Derivative free Davidon-Fletcher-Powell (DFP) for solving symmetric systems of nonlinear equations

M Mamat1*, M K Dauda1,3, M A bin Mohamed1, M Y Waziri2, F S Mohamad1, H Abdullah4

1Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Gong Badak, Terengganu, Malaysia.
2Department of Mathematical Science, Bayero University, Kano, Nigeria.
3Department of Mathematics and Statistics, Kaduna Polytechnic, Kaduna, Nigeria.
4Department of Mathematics and Computer Science, Sule Lamido University, Nigeria.

*Corresponding author: musmat567@gmail.com

Abstract: Research from the work of engineers, economist, modelling, industry, computing, and scientist are mostly nonlinear equations in nature. Numerical solution to such systems is widely applied in those areas of mathematics. Over the years, there has been significant theoretical study to develop methods for solving such systems, despite these efforts, unfortunately the methods developed do have deficiency. In a contribution to solve systems of the form $F(x) = 0, x \in \mathbb{R}^n$, a derivative free method via the classical Davidon-Fletcher-Powell (DFP) update is presented. This is achieved by simply approximating the inverse Hessian matrix with $Q_k^{-1}$ to $Q_{k+1}^{-1}$. The modified method satisfied the descent condition and possess local superlinear convergence properties. Interestingly, without computing any derivative, the proposed method never fail to converge throughout the numerical experiments. The output is based on number of iterations and CPU time, different initial starting points were used on a solve 40 benchmark test problems. With the aid of the squared norm merit function and derivative-free line search technique, the approach yield a method of solving symmetric systems of nonlinear equations that is capable of significantly reducing the CPU time and number of iteration, as compared to its counterparts. A comparison between the proposed method and classical DFP update were made and found that the proposed method is the top performer and outperformed the existing method in almost all the cases. In terms of number of iterations, out of the 40 problems solved, the proposed method solved 38 successfully, (95%) while classical DFP solved 2 problems (i.e. 05%). In terms of CPU time, the proposed method solved 29 out of the 40 problems given, (i.e.72.5%) successfully whereas classical DFP solves 11 (27.5%). The method is valid in terms of derivation, reliable in terms of number of iterations and accurate in terms of CPU time. Thus, suitable and achieved the objective.

1. Introduction
Consider the system of nonlinear equations

$$F(x) = 0, x \in \mathbb{R}^n$$

where the vectors $F : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear mapping assumed to satisfy the following assumptions:

(i) There exists an $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$ (ii) $F$ is a continuously differentiable mapping in a neighborhood of $x^*$ of the system and (iii) The Jacobian matrix of $F$ at $x$ given by $J(x) = F'(x)$ is symmetric. Among the many iterative methods for solving (1), Newton’s method is the prominent one [1,2,11], it generates a sequence of iterates $x_k$ from a given initial point $x_0$ via
where

\[ F'(x_k) \] is the Jacobian matrix of \( F \) at \( x_k \).

The Newton method has obvious shortcomings. Computing Jacobian matrix sometimes is difficult, in some cases is even impossible [9,15], in order to alleviate this deficiency, a derivative free method has been devised. This is achieved by modifying the classical Davidon-Fletcher-Powell (DFP). In the proposed method, the \( H_k \) of DFP is approximated with \( \theta_k I \) where \( I \) is the Identity matrix and \( \theta_k \) the acceleration parameter. Throughout this article, we always assume that the problem (1) is symmetric and can be converted to an equivalent global optimization problem [8,10].

The Davidon-Fletcher-Powell (DFP) Update is an iterative method that generates a sequence of \( \{x_n\}_{n=0}^{\infty} \) which converges to a local maximum of \( f(x) \).

\[ x_{k+1} = x_k - (F'(x_k))^{-1}F(x_k), \quad k = 0,1,2,\ldots \]  

where \( F'(x_k) \) is the Jacobian matrix of \( F \) at \( x_k \).

Conjugate Gradient methods for solving nonlinear systems of equations generates an iterative points \( x_k \) from initial given point \( x_0 \) using

\[ x_{k+1} = x_k + \alpha_k d_k \]  

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

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

where \( \beta_k \) is term as conjugate gradient parameter[514,18]. In the next section, the derivation of the proposed method is presented. In Section 3, the numerical results is established while section 4 presents the conclusion and future work.

2. Derivative free Davidon-Fletcher-Powell (DFDFP) Update

The Davidon-Fletcher-Powell (DFP) Update is an iterative method that generates a sequence of \( \{x_n\}_{n=0}^{\infty} \) from a given initial guess \( x_0 \) via the following

\[ x_{k+1} = x_k - \alpha_k B_k^{-1}F(x_k), \quad k = 0,1,2,\ldots \]  

where \( \alpha_k > 0 \) is a step length determined by any suitable line search [6,7]. Recall, from DFP formula and using Sherman-Morrison-Woodbury, the inverse hessian approximation \( H_k \) that corresponds to the DFP update is given by

\[ H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{y_k y_k^T}{y_k^T s_k} \]  

\[ \text{where } H_k \text{ is updated at each iteration for } k = 0,1,2,\ldots . \text{ The updated matrix } H_{k+1} \text{ is chosen in such a way that it satisfies the secant equation below} \]

\[ s_k = H_{k+1} y_k \]  

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

To improve the efficiency of the classical DFP, the inverse hessian \( H_k \) is approximated with \( \theta_k I \), where \( \theta_k = \frac{y_k^T y_k}{y_k^T s_k} \) and \( I \) is the identity matrix. This makes it derivative free and transforms (8) to

\[ H_{k+1} = \theta_k I - \frac{\theta_k y_k y_k^T \theta_k I}{y_k^T \theta_k y_k} + \frac{s_k y_k^T}{y_k^T s_k} \]  

\[ \text{equation (10) can also be written as} \]

\[ H_{k+1} = \theta_k I - \frac{\theta_k y_k y_k^T \theta_k I}{y_k^T \theta_k y_k} + \frac{s_k y_k^T}{y_k^T s_k} \]  

\[ \text{To ensure good approximation, pre-multiply (11) by } g(x_{k+1}) \text{to obtain} \]

\[ H_{k+1} g(x_{k+1}) = \theta_k g(x_{k+1}) - \frac{\theta_k y_k y_k^T \theta_k g(x_{k+1})}{y_k^T \theta_k y_k} + \frac{s_k y_k^T g(x_{k+1})}{y_k^T s_k} \]  

(12)

the direction \( H_{k+1} g(x_{k+1}) \) in (12) can be rewritten as
Then the performance profile is defined by

$$d_{k+1} = -H_{k+1}g(x_{k+1})$$  \hspace{1cm} (13)$$

Hence from (12) and (13) we have

$$d_{k+1} = -\theta_k g(x_{k+1}) + \frac{\theta_k y_{k} y_{k}^T g(x_{k+1})}{y_{k}^T g(x_{k+1})} - \frac{s_k y_{k}^T g(x_{k+1})}{y_{k}^T s_k}$$  \hspace{1cm} (14)$$

To further ensure derivative free feature of the proposed method, the following equation is used as in [3].

$$g(x_k) \approx \nabla f(x_k) = \frac{f(x_k + \alpha_k F(x_k)) - f(x_k)}{\alpha_k}$$  \hspace{1cm} (15)$$

Now the new direction is obtained via the following

$$d_k = \begin{cases} 
-\theta_k g(x_{k+1}) & \text{if } k = 0 \\
-\frac{F(x_k)}{\alpha_k} & \text{if } k \geq 1
\end{cases}$$  \hspace{1cm} (16)$$

where

$$\theta_k = \frac{y_{k} y_{k}^T}{y_{k}^T s_k}, s_k = x_{k+1} - x_k \text{ and } y_k = F(x_{k+1}) - F(x_k)$$

To avoid computing Jacobian Matrix, a non-monotone and derivative free line search from the work of Li and Fukushima [3] is used to compute step size $\alpha_k$. The line search avoid the necessity of descent directions and guarantee that each iteration is well defined.

The idea is that, the parameter are considered to be $\sigma_1 > 0, \sigma_2 > 0, \sum \eta > 0, r \in (0,1)$ be constants and $\eta_k$ be a given positive sequence such that $\eta_k < \infty$, let $\alpha_k = \text{Max}\{1, r^k\}$ that satisfy

$$f(x_k + \alpha_k d_k) - f(x_k) \leq -\sigma_1 ||\alpha_k F(x_k)||^2 - \sigma_2 ||\alpha_k d_k||^2 + \eta_k F(x_k)$$  \hspace{1cm} (17)$$

Finally, using the above procedure, the following is the algorithm for the new update.

**Algorithm: Derivative free Davidon-Fletcher-Powell (DFDFP) Update.**

**Step 1:** Given $x_0, \alpha > 0, \sigma \in (0,1)$ and $\epsilon > 0$ compute $d_0 = -F(x_0)$, set $k = 0$.

**Step 2:** Compute $F(x_k)$ and test the stopping criterion, i.e. $||F(x_k)|| \leq \epsilon$, if yes, then stop, otherwise continue with next step.

**Step 3:** Compute $\alpha_k$ by using the line search using (17).

**Step 4:** Compute $\theta_k = \frac{y_{k} y_{k}^T}{y_{k}^T s_k}$.

**Step 5:** Compute $x_{k+1} = x_k + \alpha_k d_k$

**Step 6:** Compute search direction using (16).

**Step 7:** Set $k = k + 1$ and go to step 2.

### 3. Numerical Results

This section gives the numerical results of the proposed method. The computational experiment is based on number of iterations and CPU time. The proposed method is denoted as DFDFP, and its performance is compared with the Classical DFP [7,16], this is achieved by solving 40 benchmark problems with their respective initial points using five (5) different dimensions ranging from 10 to 5000. The comparison of the performance between the methods using the benchmarks problems in [12,13], was based on the performance profile presented by Dolan and More [4]. The performance profile $P : R \rightarrow [0,1]$ is defined as follows: Let $P$ and $S$ be the set of problems and set of solvers respectively. For $n_s$ solvers and $n_p$ problems, and for each problem $p \in P$ and for each solver $s \in S$, we define $t_{p,s}$: = (number of iterations required to solve problem $p$ by solver $s$). The performance ratio is given by

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

Then the performance profile is defined by
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. 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 same parameters as $\alpha_1 = 0.01, r = 0.2, \sigma_1 = \sigma_2 = 10^{-4}$, 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 [7,16] are presented in table 1 and 2. The meaning of each column in the tables are stated as follows, “P” : stands for Benchmark problem, “ISP” : stands for initial starting points, “n” : stands for dimension of the test problems, “Iter” : the total number of iterations and “CPU” : the CPU time in seconds. In particular problem i, the DFDFP performs better if the number of iteration (iter) or the CPU time in seconds (Time) is less than the number of iteration or the CPU time corresponding to the other methods respectively.

### Table 1. Numerical results of problem 1-5 based on iter and CPU

| Prob | ISP | DFDFP | Classical DFP |
|------|-----|-------|---------------|
|      |     | Dim   | Iter | CPU     | Iter | CPU     |
| 1    | (0.5,0.5,...,0.5)^T |       | 10   | 7      | 0.001142 | 8 | 0.017238 |
|      |     |       | 100  | 7      | 0.001654 | 10 | 0.001905 |
|      |     |       | 500  | 7      | 0.002712 | 11 | 0.002654 |
|      |     |       | 1000 | 7      | 0.002367 | 11 | 0.003932 |
|      |     |       | 10000| 8      | 0.030609 | 12 | 0.030811 |
| 2    | (0.5,0.5,...,0.5)^T |       | 10   | 7      | 0.002469 | 10 | 0.018131 |
|      |     |       | 100  | 7      | 0.001869 | 12 | 0.001844 |
|      |     |       | 500  | 8      | 0.003267 | 13 | 0.004150 |
|      |     |       | 1000 | 8      | 0.005055 | 13 | 0.005513 |
|      |     |       | 10000| 8      | 0.036614 | 14 | 0.038679 |
| 3    | (0.5,0.5,...,0.5)^T |       | 10   | 15     | 0.035759 | 15 | 0.002879 |
|      |     |       | 100  | 169    | 0.042314 | 2 | 0.000905 |
|      |     |       | 500  | 14     | 0.009544 | 36 | 0.012241 |
|      |     |       | 1000 | 14     | 0.016358 | 38 | 0.025325 |
|      |     |       | 10000| 22     | 0.178635 | 43 | 0.182390 |
| 4    | (0.5,0.5,...,0.5)^T |       | 10   | 5      | 0.014770 | 9 | 0.002178 |
|      |     |       | 100  | 6      | 0.001694 | 11 | 0.001897 |
|      |     |       | 500  | 6      | 0.002481 | 11 | 0.002557 |
|      |     |       | 1000 | 6      | 0.003517 | 12 | 0.004300 |
|      |     |       | 10000| 6      | 0.026904 | 13 | 0.032316 |
Table 2. Numerical results of problem 5-8 based on iter and CPU

| Prob | ISP | Dim | Iter | CPU   | Iter | CPU   |
|------|-----|-----|------|-------|------|-------|
| 5    | $(0.5,0.5,\ldots,0.5)^T$ | 10   | 10   | 0.006476 | 11   | 0.001915 |
|      |     | 100  | 12   | 0.002042 | 13   | 0.002043 |
|      |     | 500  | 13   | 0.003809 | 16   | 0.002710 |
|      |     | 1000 | 14   | 0.005711 | 17   | 0.004552 |
|      |     | 10000| 16   | 0.037686 | 26   | 0.052033 |
| 6    | $(0.5,0.5,\ldots,0.5)^T$ | 10   | 8    | 0.013002 | 16   | 0.003420 |
|      |     | 100  | 8    | 0.001672 | 18   | 0.003797 |
|      |     | 500  | 8    | 0.002510 | 19   | 0.004238 |
|      |     | 1000 | 8    | 0.003307 | 19   | 0.008462 |
|      |     | 10000| 9    | 0.024195 | 20   | 0.059023 |
| 7    | $(0.5,0.5,\ldots,0.5)^T$ | 10   | 3    | 0.011541 | 9    | 0.001636 |
|      |     | 100  | 4    | 0.001315 | 10   | 0.001493 |
|      |     | 500  | 4    | 0.001569 | 11   | 0.002061 |
|      |     | 1000 | 4    | 0.002279 | 11   | 0.002934 |
|      |     | 10000| 4    | 0.021024 | 13   | 0.021863 |
| 8    | $(0.5,0.5,\ldots,0.5)^T$ | 10   | 7    | 0.001941 | 11   | 0.013392 |
|      |     | 100  | 7    | 0.001571 | 12   | 0.001634 |
|      |     | 500  | 7    | 0.002544 | 13   | 0.001914 |
|      |     | 1000 | 8    | 0.002835 | 13   | 0.003498 |
|      |     | 10000| 8    | 0.020552 | 15   | 0.021050 |

From the above tables, the proposed methods solves 38 problems effectively scoring 95% in terms of number of iterations, while the classical DFP scored 2% only. In terms of CPU times, the proposed method outperformed the classical DFP by scoring 72.5% of the solutions, i.e.29 out of 40 problems was solved effectively by the proposed method, while 27.5% is solved by the classical DFP.

4. Conclusion and Future work
In this article, a derivative free method for solving symmetric nonlinear systems of equations is given. The method achieved its objectives fully, i.e. it is derivative free, effective in terms of number of iterations and faster in terms of CPU time. Thus, DFDFP is a very good alternative for solving symmetric nonlinear systems of equations. For further research work, the DFDFP can be extended to non-smooth nonlinear systems of equations. Also, it can be applied to solve a real life problems.

Acknowledgments
The authors would like to thank the administration of Universiti Sultan ZainalAbidin (UniSZA) for funding this research partially under the fundamental research grant Scheme FRGS/1/2015/ICT03/UniSZA/02/1.

References
[1] Burden, R.L., Faires, J.D 2005 Numerical Analysis Numerical Solutions of Nonlinear Systems of Equations, 597-640.
[2] Chong, E.K.P. and Zak, S.H. 2005) An introduction to optimization (2nd ed) John Wiley and Sons, New York.
[3] 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.

[4] E. Dolan and J. More 2002 Bench-marking optimization software with performance profiles, *Math. Program. Ser. A* 91, 201-213.

[5] Gonglin Yuana and Maojun Zhang 2015 A three-terms Polak-Ribire-Polyak conjugate gradient algorithm for large-scale nonlinear equations, *Journal of Computational and Applied Mathematics* 286, 186-195.

[6] Griva, I., Nash, S.G. and Sofer, A. 2009 Linear and nonlinear optimization (2nd ed). *Society for Industrial and Applied Maths. Philadelphia.*

[7] J. Nocedal and S.J. Wright 2006 Numerical Optimization, 2nd ed., *Springer, New York.*

[8] 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.

[9] M. Y. Waziri, H. A. Aisha and Mustafa Mamat 2014 A Newton’s-Like Method with Extra Updating Strategy for Solving Singular Fuzzy Nonlinear Equations *Applied Mathematical Sciences* 142, 7047-7057.

[10] M. Fatemi 2016 A new efficient conjugate gradient method for unconstrained optimization, *Journal of Computational and Applied Mathematics.*

[11] Mohd Rivaie, Mustafa Mamat and Abdelrahman A. Bashar 2015 A new class of nonlinear conjugate gradient coefficients with exact and inexact line searches, *Applied Mathematics and Computation* 268, 1152-1163.

[12] Mustafa Mamat, M. K. Dauda, M. Y. Waziri, Fadhila Ahmad, and Fatma Susilawati 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.

[13] M. K. Dauda, Mustafa Mamat, M. Y. Waziri, Fadhila Ahmad 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, Issue 11, 1787-1804.

[14] Neculai Andrei 2013 A simple three-term conjugate gradient algorithm for unconstrained Optimization *Journal of Computational and Applied Mathematics* 241, 19-29.

[15] 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.

[16] Y. Ding, E. Lushi and Q. Li, Investigation of Quasi-Newton Methods for Unconstrained Optimization, Simon Fraser University, Canada.

[17] Y.U. Gaohang 2010 Derivative-Free for Method Solving Large-Scale Nonlinear Systems of Equations, *Journal of Management Optimization* 6, 149-160.

[18] Yasushi Narushima and Hiroshi Yabe 2012 Conjugate Gradient Methods Based on Secant Conditions that Generate Descent Search Directions for Unconstrained Optimization, Journal of Computational and Applied Mathematics, 236, 4303-4317.

[19] Zhifeng Dai 2016 Comments on a new class of nonlinear conjugate gradient coefficients with global convergence properties, *Applied Mathematics and Computation* 276, 297-300.