Malaysian Journal of Computing and Applied Mathematics 2020, Vol 3(1): 40-46

(Online)

MALAYSIAN JOURNAL OF COMPUTING AND APPLIED MATHEMATICS

Modified Conjugate Gradient Method via DFP Update for Solving Systems of Nonlinear Equations

M K Dauda

Department of Mathematical Sciences, Kaduna State University, Nigeria

**Corresponding author: mkdfika@kasu.edu.ng*

Received: 02/02/2020, Accepted: 24/05/2020
http://dx.doi.org/10.37231/myjcam.2020.3.1.42

Abstract

In this study, a fully derivative-free method for solving large scale nonlinear systems of equations via memoryless DFP update is presented. The new proposed method is an enhanced DFP (Davidon-FletcherPowell) update which is matrix and derivative free thereby require low memory storage. Under suitable conditions, the proposed method converges globally. Numerical comparisons using a set of large-scale test problems showed that the proposed method is efficient.

Keywords: Conjugate Gradient, DFP, Global Convergence, Nonlinear Equations

INTRODUCTION

Consider the the nonlinear system of equation

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

where $F: R^n \to R^n$ is continuously differentiable. Suppose that for each $x \in R^n$, the Jacobian $F'(x)$ of F at x is symmetric. The 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

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

where $F'(x_k)$ is the Jacobian matrix of *F* at x_k (Dauda et al., 2019). The CG methods for solving nonlinear systems of equations generates an iterative points x_k from initial given point x_0 using

$$x_{k+1} = x_k + \alpha_k d_k \quad (3)$$

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

$$d_k = \begin{cases} -F(x_k) & if\ k = 0 \\ -F(x_{k-1}) + \beta_k d_k\ if\ k \geq 1 \end{cases}$$

where β_k is term as conjugate gradient parameter (Zhou & Shen, 2014). Equation (1) is the first-order necessary condition for the unconstrained optimization problem, a saddle point and equality constrained

problem when F is the gradient mapping of some function $g: R^n \rightarrow R, min\, g(x), x \in R^n$. Problem (1) can be converted to the following global optimization problem $ming(x), x \in R^n$ with our function *f* defined by

$$g(x) = \frac{1}{2}\|F(x)\|^2 \quad (4)$$

Among various methods for solving nonlinear equations (1), Newton's method is quite welcome due to its nice properties such as the rapid convergence rate and the global convergence (Liu & Li, 2015; Waziri & Jamilu, 2015; Waziri et al., 2012). However, the Newton method has some shortcomings which includes computation of the Jacobian matrix and solving the Newton system in every iteration (Dauda et al., 2016). Quasi Newton method do not require the repeated evaluation and factorization of the Jacobian matrix. Instead, they find an approximation B_k to the Jacobian matrix which is updated at each iteration by adding a low rank matrix. In the proposed method the approximate Jacobian inverse H_k via DFP is updated with identity matrix in order to improve the efficiency of DFP hence the main aim of this paper. The proposed method is practically efficient when applied on some benchmark problems. In the next section, we present the derivation of the proposed method. In section 3, the numerical results of the proposed method is presented. Section 4, presents the conclusion and future work.

DERIVATION OF THE PROPOSED METHOD: DAVIDONFLETCHER-POWELL (DFP) UPDATE

The DFP update is an iterative method that generates a sequence of $\{x_n\}_{k\geq 0}$ from a given initial guess x_0 via the following

$$x_{k+1} = x_k - \alpha_k B_k^{-1} F(x_k),\ k = 0,1,2,\cdots \quad (5)$$

where $\alpha_k > 0$ is a step length determined by any suitable line search. The proposed method is an enhanced DFP update which is matrix and derivative free as below. Recall the DFP update and approximate the inverse hessian H_k with identity matrix as follows. Using Sherman-Morrison-Woodbury, the formula of the inverse hessian approximation H_k that corresponds to the DFP update is given by

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} - \frac{s_k y_k^T}{y_k^T s_k} \quad (6)$$

H_k is updated at each iteration for $k = 0,1,2,\cdots$ (Liu & Li, 2015). The updated matrix H_{k+1} is chosen in such a way that it satisfies the secant equation

$$s_k = H_{k+1} y_k \text{ with } s_k = x_{k+1} - x_k \text{ and } y_k = F(x_{k+1}) - F(x_k) \quad (7)$$

Letting $H_k \approx I$, where I is an identity matrix, transforms (6) to

$$H_{k+1} = I - \frac{I y_k y_k^T I}{y_k^T I y_k} - \frac{s_k y_k^T}{y_k^T s_k} \quad (8)$$

equation (8) can also be written as

$$H_{k+1} = I - \frac{y_k y_k^T}{y_k^T y_k} - \frac{s_k y_k^T}{y_k^T s_k} \quad (9)$$

To ensure good approximation, premultiply (9) by $g(x_{k+1})$ to obtain

$$H_{k+1} g(x_{k+1}) = g(x_{k+1}) - \frac{y_k y_k^T g(x_{k+1})}{y_k^T y_k} - \frac{s_k y_k^T g(x_{k+1})}{y_k^T s_k} \quad (10)$$

the direction $H_{k+1}g(x_{k+1})$ in (10) can be rewritten as

$$d_{k+1} = -H_{k+1}g(x_{k+1}) \tag{11}$$

Hence from (10) and (11) we have

$$d_{k+1} = -g(x_{k+1}) - \frac{y_k y_k^T g(x_{k+1})}{y_k^T y_k} - \frac{s_k y_k^T g(x_{k+1})}{y_k^T s_k} \tag{12}$$

Now the new direction is obtained via the following

$$d_k = \begin{cases} -g(x_k) & if\ k = 0 \\ -g(x_{k-1}) + y_k\varphi_k - s_k\delta_k\ if\ k \geq 1 \end{cases} \tag{13}$$

where $\varphi_k = \frac{y_k^T g(x_{k+1})}{y_k^T y_k}$ and $\delta_k = \frac{y_k^T g(x_{k+1})}{y_k^T s_k}$

with $s_k = x_{k+1} - x_k$ and $y_k = g(x_{k+1}) - g(x_k)$ (14)

In order to avoid computing Jacobian Matrix, a non-monotone line search by Li and Fukushima is used (Li and Fukushima, (2000) to compute step size α_k. The interesting idea is that it avoid the necessity of descent directions and guarantee that each iteration is well defined. Given $\sigma_1 > 0, \sigma_2 > 0, r \in (0,1)$ be constants and η_k be a given as positive sequence such that $\sum\eta_k < \infty$ and $\alpha_k = Max\{1, r^k\}$ such that is satisfies

$$f(x_k + \alpha_k d_k) - f(x_k) \leq -\sigma_1||\alpha_k F(x_k)||^2 - \sigma_2||\alpha_k d_k||^2 + \eta_k F(x_k) \tag{15}$$

The following is the algorithm and iterative scheme of the proposed update.

Algorithm Of The Proposed Method
Step 1: Given $x_0, \alpha > 0, \sigma\epsilon(0,1)$ and $\epsilon > 0$ compute $d_0 = -g_0$, set $k = 0$.
Step 2: Compute $g(x_k)$ and test the stopping criterion, i.e. $||g(x_k)|| \leq \epsilon$, if yes, then stop, otherwise continue with step 3
Step 3: Compute α_k by using the line search (15)
Step 4: Compute $x_{k+1} = x_k + \alpha_k d_k$
Step 5: Compute search direction using (13)
Step 6: Set $k = k + 1$ and go to step 2.

NUMERICAL RESULTS

In this section, the numerical results of the implementation of the proposed algorithm were reported by solving several benchmark problems listed in the following. The initial points used in the experiments are denoted as *a, b, c* and *d* as shown in the table 1. The following are benchmark test problems used in the experiments:

Table 1. Initial points used in the experiments

a	$(1,1,1,\ldots,1)^T$
b	$(0.5,0.5,0.5,\ldots,0.5)^T$
c	$(0.5,0.5,0.5,\ldots,0.5)^T$
d	$(1,1,1,\ldots,1)^T$

Problem 1: (Nocedal and Wright, 2006)

$$F_i(x) = \begin{pmatrix} 2 & -1 & \cdots \\ 0 & 2 & -1 \\ \ddots & \ddots & \ddots \\ \ddots & \ddots & -1 \\ \cdots & -1 & 2 \end{pmatrix} x_i + (e_1^x - 1, e_2^x - 1, \cdots, e_n^x - 1)^T; \ \ i = 1,2,3, \cdots, n. \quad (16)$$

Problem 2: (Dauda et al., 2016)

$$F(1) = x_1 - e^{cos(\frac{x_1 + x_2}{n+1})};$$
$$F_i(x) = x_i - e^{cos(\frac{x_i + x_{i+1}}{n+1})};$$
$$F_i(n) = x_n - e^{cos(\frac{x_n + x_{n+1}}{n+1})};$$
$$i = 1,2,3, \cdots, n. \quad (17)$$

Problem 3: (Nocedal and Wright, 2006)

$$F_i(x) = \begin{pmatrix} 2 & -1 & \cdots \\ 0 & 2 & -1 \\ \ddots & \ddots & \ddots \\ \ddots & \ddots & -1 \\ \cdots & -1 & 2 \end{pmatrix} x_i + (sinx_1 - 1, sinx_2 - 1, \cdots, sinx_n - 1)^T; \ \ i = 1,2,3, \cdots, n. \quad (18)$$

Problem 4: (Dauda et al., 2016)

$$F_i = ((sin(x_i) \times cos(x_i))^2) \times x_i - (cos(x_i) - x_i - 1) \times x_i \quad (19)$$
$$i = 1,2,3, \cdots, n.$$

The proposed algorithm is denoted as *DFDFP* and compare its performance with the method for solving symmetric nonlinear equations in Dauda et al. (2016) denoted as *DFCG*. The above test problems with different given initial points are considered (see table 1), each problem has been tested using both methods with different values of n, (see table 2-3), where n is the number of variables of each problem. The comparison of the performance between the methods using the benchmarks problems above was based on the performance profile presented by Dolan & More (2002). The computational experiment is based on number of iterations, CPU time and residual norm of $F(x_k)$. 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 the methods was implemented with the parameters $\alpha_1 = 0.01, r = 0.2, \sigma_1 = \sigma_2 = 10^{-4}$, and $\eta_k = \frac{1}{\{k+1\}^2}$.The search is stopped if: (i) $\| F(x_k) \| < \epsilon$ with $\epsilon < 10^{-4}$ or (ii) The total number of iteration exceeds 1000. The numerical results of the comparison between the proposed method and the result in Dauda et al. (2016) is presented in table 2-3. The meaning of each acronym in the tables stands for a meaning as follows:"*ISP*" : Initial Starting Points, "*Dim*" : Dimension of the test problems, "Iter:" Total Number of Iterations, "Time:" CPU Time in Seconds, "*NF* :" Norm of $F(x_k)$ while " $-$ ": indicates a failure at a point due to stopping criterion stated above. In the particular problem *i*, the *DFDFP* performs better than the performance of *DFCG* if the number of iteration (*Iter*) or the CPU time in seconds (*Time*) of *DFDFP* is less than the number of iteration or the CPU time corresponding to the *DFCG* method respectively. By the given results, the proposed method performs effectively compared to *DFCG* with the given benchmark problems as in the following table.

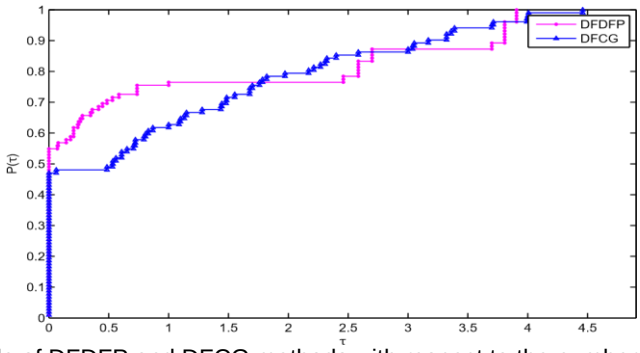

Figure 1. Performance profile of DFDFP and DFCG methods with respect to the number of iteration for the problems.

Table 2. Numerical results based on dimension of problem (n), Number of iterations, Norm and CPU Time (in seconds) of problems

			DFDFP			DFCG		
	ISP	**Dim**	**iter**	**NF**	**Time**	**iter**	**NF**	**Time**
Problem 1	a	10	16	9.71E-04	0.281856	34	8.54E-04	0.047385
		100	17	9.48E-04	0.023904	28	8.53E-04	0.051521
		500	18	7.93E-04	0.087409	236	6.87E-04	1.80593
		1000	17	4.96E-04	0.324446	141	7.36E-04	3.656791
		5000	16	5.88E-04	4.82253	16	8.95E-04	0.052205
		10000	17	5.73E-04	19.54542	178	8.68E-04	350.6963
Problem 2	b	10	16	4.97E-04	0.016667	28	8.21E-04	0.038541
		100	16	6.45E-04	0.020065	32	9.51E-04	0.058559
		500	15	5.81E-04	0.067696	48	8.01E-04	0.366705
		1000	15	7.60E-04	0.222826	44	8.51E-04	1.215524
		5000	17	6.84E-04	4.835957	46	8.38E-04	21.2679
		10000	17	9.38E-04	19.7239	60	6.95E-04	111.1503
Problem 3	c	10	15	9.79E-04	0.025416	26	8.25E-04	0.036343
		100	19	5.25E-04	0.045846	28	9.98E-04	0.048133
		500	18	8.44E-04	0.082166	32	7.51E-04	0.227111
		1000	21	7.77E-04	0.326295	32	5.63E-04	0.780208
		5000	19	7.52E-04	5.456086	31	7.02E-04	14.64706
		10000	20	8.05E-04	24.27089	33	8.54E-04	62.52103
Problem 4	d	10	18	7.67E-04	0.017997	26	7.42E-04	0.038476
		100	21	7.86E-04	0.028005	33	8.43E-04	0.056056
		500	21	8.26E-04	0.096645	32	9.13E-04	0.216819
		1000	23	6.79E-04	0.346432	42	8.05E-04	1.055155
		5000	23	8.36E-04	6.691444	51	8.36E-04	25.30989
		10000	29	8.22E-04	37.05071	42	9.90E-04	81.37287

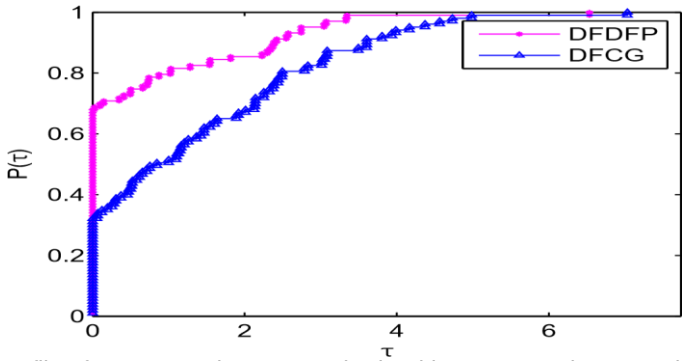

Figure 2. Performance profile of DFDFP and DFCG methods with respect to the CPU Time for the problems 1-4.

Figure 1-2 presents the graphical results of the problems 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, we note that the performance of *DFDFP* methods relative to the number of iteration, outperforms *DFCG*. It is clear that *DFDFP* method is the fastest and the top performer. Similarly, Figure 2 shows the performance of *DFDFP* and *DFCG* methods relative to CPU Time, the *DFDFP* is the top performer, showing that the proposed method is faster and achieved better results, thus we conclude that *DFDFP* outperformed *DFCG*.

CONCLUSION AND FUTURE WORK

In this paper, a method for solving nonlinear systems of equations (1) via modification of Davidon-Fletcher-Powell (DFP) update is presented. The proposed method is particularly suitable for large-scale equations because of low memory requirement. We have compared the *DFDFP* method with *DFCG* by Dauda et al. (2016) and found that the proposed method is effective in practical computation and superior to *DFCG* in many situations and the preliminary numerical results show that our method is substantial and efficient. Thus we conclude that the proposed method is an alternative option for solving large scale system of non-linear equations (1). To extend this work, one could establish the global Convergence of the proposed Algorithm.

References

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

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.

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

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.

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.

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

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.

Waziri M Y, Leong W, and Mamat M. (2012). A two-step matrix-free secant method for solving large-scale systems of nonlinear equations. Journal of Applied Mathematics, Article ID 348654, 6 pages.

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.