KASU JOURNAL OF MATHEMATICAL SCIENCES (KJMS) VOL. 1, ISSUE 2, December 2020
ISSN 2734-3839 (Print), ISSN 2735-962X (E-Copy)
http:/www.journal.kasu.edu.ng/index.php/kjms

KASU JOURNAL OF MATHEMATICAL SCIENCES

Improved Computational Performance of a Modified Conjugate Gradient Coefficient for Solving Nonlinear System of Equations

[1*]M.K. Dauda

[1]Department of Mathematical Sciences, Kaduna State University, Nigeria.
*mkdfika@kasu.edu.ng, mkdfika@gmail.com

Abstract
Conjugate gradient (CG) method is an evolution of computational method in solving optimization problems. The aim of this research is to improve the computational performance of an existing conjugate gradient parameter. In this article, an existing conjugate gradient coefficient is considered, and its computational performance is improved using a derivative free line search. The computational performance is based on number of iteration and the CPU time it take to solve a particular problem. Numerical results was implemented using some benchmark problems coded using MATLAB and run on a personal computer 2.4GHz, Intel (R) Core (TM) i7-5500U CPU processor, 4GB RAM memory and on an Acer Aspire with a Windows 7 operating system. The two methods are compared numerically on Microsoft excel 2016. The approach is easy to implement due to its derivative-free nature. It shows that the proposed method is efficient and accurate, thereby appropriate for solving large-scale nonlinear system of equations.

Mathematics Subject Classification: 65H11, 65K05, 65H12, 65H18

Keywords: Conjugate gradient, derivative free, Nonlinear Equations

1. Introduction
Consider a nonlinear system of equations

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

where $F: R^n \to R^n$ is continuously differentiable. There are many method for solving (1), among the most widely used methods are Newton and quasi-Newton methods. This is due to their very attractive convergence properties and practical application, (see for instance, [3,10,11,12]). However, these famous methods do have shortcomings, because at every iteration they require Jacobian matrix or an approximation to Jacobian matrix while solving optimization problems. Thus, they are not usually suitable for large-scale nonlinear systems of equations. This article contribute in proffering solution to these shortcomings. The iterative method used to solve (1) is formed by

$$x_{k+1} = x_k + \alpha_k d_k, k = 0,1,2,\cdots, \quad (2)$$

where $\alpha_k > 0$ is the step-size, x_{k+1} is the iterative point, x_k is the previous iterative point and d_k is the search direction. The search direction is determined using

$$d_k = \begin{cases} -F(x_k) & if\ k = 0 \\ -F(x_k) + \beta_k d_k\ if\ k \geq 0 \end{cases} \quad (3)$$

where $F: R^n \to R^n$ is continuously differentiable, while the scalar parameter β_k is the conjugate gradient coefficient. Improving the efficiency of any parameter in (2) or (3), will automatically improve the computational performance of the entire process. Many researchers worked on improving the conjugate gradient parameter β_k. These includes the following

$\beta_k^{CD} = \frac{g_{k+1}^T g_{k+1}}{g_k^T d_k}$, Conjugate Descent (CD)[6,15], $\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$, Dai and Yuan (DY)[5,15], $\beta_k^{PR} = \frac{g_k^T y_k}{g_k^T g_k}$, Polak and Ribiere (PR) [5,15], $\beta_k^{LS} = \frac{y_k^T g_{k+1}}{g_k^T d_k}$, Liu and Storey (LS)[6,15], $\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$, Fletcher-Reeves (FR)[15], $\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}$, Hestenes-Stiefel (HS)[6,15] and so on, where $g_{k+1} = \nabla f\ (x_{k+1})$. For further reading on β_k methods, refer to [2,3,4,5,6,7,9]. In this research, we considered the method for solving unconstrained optimization problems in [15]. This article gives an improved version of the scheme by introducing a derivative free line search method [8,9]. Recall [12], a nonlinear conjugate gradient method for the unconstrained optimization problem is considered.

$$\min_{x \in R^n} f(x), \tag{4}$$

where the function f is assumed to be continuously differentiable from R^n into R, and the gradient $\nabla f(x_k)$ at point x_k denoted as g_k is available. The nonlinear conjugate gradient method also generates a sequence $\{x_k\}$ by the recursive relation

$$x_{k+1} = x_k + \alpha_k d_k, k = 0,1,2,\cdots, \tag{5}$$

where α_k is the step-length and the search direction d_k is updated by

$$d_k = \begin{cases} -g(x_k) & if\ k = 0 \\ -g(x_k) + \beta_k d_{k-1}\ if\ k \geq 0 \end{cases} \tag{6}$$

where β_k, a scalar as defined above in [12] with

$$f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0}(x_k + \alpha d_k). \tag{7}$$

With the aid of penalty function (8) below, the objective function in problem (1) can be viewed as the first-order optimality condition of the problem (4), where $F(x)$ is the gradient of $f: R^n \to R$.

$$f(x) = \frac{1}{2}\|F(x)\|^2. \tag{8}$$

Now the following section described the proposed method for solving large-scale nonlinear systems of equations (1) and its Algorithm.

While section 3 gives sufficient descent condition. Implementation of proposed method and discussion on Numerical results are presented in section 4. Conclusion was drawn based on the performance profile [13].

2. The Improved Modified Conjugate Gradient Method and its Algorithm (MCGM)

This section presents the process for the scheme. Recall the modified conjugate gradient method for solving nonlinear system of equations [15].

$$\beta_k^{\text{DSHM}} = \frac{F^T(x_k)\left(F(x_k) - \frac{\|F(x_k)\|}{\|F(x_{k-1})\|}F(x_{k-1})\right)}{F^T(x_{k-1})(F(x_k) - d_{k-1})} \tag{9}$$

where $F(x_{k-1}) = \nabla f\ (x_{k-1})$ and $\|.\|$ indicates the Euclidian norm of vectors. As described in [15], the proposed modified Conjugate Gradient coefficient is denoted as β_k^{DSHM}, where the acronym DSHM comes from the names of the researchers and stands for Dauda, Shehu, Hayatu and Mustafa respectively. The main aim of this research is to improve the computational performance of the existing conjugate gradient parameter (9), this is achieved using a derivative free line search [8,9]. This enable the algorithm to function without computing Jacobian matrix. The line search used is given by

$$\alpha_k = max\{s, rs, r^2 s, \cdots\} \tag{10}$$

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 $r, \omega \in (0,1)$.

Remark: It is clear that the line search is well defined.

The nonlinear system of equation (1), generates a sequence $\{x_k\}$ by the recursive relation

$$x_{k+1} = x_k + \alpha_k d_k,\ k = 0,1,2,\cdots,$$

with α_k satisfying (10) and the search direction d_k is updated by (3) with (9) as its conjugate parameter. Thus,

$$d_k = \begin{cases} -F(x_k) & if\ k = 0 \\ -F(x_k) + \beta_k^{\mathrm{DSHM}}\, d_k & if\ k \geq 0 \end{cases} \tag{11}$$

Algorithm of the Scheme

The following is the MCGM algorithm.

Step 1: Given $x_0, \epsilon = 10^{-4}, r = \frac{1}{2}, \omega = \frac{1}{2}$, set $k = 0$ and $d_0 = -g_0$.

Step 2: Compute β_k^{MCGM} based on 9

Step 3: Compute d_k based on (11). If $F_k = 0$, then stop.

Step 4: Compute α_k based on (10).

Step 5: Update the new point based on (2) or (5).

Step 6: Check the stopping criteria. If $\|F(x_k)\| \leq \varepsilon$, then stop. Otherwise go to Step 2 with $k = k + 1$.

3. Convergent Analysis

The Modified Conjugate Gradient coefficient satisfy the convergence properties and the descent condition sufficiently. Thus, the numerical solutions converges.

Sufficient Descent Condition of MCGM Algorithm.

Consider the following theorem with derivative free line search as in (10).

Theorem: Consider the Modified Conjugate Gradient Method with (11), the MCGM is said satisfy the descent condition sufficiently if

$F_k^T d_k \leq -C\|F_k^2\|$ for all $k \geq 0$, and $C \in \Re$.

Proof. If $k = 0$, then clearly we can get $F_o^o(x_o)d_o = -C\|F_0^2\|$ from (11). Hence, condition $F_k^T d_k \leq -C\|F_k^2\|$ holds true, at $k = 0$.
Next, for $k > 0$
Multiply (3) by F_k^T, then
$F_k^T d_k = F_k^T(-F_k + \beta_k^{MCGM} d_{k-1}) = -\|F_k\|^2 + \beta_k^{MCGM} F_k^T d_{k-1}$.
For exact line search, $F_k^T d_{k-1} = 0$.
Thus, $F_k^T d_k = -\|F_k\|^2$,
which implies d_k is sufficient descent direction. Hence, $F_k^T d_k \leq -C\|F_k^2\|$ holds true, thus, the Modified Conjugate Gradient coefficient possess sufficient descent condition. The proof is complete■.

4. Implementation of MCGM and its Numerical Results

This section presents the numerical results of the implementation of the modified conjugate gradient coefficient for solving nonlinear system of equations. The test problems considered in [1,3,5,6,13,15] are used. The Table 1-6 presents the analysis of proposed method denoted as MCGM and the method in [15] denoted as CGM. The methods are compared by solving several benchmark problems with their respective initial points using dimensions ranging from 10 to 10,000.

List of Benchmark Test Problem Used

Problem 1[3]

$$F_i(x) = e^{x_i} + x_i^2 - 1; \;\; i = 1,2,3,\cdots,n.$$
$$i = 1,2,3,\cdots,n.$$

Problem 2[13]

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

Problem 3[1]

$$F_i(x) = x_i(sinx_i cosx_i)^2 + x_i(cosx_i - x_i - 1);$$
$$i = 1,2,3,\cdots,n.$$

Problem 4[13,15]

$$F_i(x) = \; e^{x_i} - 1; \; i \; = \; 1,2,3,\ldots,n.$$
$$i = 1,2,3,\cdots,n.$$

Problem 5[5,6]

$$F_i(x) = x_i^2 - 1;$$
$$i \; = \; 1,2,3,\ldots,n.$$

Problem 6[15]

$$F(x) = e^{x_i^2 - 1} - cos(1 - x_i^2);$$
$$i \; = \; 1,2,3,\ldots,n.$$

The analysis is based on the performances result of each method in terms of number of iteration and CPU time. The meaning of each column in the table are respectively stated "Prob": Benchmark problem; ISP: Initial Starting Point, "Dim": Dimension of the test problems; "Iter": the total number of iterations; "CPU": the CPU time in seconds. A particular method is said to fail if the number of iterations or CPU time exceeds its limit and is denoted by "*". A particular method is said to perform better if it produce less number of iteration and/or CPU time than the other one. Based on Table 1-6, the performance of modified conjugate gradient coefficient with derivative free line search is more efficient than [15]. In overall, the modified conjugate gradient coefficient with derivative free line search managed to solve all given functions with less iteration and time. Thus, with these benchmark problems, is considered as the best approximation method for solving large-scale systems of nonlinear equations. The numerical implementation tests were performed on Microsoft excel 2016, the codes are computed using MATLAB 7.1, R2009b programming environment [14] and run on a personal computer 2.4GHz, Intel (R) Core (TM) i7-5500U CPU processor, 4GB RAM memory and on an Acer Aspire with a Windows 7 operating system. Using the performance profile introduced by Dolan and Mor´e [13], the graphical performance results is presented in Figures 1 and 2, respectively. Let P be the set of benchmark problems and let S be the set of algorithms. We define $t_{p,s}$ to be the number of iterations (or the CPU time in seconds) required to solve the problem $p \in P$ by algorithm $s \in S$. The comparison of each of the two measures is based on the performance ratio given by

$$r_{p;s} =:= t_{p,s}/min\{t_{p,s}\}.$$

Then the performance profile is defined by

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

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, where P is the number of benchmark problems.
The top curve is the method that performs better in a time that was within a factor τ of the best time. Therefore, from Figure 1, the proposed methods MCGM performs better in terms of the number of iteration. Figure 2 gives the performance of the MCGM methods relative to CPU time as compared with CGM.

Table 1: The performance results of problem 1 in each method in terms of number of iteration and CPU time.

			MCGM		**CGM**	
Prob	ISP	Dim	Iter	CPU Time	Iter	CPU Time
1	A	10	2	0.44433	23	0.08413
		50	2	0.07508	17	0.40897
		100	2	0.21035	23	0.32005
		1000	2	0.81157	38	0.55585
		5000	2	2.12648	21	0.81418
		10000	3	16.16561	50	10.27528

Table 2: The performance results of problem 2 in each method in terms of number of iteration and CPU time.

			MCGM		**CGM**	
Prob	ISP	Dim	Iter	CPU Time	Iter	CPU Time
2	B	10	3	0.17667	3	0.17823
		50	3	0.06205	10	0.03633
		100	4	0.1121	10	0.07558
		1000	4	0.81112	10	0.25121
		5000	4	0.67543	10	1.01534
		10000	4	0.57876	12	1.08977

Table 3: The performance results of problem 3 in each method in terms of number of iteration and CPU time.

			MCGM		**CGM**	
Prob	ISP	Dim	Iter	CPU Time	Iter	CPU Time
3	C	10	10	0.19289	4	0.02573
		50	10	0.13698	10	0.05326
		100	17	0.03577	14	0.05442
		1000	20	0.25247	26	0.25592
		5000	20	0.77096	21	0.89268
		10000	20	4.34788	23	6.07403

Table 4: The performance results of problem 4 in each method in terms of number of iteration and CPU time.

			MCGM		**CGM**	
Prob	ISP	Dim	Iter	CPU Time	Iter	CPU Time
4	D	10	9	0.01636	5	0.13752
		50	12	0.19998	1000	18.42693
		100	12	0.04573	*	*
		1000	10	7.11197	*	*
		5000	27	0.93975	*	*
		10000	26	5.53954	10	5.21812

Table 5: The performance results of problem 5 in each method in terms of number of iteration and CPU time.

			MCGM		**CGM**	
Prob	ISP	Dim	Iter	CPU Time	Iter	CPU Time
5	E	10	5	0.20765	11	0.01779
		50	6	0.24014	11	0.05944
		100	6	0.11044	12	0.31941
		1000	6	0.1719	12	0.93839
		5000	7	0.55043	12	3.72989
		10000	10	3.50237	16	5.19109

Table 6: The performance results of problem 6 in each method in terms of number of iteration and CPU time.

			MCGM		**CGM**	
Prob	ISP	Dim	NI	CPU Time	NI	CPU Time
6	F	10	5	0.01148	11	0.01029
		50	8	0.01488	12	0.02641
		100	95	0.04344	12	0.03914
		1000	5	0.2454	13	0.09055
		5000	9	0.67137	14	0.59221
		10000	9	8.02988	8	32.11532

Table 7: Initial starting points.

a	$(0.2,0.2,0.2,\cdots,0.2)^T$
b	$(0.1,0.1,0.1,\cdots,0.1)^T$
c	$(0.9,0.9,0.9,\cdots,0.9)^T$
d	$(0.2,0.2,0.2,\cdots,0.2)^T$
e	$(0.7,0.7,0.7,\cdots,0.7)^T$
f	$(0.4,0.4,0.4,\cdots,0.4)^T$

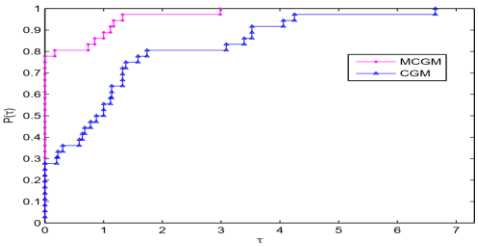

Figure 1: Performance result of MCGM and CGM methods with respect to the number of iterations.

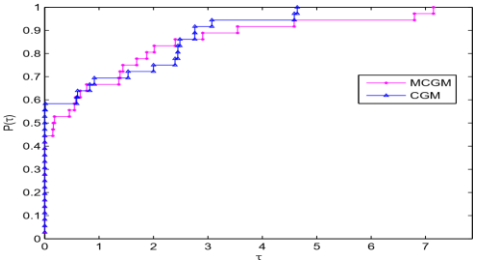

Figure 2: Performance profile of MCGM and CGM methods with respect to the CPU time.

5. Discussion and Conclusion

In this paper, an improved version of a Modified Conjugate Gradient Method is presented. A numerical implementation of the method was carried out using the benchmark problems above with their respective initial points as in Table 7. A particular method is said to perform better if it solve a particular problem with less number of iteration and/or CPU time. In Table 1-6, it is noted that MCGM solved most of the problems with less number of iteration and/or CPU time as compared with its counterpart CGM. This is clearly illustrated if Fiure1 and figure 2.

The method satisfied the convergence properties and the descent condition sufficiently, thus, the numerical solutions converges. The numerical results confirmed that the method is valid in terms of implementation and accurate in terms of output, hence reliable. Thus, an MCGM is recommended for solving system of the form (1). Further research on smooth and non-smooth equations is an area for future research investigation.

Acknowledgement

Author would like to thank the editor and the anonymous referee for their constructive suggestions that improved the quality of this paper greatly.

References

[1] Burden, R. L. and Faires, J. D. (2005). Numerical Solutions of Nonlinear Systems of Equations. Belmount: Thomson Brooks/Cole, 597-640.

[2] 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.

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

[4] M. K. Dauda, Mustafa Mamat, Mohamad Afendee Mohamed, Fatma Susilawati 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)2017, 2599-2610.

[5] Chong, E. K. P. and Zak, S. H. (2005). An Introduction to Optimization (2nd ed). *John Wiley and Sons*, New York.

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

[7] M. Fatemi, (2016). A New Efficient Conjugate Gradient Method for Unconstrained Optimization, *Journal of Computational and Applied Mathematics*.

[8] 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.

[9] 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

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

[11] Zhifeng Dai (2016). Comments on a New Class of Nonlinear Conjugate Gradient Coefficients with Global Convergence Properties, *Applied Mathematics and Computation*, 276, 297-300.

[12] N Hajar, Nur Aini, N Shapiee, Z Z Abidin, W Khadijah, M Rivaie and M Mamat, (2017). A New Modified Conjugate Gradient Coefficient for Solving System of Linear Equations, Journal of Physics, doi :10.1088/1742-6596/890/1/012108.

[13] E. Dolan and J. More, (2002). Benchmarking Optimization Software with Performance Profiles, *Mathematics Program Ser. A*, 91, 201-213.

[14] J. J. Mor´e, B. S. Garbow, and K. E. Hillstrom, (1981). Testing unconstrained optimization software, *ACM Transactions on Mathematical Software*, vol. 7, no. 1, pp. 17–41.

[15] M.K. Dauda, Shehu Usman, Hayatu Ubale and M. Mamat (2019). An Alternative Modified Conjugate Gradient Coefficient for Solving Nonlinear System of Equations. Open Journal of Science and Technology 2(3); 5-8.