCCG			CCG	
S/N	**Dim**	**ISP**	**NI**	**CPUT**	$\|\|F(x_k)\|\|$	**NI**	**CPUT**	$\|\|F(x_k)\|\|$
1	10	0.2	4	1.130041	8.51×10^{-4}	6	0.143981	2.30×10^{-4}
	100		5	0.004332	1.41×10^{-5}	6	0.004506	7.28×10^{-4}
	1000		5	0.003294	4.47×10^{-5}	7	0.003664	4.60×10^{-4}
	10000		5	0.11432	1.41×10^{-4}	8	0.022551	2.91×10^{-4}
	100000		5	0.340451	4.47×10^{-4}	8	0.289295	9.21×10^{-4}
	10	0.3	6	0.001291	5.90×10^{-5}	5	0.001763	9.400500
	100		6	0.001394	1.86×10^{-4}	6	0.001791	5.94×10^{-4}
	1000		6	0.003372	5.90×10^{-4}	7	0.004359	3.76×10^{-4}
	10000		7	0.028916	2.62×10^{-6}	8	0.032299	2.38×10^{-4}
	100000		7	0.308379	8.27×10^{-6}	8	0.332582	7.52×10^{-4}

Table 2: The numerical comparison of NCCG and CCG methods for problem 2

			NCCG			CCG	
S/N	**Dim**	**ISP**	**NI**	**CPUT**	$\|\|F(x_k)\|\|$	**NI**	**CPUT**	$\|\|F(x_k)\|\|$
2	10	0.3	6	0.039174	5.90×10^{-5}	5	0.002866	9.40×10^{-4}
	100		6	0.001288	1.86×10^{-4}	6	0.001639	5.94×10^{-4}
	1000		6	0.002898	5.90×10^{-4}	7	0.003111	3.76×10^{-4}
	10000		7	0.022872	2.62×10^{-6}	8	0.036739	2.38×10^{-4}
	100000		7	0.231628	8.27×10^{-6}	8	0.257751	7.52×10^{-4}
	10	0.4	4	0.001074	8.28×10^{-4}	5	0.001156	2.58×10^{-4}
	100		5	0.001948	2.44×10^{-5}	5	0.001513	8.17×10^{-4}
	1000		5	0.002523	7.73×10^{-5}	6	0.006202	5.17×10^{-4}
	10000		5	0.020067	2.44×10^{-4}	7	0.017179	3.27×10^{-4}
	100000		5	0.247454	7.73×10^{-4}	8	0.161788	2.07×10^{-4}

Table 3: The numerical comparison of NCCG and CCG methods for problem 3

			NCCG			CCG	
S/N	**Dim**	**ISP**	**NI**	**CPUT**	$\|\|F(x_k)\|\|$	**NI**	**CPUT**	$\|\|F(x_k)\|\|$
3	10	0.1	3	0.00138	1.85×10^{-4}	7	0.05084	6.87×10^{-4}
	100		3	0.001256	6.91×10^{-4}	8	0.001458	8.95×10^{-4}
	1000		4	0.002747	1.04×10^{-5}	10	0.003827	4.54×10^{-4}
	10000		4	0.018833	3.31×10^{-5}	11	0.032131	5.75×10^{-4}
	100000		4	0.199151	1.05×10^{-4}	12	0.356706	7.27×10^{-4}
	10	0.6	14	0.001544	8.80×10^{-4}	11	0.002995	8.97×10^{-4}
	100		10	0.002066	6.25×10^{-6}	13	0.002503	7.87×10^{-4}
	1000		10	0.011984	7.78×10^{-4}	15	0.006531	4.68×10^{-4}
	10000		9	0.04547	7.03×10^{-6}	16	0.045158	6.06×10^{-4}
	100000		9	0.48783	4.49×10^{-5}	17	0.508228	7.69×10^{-4}

Table 4: The numerical comparison of NCCG and CCG methods for problem 4

			NCCG			CCG		
S/N	**Dim**	**ISP**	**NI**	**CPUT**	$\|\|F(x_k)\|\|$	**NI**	**CPUT**	$\|\|F(x_k)\|\|$
4	10	0.3	89	0.008401	5.32×10^{-4}	5	0.018059	8.95×10^{-4}
	100		90	0.001346	1.34×10^{-4}	6	0.025939	5.67×10^{-4}
	1000		90	0.003513	4.23×10^{-4}	7	0.059255	3.58×10^{-4}
	10000		91	0.034007	9.06×10^{-8}	8	0.481044	2.27×10^{-4}
	100000		91	0.291538	2.87×10^{-7}	8	5.352658	7.17×10^{-4}
	10	0.5	6	0.001167	2.40×10^{-5}	5	0.001656	8.61×10^{-4}
	100		6	0.001423	7.58×10^{-5}	6	0.002133	5.45×10^{-4}
	1000		6	0.003551	2.40×10^{-4}	7	0.004556	3.44×10^{-4}
	10000		6	0.031712	7.58×10^{-4}	8	0.036236	2.18×10^{-4}
	100000		7	0.308359	1.99×10^{-6}	8	0.37573	6.89×10^{-4}

Table 5: The numerical comparison of NCCG and CCG methods for problem 5

			NCCG			CCG		
S/N	**Dim**	**ISP**	**NI**	**CPUT**	$\|\|F(x_k)\|\|$	**NI**	**CPUT**	$\|\|F(x_k)\|\|$
5	10	0.2	11	0.047317	3.37×10^{-4}	7	0.002663	1.38×10^{-4}
	100		12	0.00189	7.05×10^{-6}	7	0.003266	4.36×10^{-4}
	1000		12	0.005069	2.23×10^{-5}	8	0.00891	1.65×10^{-4}
	10000		12	0.044967	7.05×10^{-5}	8	0.060027	5.23×10^{-4}
	100000		12	0.459352	2.23×10^{-4}	9	0.703779	1.98×10^{-4}
	10	0.3	6	0.001542	3.86×10^{-5}	7	0.001947	1.22×10^{-4}
	100		6	0.001832	1.22×10^{-4}	7	0.002061	3.85×10^{-4}
	1000		6	0.00545	3.86×10^{-4}	8	0.004526	1.46×10^{-4}
	10000		7	0.038168	2.49×10^{-6}	8	0.040112	4.62×10^{-4}
	100000		7	0.472431	7.88×10^{-6}	9	0.423607	1.75×10^{-4}

Table 6. The Numerical Comparison of NCCG and CCG methods for problem 6

			NCCG			CCG		
S/N	**Dim**	**ISP**	**NI**	**CPUT**	$\|\|F(x_k)\|\|$	**NI**	**CPUT**	$\|\|F(x_k)\|\|$
6	10	0.7	4	0.00139	1.44×10^{-5}	5	0.015706	3.92×10^{-4}
	100		4	0.001618	4.54×10^{-5}	6	0.001932	1.48×10^{-4}
	1000		4	0.004974	1.44×10^{-4}	6	0.00477	4.68×10^{-4}
	10000		4	0.026852	4.54×10^{-4}	7	0.041989	1.76×10^{-4}
	100000		5	0.303721	2.97×10^{-7}	7	0.431781	5.57×10^{-4}
	10	0.2	7	0.001653	2.80×10^{-5}	5	0.001909	7.44×10^{-4}
	100		7	0.001754	8.86×10^{-5}	6	0.002076	2.81×10^{-4}
	1000		7	0.004562	2.80×10^{-4}	6	0.005727	8.87×10^{-4}
	10000		7	0.040687	8.86×10^{-4}	7	0.039254	3.34×10^{-4}
	100000		8	0.472439	1.07×10^{-6}	8	0.448025	1.26×10^{-4}

Therefore, after comparing the two algorithms, notice that the new algorithm performs better in Problems 1, 2, 4 and 6, as compare to the existing algorithm. While in problems 3 and 5, both the algorithms compete remarkably and never failed to converge. The efficiency of the parameter β_k^* in (20) gives a numerical evidence that it has greatly improve the performance, since it has remarkably reduced the number of iterations (NI), CPU time (CPUT) to compute a solution and the value of function evaluations $\|F(x_k)\|$, thereby indicating that the solution obtained is a true approximation of the exact solution compared to the existing method.

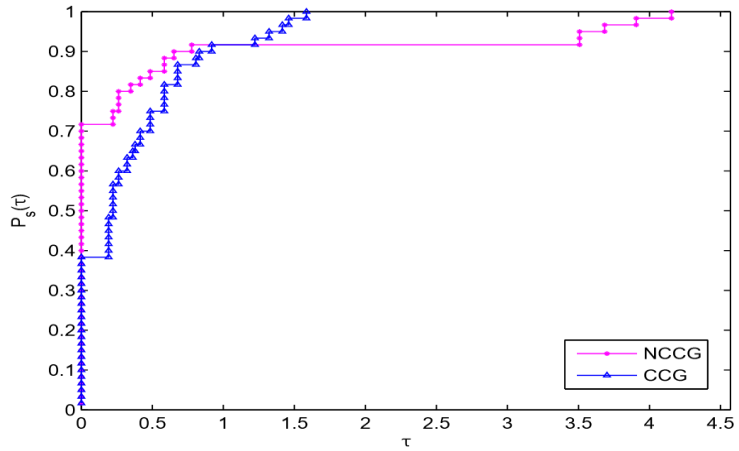

Figure 1: Performance profile of NCCG and CCG methods with respect to the number of iterations.

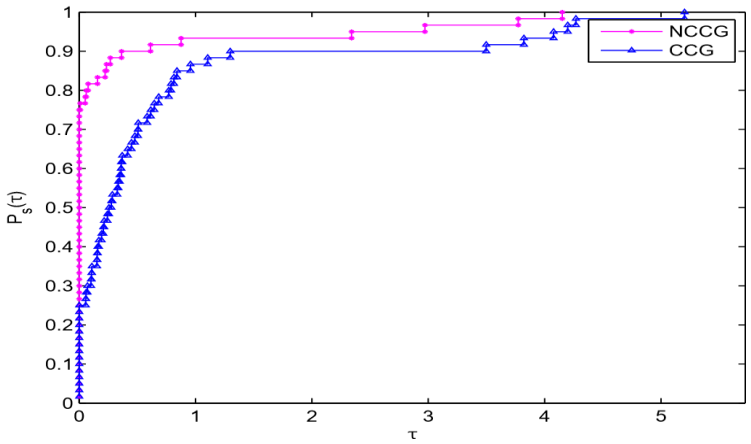

Figure 2: Performance profile of NCCG and CCG methods with respect to the CPU time.

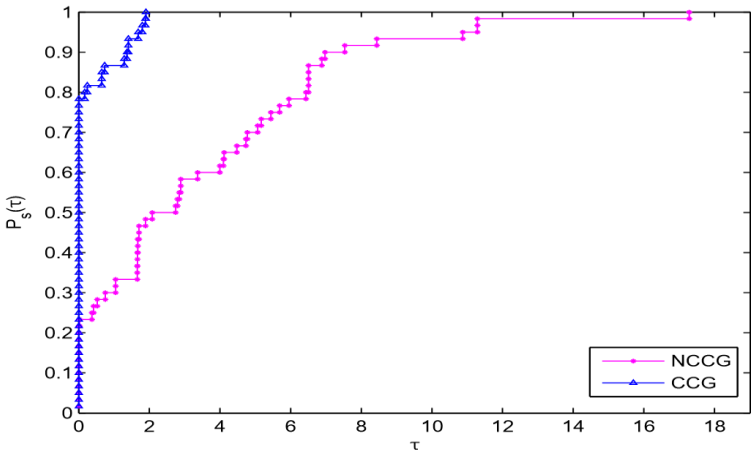

Figure 3: Performance profile of NCCG and CCG methods with respect to the functions evaluation.

The performance profile used on numerical results was derived by Dolan and Moore [15] and used to plot Figures 1, 2 and 3. Figure 1 gives the performance profiles for the number of iterations for the solved problems. In the figure the new algorithms outperformed the existing method when the $\tau \geq 0.1$. While Figure 2 shows the CPU time of the new algorithms is less than that of its counterpart when the value is $\tau \leq 0.2$. Figure 3 shows that the new algorithms outperform the other methods in terms of the number of function evaluations when $\tau \geq 0.2$. The overall performance is remarkably commendable since it solves a higher percentage of the given problems.

4 CONCLUSION

In this article, numerical results of the entire work are presented. The performance of the existing work in [9] titled CCG and that of our proposal in algorithms NCCG are compared. The comparison are in terms of iterations (NI), CPU times (CPUT) and function evaluations. The new parameter presented enhanced the overall behavior of the method, in other words, it outperforms the method [9] in all the metrics. It proved to be efficient in terms iterations (NI), reliable in terms of CPU time (CPUT) and accurate in terms function evaluations when the benchmark test in the appendix are used. The line search procedure used preserved the conjugacy conditions of the overall algorithm. Thus, making the profiles reveal an appreciable improvement of the efficiency as long as the robustness. Therefore, it is recommended for the solution of Non-Linear Equations of the form (1). In an effort to verify the effectiveness of the proposed method, an ardent researcher may confirm the global convergence result of the proposed method.

ACKNOWLEDGEMENT

The authors are grateful to the reviewers for their suggestions and corrections.

REFERENCES

[1] D. H. Li and M. Fukushima, "A Derivative-Free Line Search and Global Convergence of Broyden-Like Methods for Nonlinear Equations," *Optimization Methods and Software*, vol. 13, pp. 181-201, 2000.

[2] A. M. Awwal, L. Wang, P. Kumam, and H. Mohammad, "A Two-Step Spectral Gradient Projection Method for System of Nonlinear Monotone Equations and Image Deblurring Problems," *Symmetry*, vol. 12, no. 6, p. 874, 2020.

[3] M. Mamat, M. K. Dauda, M. A. Mohamed, M. Y. Waziri and F. S. Mohamad, H. Abdullah, "Derivative Free Davidon-Fletcher-Powell (DFP) for Solving Symmetric Systems of Nonlinear Equations," *IOP Conf. Series: Materials Science and Engineering 332*, 2018, doi:10.1088/1757-899X/332/1/012030.

[4] M. K. Dauda, M. Mamat, M. Y. Waziri, F. Ahmad and F. S. Mohamad, "Improved Quasi-Newton Method Via PSB Update for Solving Systems of Nonlinear equations," in *AIP Conference Proceedings (ICOQSIA 2016)*, vol. 1782, no. 1, 2016, doi: 10.1063/1.4966066.

[5] W. Zhou and D. Shen, "Convergence Properties of an Iterative Method for Solving Symmetric Non-Linear Equations," *Journal of Optimization Theory and Applications*, vol. 164, pp. 277- 289, 2015.

[6] M. K. Dauda, M. Mamat, M. Y. Waziri, F. Ahmad and F. S. Mohamad, "Inexact CG-Method via SR1 Update for Solving Systems of Nonlinear Equations," *Far East Journal of Mathematical Sciences (FJMS)*, vol. 100, no. 11, pp. 1787-1804, 2016, doi: doi.org/10.17654/MS100111787.

[7] M. K. Dauda, M. Mamat, M. A. Mohamed and M. Y. Waziri, "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, pp. 117-120, 2019, doi: doi.org/10.11113/mjfas.v15n2019.1085.

[8] M. Mamat, M. K. Dauda, F. S. Mohamad, A. S. Magaji and M. Y. Waziri, "Derivative Free Conjugate Gradient Method via Broyden's Update for Solving Symmetric Systems of Nonlinear Equations," *Journal of Physics: Conference Series*, vol. 1366, p. 012099, 2019, doi:10.1088/1742-6596/1366/1/012099.

[9] N. Andrei, "Accelerated Conjugate Gradient Algorithm with Finite Difference Hessian/Vector Product Approximation for Unconstrained Optimization," *Journal of Computational and Applied Mathematics*, vol. 230, pp. 570-582, 2009.

[10] J. Sabi'u, A. Shah, M. Y. Waziri and M. K. Dauda, "A New Hybrid Approach for Solving Large-scale Monotone Nonlinear Equations," *J. Math. Fund. Sci.*, vol. 52, no. 1, pp. 17-26, 2020, doi: 10.5614/j.math.fund.sci.2020.52.1.2.

[11] M. K. Dauda, A. S. Magaji, H. Abdullah, J. Sabi'u and A. S. Halilu, "A New Search Direction via Hybrid Conjugate Gradient Coefficient for Solving Nonlinear System of Equations," *Malaysian Journal of Computing and Applied Mathematics*, vol. 2, no. 1, pp. 8-15, 2019, doi: doi.org/10.37231/myjcam.2019.2.1.24.

[12] M. K. Dauda, M. Mamat, M. A. Mohamed and N. S. A. Hamzah, "Hybrid Conjugate Gradient Parameter for Solving Symmetric Systems of Nonlinear Equations," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 1, pp. 539-543, 2019, doi: 10.11591/ijeecs.v16.i1.

[13] X. Fang and Q. Ni, "A New Derivative-Free Conjugate Gradient Method for Large-Scale Nonlinear Systems of Equations," *Bulletin of the Australian Mathematical Society*, vol. 95, no. 3, pp. 500-511, 2017, doi:10.1017/S0004972717000168.

[14] N. A. Bakar and M. Rivaie, "Solving Large System of Linear Equation Using Successive Over-Relaxation, Conjugate Gradient and Preconditioned Conjugate Gradient," *Applied Mathematics and Computational Intelligence (AMCI)*, vol. 6, pp. 83-98, 2017.

[15] E. D. Dolan and J. J. Mor´e, "Benchmarking Optimization Software with Performance Profiles," *Mathematical Programming*, vol. 91, pp. 201-213, 2002.

[16] J. H. Mathews and K. D. Fink, *Numerical method using MATLAB*. Prentice Hall, Upper Saddle River, NJ 07458, 1999.

[17] A. Galántai, "Always convergent methods for nonlinear equations of several variables," *Numerical Algorithms*, vol. 78, no. 2, pp. 625–641, 2017, doi: 10.1007/s11075-017-0392-z.

APPENDIX

This section presents the test problems used for the numerical experiments. The test problems are taken from the Estonian test problem collection [8] and [17].

Problem 1. (System of Nonlinear Equations)

$$F_i(x) = e^{x_i^2-1} - cos\left(1 - x_i^2\right) = \vec{0};\ i\ =\ 1,2,3,\ldots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$.

Problem 2. (System of n Nonlinear Equations)

$$F_i(x) = \left(1 - x_i^2\right) + x_i(1 + x_i x_{n-2} x_{n-1} x_n) - 2 = \vec{0}; i = 1,2,3,\cdots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$.

Problem 3. (System of n Nonlinear Equations)

$$F_i(x) = x_i -\ 0.1x_{i+1}^2 = \vec{0}; i\ = 1,2,3,\ldots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$.

Problem 4. (System of Non-Smooth Equations)

$$F(1) = x_1 * (x_1^2 + x_2^2) - 1 = \vec{0},$$
$$F(i) = x_i * \left((x_{i-1})^2 + 2x_i^2 + x_{i+1}^2\right) - 1 = \vec{0},$$
$$F(n) = x_n * (x_{n-1}^2 + x_n^2) = \vec{0},$$

$$i\ =\ 1,2,3,\ldots,n.$$

Problem 5. (Weber-Werner)

$$F_1(x) = x_1^2 - 2x_1\ +\ \frac{1}{3}x_2^3 + \frac{2}{3} = \vec{0},$$
$$F_2(x) = x_1^3 - x_1 x_2 -\ 2x_1 +\ 0.5x_2^2 +\ 1.5 = \vec{0}.$$

Problem 6. (Allgower-Georg)

$$F_1(x) = (x_1 - x_2^2)\left(x_1 - sin\ (x_2)\right) = \vec{0},$$
$$F_2(x) = (cos\ (x_2) - x_1)\left(x_2 - cos\ (x_1)\right) = \vec{0}.$$