Malaysian Journal of Computing and Applied Mathematics 2019, Vol 2(1): 8-15
©Universiti Sultan Zainal Abidin
(Online)

MALAYSIAN JOURNAL OF COMPUTING AND APPLIED MATHEMATICS

A New Search Direction via Hybrid Conjugate Gradient Coefficient for Solving Nonlinear System of Equations

M K Dauda[a], Abubakar S Magaji[a], Habib Abdullah[b], Jamilu Sabi'u[c] and Abubakar S Halilu[b]

[a]Department of Mathematical Sciences, Kaduna State University, Nigeria
[b]Department of Mathematics and Computer Science, Sule Lamido University, Nigeria
[c]Department of Mathematics Northwest University Kano, Nigeria

*Corresponding author: mkdfika@gmail.com

Received: 07/01/2019, Accepted: 22/05/2019

Abstract

Conjugate gradient methods are widely used for unconstrained optimization problems. Most of the conjugate gradient methods do not always generate a descent search direction. In this article, a new search direction via hybrid conjugate gradient method by convex combination of the earlier works is used and extended for the solution of nonlinear system of equations. The proposed search direction generates a descent direction at every iteration and without convexity assumption on the objective function. The new method possesses, preserved and inherits the properties of the Conjugate Gradient (CG) coefficient used. The proposed hybridized method played a useful, vital and powerful role in solving large-scale nonlinear system of equations as indicated in the numerical results.

Keywords: Unconstrained optimization; Conjugate gradient; Search direction; Nonlinear equations.

Introduction

Consider the equation below

$$F(x) = 0 \tag{1}$$

where $F: R^n \to R^n$ and assumed to satisfy that (i) F is continuously differentiable in an open convex set S (ii) There exist 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^*. Most problems that emanates from the work of engineers, physicists, mathematicians and many other scientists are Nonlinear in nature.

Many researchers (Yuana, and Zhang, 2015; Liu, and Li, 2015; Mamat et al, 2016; Dauda et al, 2016) developed different iterative methods for solving nonlinear systems of equations. However, some of the methods developed are not suitable for solving large-scale nonlinear systems of equations. This is due to the need to solve Jacobian matrix, an approximation of Jacobian matrix at each iteration as in Newton method or inefficiency in solving complicated problems. Among the iterative methods for solving (1) are Newton's method, Quasi-Newton's method, Diagonal Broyden-like method etc. but the prominent method

for finding the solution of (1) is the classical Newton's method that generates a sequence of iterates x_k from a given initial point x_0 via

$$x_{k+1} = x_k + s_k$$

The method generates an iterative sequence x_k from a given initial guess vector x_0 in the neighborhood of x^* from the following algorithm

Newton Algorithms

For $k = 0,1,2,\cdots$ 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 and the equation of step 2 is 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^* (Yuana, and Zhang, 2015; Liu, and Li, 2015). To overcome these deficiencies, Kafaki 2011 gave a new conjugate gradient method based on the idea of Dai and Yuan, and showed that their methods always produces a descent direction and converges globally if the Wolfe conditions are satisfied. In the research conducted by Sabi'u et el 2017, a new hybrid conjugate gradient parameter for solving the system of nonlinear equation by the combination of the Dai-Yuan and Hestenes-Stiefel conjugate gradient parameters was suggested. The method solved relatively large-scale problems with lower storage requirement compared to some existing methods but yet needs improvement on benchmark test problems. For more on Hybrid Conjugate Gradient Coefficient and derivative free methods for Solving Nonlinear System of Equations see (Sabi'u et al, 2017; Waziri and Jamilu, 2015; Dauda et al, 2017; Waziri and Sabi'u, 2015; Babaie-Kafaki and Ghanbaric, 2012; Li and Fukushima, 2000) and the references therein.

In an effort to solve Newton's deficiency, A New descent Search Direction via Hybrid Conjugate Gradient Coefficient for Solving Nonlinear System of Equations is presented in this article. The Search Direction was obtained via a convex combination of the work of Dauda et al (2017) and Waziri et el (2015) and extend it to the solution of Nonlinear System of Equations. The next section of this paper present the derivation of the proposed method. In section 3, report on Implementation of the proposed method and discussion on Numerical results are presented. In the last section, conclusion was drawn based on the performance profile.

Derivation

The following direction is obtaind as a result of combining CG-Coefficient in the work of Waziri and Sabi'u (2015) with Dauda et el. (2016).

Recall that $\beta_k^M = \frac{(y_k - s_k)^T F(x_{k+1})}{y_k^T d_k}$ see (Waziri and Jamilu, 2015) (2)

and $\beta_k^J = \frac{(\theta y_k - s_k)^T g(x_{k+1})}{y_k^T d_k}$ $\theta = \frac{s_k^T s_k}{s_k^T s_k}$ see (Dauda et al, 2017) (3)

as defined. Recalling the classical Newton direction defined by
$d_{k+1} = F(x_{k+1}) + \beta_k d_k$ where $\beta_k$is the CG coefficient

With the concept of convex combination, thus
$\beta_k^{new} = \mu\beta_k^M + (1-\mu)\beta_k^J$ with $\mu\epsilon[0,1]$. (4)
If $\mu = 0$, then $\beta_k^{new} = \beta_k^J$
and
if $\mu = 1$, then $\beta_k^{new} = \beta_k^M$.
For $\mu\epsilon(0,1)$, supposed we have β_k^{new} as in (4)
$d_{k+1} = -F(x_{k+1}) + \beta_k^{new} d_k$

$d_{k+1} = -F(x_{k+1}) + [\mu\beta_k^M + (1-\mu)\beta_k^J]d_k$
Since both directions are descent, then
$F_k^T d_k = -c\|F_k\|^2$ thus
$F_{k+1}^T d_{k+1} = -c\|F_{k+1}\|^2$ hence
$F_{k+1}^T d_{k+1} = -F(x_{k+1}) + [\mu\beta_k^M + (1-\mu)\beta_k^J]d_k$ (5)
Multiply both side of (a) by $(F_{k+1}^T)^{-1}$
$d_{k+1} = \left(F_{k+1}^T\right)^{-1}(-F(x_{k+1})) + (F_{k+1}^T)^{-1}[\mu\beta_k^M + (1-\mu)\beta_k^J]d_k$ (6)
Substitute (2) and (3) in (6) thus
$d_{k+1} =$
$\left(F_{k+1}^T\right)^{-1}\left(-F(x_{k+1})\right) + \left(F_{k+1}^T\right)^{-1}\left[\mu\frac{(y_k - s_k)^T F(x_{k+1})}{y_k^T d_k} + (1-\mu)\frac{(\theta y_k - s_k)^T g(x_{k+1})}{y_k^T d_k}\right]d_k \; if \; k \geq 0$ (7)
Therefore, the iterative method used to solve (1) is formed by
$x_{k+1} = x_k + \alpha_k d_k, \; k = 0,1,2,\cdots,$ (8)
where $\alpha_k > 0$ is the step-size, x_k is the iterative point and d_k is the previous search direction. The search direction and the updates are described by (7) and (8) respectively. With $F: R^n \rightarrow R^n$ satisfying the three (3) in section 1 above, while the scalar parameter β_k^{new} is the improved conjugate gradient coefficient obtained. The use of the function $F(x_{k+1})$ approximate the gradient $\nabla f(x_k)$, and avoids computing exact gradient. It is clear that when $\|F(x_k)\|$ is small, then $g_k \approx \nabla f(x_k)$.

The choice of derivative-free line search used in Zhou and Shen (2014) is the best choice to compute the step length α_k. Therefore, Let $\omega_1 > 0, \omega_2 > 0$ and $q, r \in (0,1)$ be constants and let $\{\eta_k\}$ be a given positive sequence such that

$$\sum_{k=0}^{\infty}\eta_k < \eta < \infty \quad (9)$$

$$f(x_k + \alpha_k d_k) - f(x_k) \leq -\omega_1\|\alpha_k F(x_k)\|^2 - \omega_2\|\alpha_k d_k\|^2 + \eta_k f(x_k). \quad (10)$$

Let i_k be the smallest non negative integer i such that (10) holds for $\alpha = r^i$. Let $\alpha_k = r^i k$. The following gives the Algorithm of the proposed method.

Algorithm
Step 1: Given $x_0 = 1, \mu = 0.6, \epsilon = 10^{-4}$, set $k = 0$.
Step 2: Compute $F(x_k)$ and test a stopping criterion i.e. $\|F(x_k)\| \leq 10^{-4}$. If yes, then stop; otherwise continue with Step 3:
Step 3: Compute the step-length α_k using (9) and (10)
Step 4: Determine, B_k^{new}
Step 5: Compute $F(x_{k+1})$ and $(F^T(x_{k+1}))^{-1}$
Step 6: Set $x_{k+1} = x_k + \alpha_k d_k$.
Step 7: Compute search direction d_k using (7)
Step 8: Consider $k = k + 1$, and go to Step 3.

Numerical Result and Discussion

In this section, the numerical tests for the proposed method in application to some benchmark problems is presented. To ensure the reliability and validity of the method, the performance of the proposed method (labelled M) is compared with conjugate gradient method by Dauda et el. (2017) (labelled $M1$) and conjugate gradient method by Waziri et el. (2015) (labelled $M2$). The following parameters are set for the implementation of the results:

$$\omega_1 = \omega_2 = 10^{-4}, r = 0.2 \text{ and } \eta_k = \frac{1}{(k+1)^2}.$$

All the codes for all the three methods were written in Matlab 7.4 R2010a and run on a personal computer 1.8 GHz CPU processor and 4 GB RAM memory. Six benchmark test problems with different initial starting points (ISP) and dimensions (DIM) with $10^n, n = 1,2,3,4,5$

values were tested and solved. Problems 1-4 are from (Roose, 1990), while 5 and 6 are from Mamat et al. (2016) (See appendix). The most famous inexact line search in practice (Zhou and Shen, 2014) is used. The line search sufficiently decreased the function values along the ray $x_k + \alpha_k d_k > 0$. i.e. $x_{k+1} = x_k + \alpha_k d_k \leq F(x_k)$.

In the Tables 1-3, the numerical comparison of all methods are in terms of number of iterations (Iter) needed to converge to an approximate solution, CPU time in seconds (TIME) and the Norm of the objective function F at the approximate solution x^* were presented in columns respectively.

The symbol "-" in the tables indicates a failure due to memory shortages or/and when the number of iterations exceeds 1000, this entails no solution is reached. The iterations are terminated whenever $\|F(x_k)\| \leq 10^{-4}$ is satisfied.

However, from the table, it can be seen that in most of the experiments, the proposed method recorded less number of iterations and less CPU time(s) compared to the other two methods. This is due to the search direction is obtained without the computation and storage of Jacobian matrices. These remarkable improvement indicates the robustness and efficiency of the method, thus gives the scheme aptness to always produce a descent search direction. It is worth noting that, the proposed method converges to approximate solutions of problems with some of the initial starting point (ISP), these can be seen from Tables. This is also achieved from the logical and the good selection of the hybridization parameter α_k.

Figures 1 and 2 Shows the graphical performance of the entire results. The Figures are sketched using Matlab R2013a with data analyzed in Microsoft Excel 2016. Dolan and More (2002) performance profiles was adopted to compare the performance of the proposed method and other two methods (See Figures 1-2).

Table 1. Numerical Comparison of M, $M1$ and $M2$ methods for problem 1 ad 2.

			M			$M1$			$M2$		
	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 M, $M1$ and $M2$ methods for problem 3 ad 4.

			M			$M1$			$M2$		
	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

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

			M			$M1$			$M2$		
	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	2000	0.348713	1.1412	12	0.003284	6.58E-05
	100		16	0.002461	6.91E-04	2000	0.419275	4.4653	13	0.003611	4.66E-05
	1000		19	0.007444	4.78E-04	2000	1.136226	16.4158	14	0.010835	5.15E-05
	10000		21	0.070103	6.88E-04	2000	9.293672	52.3308	15	0.071868	6.31E-05

	100000		23	0.829604	8.55E-04	2000	96.63993	154.1355	16	0.814324	7.83E-05

The comparison was based on the number of iterations and Time (CPU time is seconds). In Figure 1, the line labelled M was at the top of the other line of $M1$ and $M2$, this indicates the performance of methods M outperformed that of $M1$ and $M2$, thus the proposed method is efficient with respect to the number of iterations.

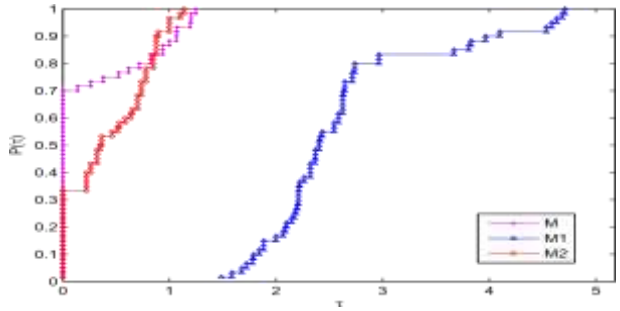

Figure 1. Performance profile of $M1$ and $M2$ methods compared to method M with respect to the number of iterations.

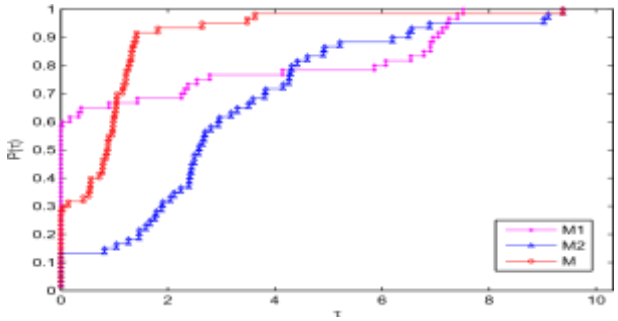

Figure 2. Performance profile of $M1$ and $M2$ methods compared to method M with respect to the CPU time.

Also, in Figure 2, the same process shows the performance of methods M in terms of CPU time outperformed that of $M1$ and $M2$, only that in some instances the method $M1$ and $M2$ compete with the proposed method, yet the proposed method is best in term of CPU time as the dimension increases. In general, the performance profile of Dolan and More′ shows that the hybrid of Dauda et al. (2017) and Waziri et al. (2015) outperformed the individual performance of Dauda et al. (2017) and Waziri et al. (2015) both in number of iteration and the CPU time with increase in dimension. For more details on performance profile, see (Mor´e and Wild, 2009).

Conclusion

The nonlinear system of equations are very interesting and essential in many disciplines. In this research, a new direction via hybrid conjugate gradient method for solving large-scale system of nonlinear equations was presented. The search direction was uniquely obtained and satisfies the sufficient descent property. The method is completely derivative-free. The main aim of this paper is to proffer solution to overcome the drawbacks bedeviling the solution of nonlinear systems of equations.

The proposed method addressed these problems practically and reduced the computational costs, storage requirements and processing time (CPU time) immensely. When applied on some benchmark problems (see appendix), the proposed method solved problems both in terms of CPU time and in terms of number of iterations respectively. The method is very suitable, efficient and accurate, hence recommended for solving large-scale problems whose

Jacobians are very difficult to compute or not available. An ardent researcher may however extend the proposed method to non-smooth equations. This will enhance the scheme to solve nonlinear equations with larger dimensions.

Conflict of Interests

There is no conflict of interest regarding the publication of this paper.

Appendix

In this section, 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 are taken from the Estonian test problem collection (Mamat et al, 2016) and (Roose, 1990).

1. (Allgower-Georg)

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

2. (Brezinski)

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

3. (Weber-Werner)

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

4. (Powell)

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

5. (Freudenstein)

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

6. (Powell)

$$f_1(x) = 10,000x_1x_2 - 1,$$
$$f_2(x) = exp\,(-x_1) + exp\,(-x_2) - 1.0001.$$

Reference

Yuana, G and Zhang, M. (2015). A Three-Terms Polak-Ribière-Polyak Conjugate Gradient Algorithm for Large-Scale Nonlinear Equations. Journal of Computational and Applied Mathematics, 286, 186-195.

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

Mamat, M, Dauda, M K, Waziri, M Y, Ahmad, F and Mohamad, F S. (2016). Improved Quasi-Newton Method via PSB Update for Solving Systems of Nonlinear Equations, AIP Conference Proceedings 1782, 030009; doi:10.1063/1.4966066.

Dauda, M K, Mamat, M, Waziri, M Y, Ahmad, F and Mohamad, F S. (2016). Inexact CG-Method via SR1 Update for Solving Systems of Nonlinear Equations. Far East Journal of Mathematical Sciences, 100(11), 1787-1804.

Kafaki, S B. (2011). A Hybrid Conjugate Gradient Method Based on a Quadratic Relaxation of the Dai-Yuan Hybrid Conjugate Gradient Parameter. Journal of Mathematical Programming and Operations Research, 62(7), 929-941.

Sabi'u, J, Waziri, M Y, Idris, A. (2017). A New Hybrid Dai-Yuan and Hestenes-Stiefel Conjugate Gradient Parameter for Solving System of Nonlinear Equations. MAYFEB Journal of Mathematics, 1, 44-55.

Waziri, M Y, and Jamilu, S. (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.

Dauda, M K, Mamat, M, Mohamed, M A, Mohamad, F S, and Waziri, M Y. (2017), Derived Conjugate Gradient Parameter For Solving Symmetric Systems of Nonlinear Equations. Far East Journal of Mathematical Sciences, 102(11), 2599-2610.

Waziri, M Y, and Sabi'u, J. (2015). A Derivative-Free Conjugate Gradient Method and Its Global Convergence for Solving Symmetric Nonlinear Equations. International Journal of Mathematics and Mathematical Sciences, Article ID 961487, 8 pages.

Babaie-Kafaki, S and Ghanbaric, R. (2012). Two Hybrid Nonlinear Conjugate Gradient Methods Based on a Modified Secant Equation. Journal of Mathematical Programming and Operations Research, 63, 1027-1042.

Li, D H, and Fukushima, M. (2000). A Derivative-Free Line Search and Global Convergence of Broyden-Like Methods for Nonlinear Equations. Optimization Methods and Software, 13, 181-201.

Zhou, W and Shen, D. (2014). An Inexact PRP Conjugate Gradient Method for Symmetric Nonlinear Equations. Numerical Functional Analysis and Optimization, 35(3), 370-388.

Roose, A., Kulla, V., Lomp, M., Meressoo (eds.): (1990). Test Examples of Systems on Nonlinear Equations Version, vol. 3–90. Estonian Software and Computer Service Company, Tallinn.

Dolan, E.D., Mor´e, J.J. (2002). Benchmarking optimizations software with performance profiles. Math. Program. Series A, 91, 201–213.

Mor´e, J.J., Wild, S.M. (2009). Benchmarking derivative-free optimization algorithms. SIAM J. Optim., 20, 1, 172–191.