Journal of Numerical Mathematics and Stochastics, **9** (1) : 20-32, 2017 © ***JNM@S***
http://www.jnmas.org/jnmas9-2.pdf Euclidean Press, LLC
Online: ISSN 2151-2302

A Transformed Double Step Length Method for Solving Large-Scale Systems of Nonlinear Equations

A. S. HALILU[1], **and M. Y. WAZIRI**[2]

[1]Department of Mathematics and Computer Sciences, Sule Lamido University, Kafin Hausa, Jigawa, Nigeria;[2]Department of Mathematical Sciences, Faculty of Science, Bayero University Kano, Kano, Nigeria, E-mail: abubakarsaddiqu@gmail.com

Abstract. *In this paper, we present a double direction and step length method for solving large-scale systems of nonlinear equations, that is based on approximating the Jacobian with a diagonal matrix by means of an acceleration parameter. This method is a new approach that reduces the two step length into a single step length and employs a derivative-free line search in order to obtain a suitable step length. Furthermore, this method is matrix-free, and so is advantageous when solving large-scale nonlinear systems of equations. Under appropriate conditions, we show that the proposed method is globally convergent. The preliminary numerical results, reported in this paper, show that the method is practically quite effective.*

Key words : Double Step Length, Double Direction, Global Convergence, Acceleration Parameter, Systems of Nonlinear Equations.

AMS Subject Classifications : 65H11, 65K05, 65H12, 65H18

1. Introduction

This work deals with the system of nonlinear equations:

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

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is nonlinear map.

A renowned method for finding a numerical solution to (1) is the Newton's method. The method is simple to implement, and generates an iterative sequences $\{x_k\}$ from a given initial guess x_0 in a neighborhood of the solution x^* via

$$x_{k+1} = x_k - (F'(x_k))^{-1}F(x_k), \tag{2}$$

where $k = 0, 1, 2, \ldots$ and $(F'(x_k)$ is the Jacobian matrix.

The attractive features of this method are easy implementation and rapid convergence [4]. However, Newton's method requires the computation of Jacobian matrix, which invokes the first-order derivative of the system. It is well known that the computation of some function derivatives are costly in practice, sometimes they are not even available or could not be obtained exactly. In this case Newton's method cannot be directly applied [3,15,16]. To overcome such a difficulty, simple modifications of the Newton method have been introduced, like the fixed Newton method [19] for the determination of solution x^*, which is given by

$$x_{k+1} = x_k - (F'(x_0))^{-1}F(x_k), \quad (3)$$

where $k = 0, 1, 2, \ldots$ A method that avoids the computation and storing the Jacobian in each iteration (except at $k = 0$), but it still requires solving the system of n linear equations which consumes increased CPU time as the system's dimension increases [19].
A quasi-Newton's method is another variant of Newton-type methods that replaces the Jacobian, or its inverse, with an approximation which can be updated at each iteration [18]. Its updating scheme is given by

$$x_{k+1} = x_k - B_k^{-1}F(x_k), \quad (4)$$

where B_k is the approximation of the Jacobian at x_k. The rationale behind a quasi- Newton method is to do with the evaluation cost of the Jacobian matrix [1,18].

It is vital to mention that due to the well known shortcomings of Newton's method, a double step length has been proposed in [1] and its pertaining iterative procedure is given by

$$x_{k+1} = x_k + \alpha_k d_k + \beta_k c_k, \quad (5)$$

where x_{k+1} represents a new iteration point, x_k is the previous iterative point, α_k and β_k denote the step lengths, while b_k and c_k are search directions respectively.

A transformation of double step length methods has originally been used in unconstrained optimization problems. They are particularly efficient due to their convergence properties, simple implementation, and low storage requirements [11]. Nevertheless, the study of transformation of double step length methods for solving systems of nonlinear equations is very scanty. For this reason we are motivated to write this paper.

We are interested in approximating the Jacobian with a diagonal matrix via

$$F'(x_k) \approx \gamma_k I, \quad (6)$$

where I is an identity matrix.

Furthermore (1) can emerge from an unconstrained optimization problem, a saddle point, and equality constrained problem [6]. Let f be a norm function defined by

$$f(x) = \tfrac{1}{2}\|F(x)\|^2. \quad (7)$$

The nonlinear equations problem (1) is equivalent to the following global optimization problem

$$\min f(x), \quad x \in \mathbb{R}^n. \quad (8)$$

The double direction method has been proposed in [2]. Using multi-step iterative information and curve search to generate new iteration points. However, a multi-step algorithm for minimization of a nondifferentiable function is presented in [2]. Moreover, a double direction method for solving unconstrained optimization problem is presented in [8]. Recently, a double step size method for solving unconstrained optimization problems is proposed in [9].

There are several procedures for the choice of the search direction [7, 8, 10, 13, 14, 2, 11] mentioned above. The step length α_k can also be computed either exact or inexact. It is very expensive to find the exact step length in a practical computation. Therefore the most frequently used line search in practice is inexact line search [18, 8, 9, 12, 17, 20]. A basic

requirement of the line search is to sufficiently decrease the function values i.e. to establish $\|F(x_{k+1})\| \leq \|F(x_k)\|$.

We have organized the paper as follows. In the next section, we present the proposed method. Convergence results are presented in section 3. Some numerical experiments and results are reported in section 4. Finally, the paper concludes in section 5.

2. Main Result

In this section we intend to reduce the two step lengths α_k and β_k in (5) into a single step length. This reduction is made possible by the additional assumption:

$$\beta_k = \tfrac{1}{2}\alpha_k. \tag{9}$$

In order to incorporate more information on the iterates at each iteration and to improve on the direction towards the solution, we suggest new directions b_k and c_k in (5) to be defined as:

$$d_k = -\gamma_k^{-1}F(x_k), \tag{10}$$

where $\gamma_k > 0$ is an acceleration parameter, and

$$c_k = -F(x_k). \tag{11}$$

So by putting (9),(10) and (11) into (5) we have the general scheme as:

$$x_{k+1} = x_k + (\alpha_k + \tfrac{1}{2}\alpha_k\gamma_k)d_k. \tag{12}$$

We proceed to obtain the acceleration parameter by using Taylor's expansion of the first order that leads to the following approximation:

$$F(x_{k+1}) \approx F(x_k) + F'(\xi)(x_{k+1} - x_k), \tag{13}$$

where the parameter ξ fulfills the conditions $\xi \in [x_k, x_{k+1}]$,

$$\xi = x_k + \delta(x_{k+1} - x_k), \quad 0 \leq \delta \leq 1. \tag{14}$$

Bearing in mind that the distance between x_k and x_{k+1} is small enough, we can take $\delta = 1$ in (14) and assume $\xi = x_{k+1}$, so as

$$F'(\xi) \approx \gamma_{k+1}I. \tag{15}$$

Now from (13) and (15) it is not difficult to verify that:

$$F(x_{k+1}) - F(x_k) = \gamma_{k+1}(x_{k+1} - x_k). \tag{16}$$

Taking $y_k = F(x_{k+1}) - F(x_k)$ and $s_k = x_{k+1} - x_k$, we have

$$y_k = \gamma_{k+1}s_k. \tag{17}$$

By multiplying both side of (17) with y_k^T, the acceleration parameter yields:

$$\gamma_{k+1} = \frac{y_k^T y_k}{y_k^T s_k}. \tag{18}$$

We then use the derivative-free line search proposed in [6] in order to compute our step length α_k. Let $\omega_1 > 0$, $\omega_2 > 0$ and $r \in (0,1)$ be constants and let η_k be a given positive sequence such that

$$\sum_{k=0}^{\infty} \eta_k < \eta < \infty, \tag{19}$$

and

$$f(x_k + (\alpha_k + \tfrac{1}{2}\alpha_k\gamma_k)d_k) - f(x_k) \leq -\omega_1\|\alpha_k F(x_k)\|^2 - \omega_2\|\alpha_k d_k\|^2 + \eta_k f(x_k). \tag{20}$$

Finally, let i_k be the smallest non negative integer i such that (45) holds for $\alpha = r^i$, and let $\alpha_k = r^{i_k}$. Now we can list the algorithm of the proposed method.

Algorithm 1(TDS)
STEP 1: Given x_0, $\gamma_0 = 0.01$, $\epsilon = 10^{-4}$, set $k = 0$.
STEP 2: Compute $F(x_k)$.
STEP 3: If$\|F(x_k)\| \leq \epsilon$ then stop, else go to STEP 4.
STEP 4: Compute search direction $d_k = -\gamma_k^{-1}F(x_k)$.
STEP 5: Compute step length α_k(using (20)).
STEP 6: Set $x_{k+1} = x_k + (\alpha_k + \frac{1}{2}\alpha_k\gamma_k)d_k$.
STEP 7: Compute $F(x_{k+1})$.
STEP 8: Determine $\gamma_{k+1} = \frac{y_k^T y_k}{y_k^T s_k}$.
STEP 9: Set $k = k + 1$, and go to STEP 3.

3. Convergence Analysis

This section is devoted to a study of the global convergence of our method (TDS). To begin with, let us define the level set

$$\Omega = \{x|\|F(x)\| \leq \|F(x_0)\|\}. \quad (21)$$

In order to analyze the convergence of algorithm 1, we state the following assumption.

• **A1**
(1) There exists $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$.
(2) F is continuously differentiable in some neighborhood, say N, of x^* containing Ω.
(3) The Jacobian of F is bounded and positive definite on N, i.e. there exists positive constants $M > m > 0$ such that

$$\|F'(x)\| \leq M \quad \forall x \in N, \quad (22)$$

and

$$m\|d\|^2 \leq d^T F'(x)d \quad \forall x \in N, d \in \mathbb{R}^n. \quad (23)$$

From the level set we have:

$$\|F(x)\| \leq m_1 \quad \forall x \in \Omega. \quad (24)$$

Remark 3.1. Assumption A1 implies that there exist constants $M > m > 0$ such that

$$m\|d\| \leq \|F'(x)d\| \leq M\|d\| \quad \forall x \in N, d \in \mathbb{R}^n. \quad (25)$$

$$m\|x - y\| \leq \|F(x) - F(y)\| \leq M\|x - y\| \quad \forall x, y \in N. \quad (26)$$

In particular $\forall x \in N$ we have

$$m\|x - x^*\| \leq \|F(x)\| = \|F(x) - F(x^*)\| \leq M\|x - x^*\|, \quad (27)$$

where x^* stands, as usual, for the unique solution of (1) in N.

Since $\gamma_k I$ approximates $F'(x_k)$ along direction s_k, we can contemplate another assumption:

• **A2**
$\gamma_k I$ is a good approximation to $F'(x_k)$, i.e.

$$\|(F'(x_k) - \gamma_k I)d_k\| \leq \epsilon\|F(x_k)\|, \quad (28)$$

where $\epsilon \in (0, 1)$ is a small quantity [18].

Lemma 3.1. *Let A2 hold and $\{x_k\}$ be generated by algorithm 1. Then d_k is a descent direction for $f(x_k)$ at x_k i.e.*

$$\nabla f(x_k)^T d_k < 0. \tag{29}$$

Proof. From (10) we have

$$\nabla f(x_k)^T \mathrm{d}_k = F(x_k)^T F'(x_k)\mathrm{d}_k = F(x_k)^T[(F'(x_k)-\gamma_k \mathrm{I})\mathrm{d}_k - F(x_k)]$$
$$= F(x_k)^T((F'(x_K)-\gamma_k \mathrm{I})\mathrm{d}_k - \|F(x_k)\|^2. \tag{30}$$

By Cauchy Schwartz inequality it follows that

$$\nabla f(x_k)^T \mathrm{d}_k \leq \|F(x_k)\|\|((F'(x_K)-\gamma_k \mathrm{I})\mathrm{d}_k\| - \|F(x_k)\|^2 \leq -(1-\epsilon)\|F(x_k)\|^2. \tag{31}$$

Hence for $\epsilon \in (0,1)$ this lemma is true. ■

By lemma 3.1, we can deduce that the norm function $f(x_k)$ is a descent along d_k , which means that $\|F(x_{k+1})\| \leq \|F(x_k)\|$ is true.

Lemma 3.2. *Let A2 hold and $\{x_k\}$ be generated by algorithm 1. Then $\{x_k\} \subset \Omega$.*

Proof. By lemma 3.1 we have $\|F(x_{k+1})\| \leq \|F(x_k)\|$. Moreover, we have for all *k*,
$\|F(x_{k+1})\| \leq \|F(x_k)\| \leq \|F(x_{k-1})\| \ldots \leq \|F(x_0)\|$.
This implies that $\{x_k\} \subset \Omega$. ■

Lemma [18] **3.3**. *Suppose that A1 holds and {x_k} is generated by algorithm1. Then there exists a constant $m > 0$ such that, for all k,*

$$y_k^T s_k \geq m\|s_k\|^2. \tag{32}$$

Lemma 3.4. *Suppose that A1 holds and $\{x_k\}$ is generated by algorithm 1. Then*

$$\lim_{k\to\infty} \|\alpha_k d_k\| = 0, \tag{33}$$

and

$$\lim_{k\to\infty} \|\alpha_k F(x_k)\| = 0, \tag{34}$$

both hold.

Proof. By (20) we have for all $k > 0$

$$\omega_2\|\alpha_k d_k\|^2 \leq \omega_1\|\alpha_k F(x_k)\|^2 + \omega_2\|\alpha_k d_k\|^2$$
$$\leq \|F(x_k)\|^2 - \|F(x_{k+1})\|^2 + \eta_k\|F(x_k)\|^2. \tag{35}$$

By summing the above inequality, we can write

$$\omega_2 \sum_{i=0}^{k}\|\alpha_i d_i\|^2 \leq \sum_{i=0}^{k}(\|F(x_i)\|^2 - \|F(x_{i+1})\|^2) + \sum_{i=0}^{k}\eta_i\|F(x_i)\|^2$$
$$= \|F(x_0)\|^2 - \|F(x_{k+1})\|^2 + \sum_{i=0}^{k}\eta_i\|F(x_i)\|^2 \tag{36}$$
$$\leq \|F(x_0)\|^2 + m_1^2\sum_{i=0}^{k}\eta_i \leq \|F(x_0)\|^2 + \mathrm{m}_1^2\sum_{i=0}^{\infty}\eta_i. \tag{37}$$

So from (24) and the fact that $\{\eta_k\}$ satisfies (19), it follows that the series $\sum_{i=0}^{\infty}\|\alpha_i d_i\|^2$ is

convergent. This implies (33). In a similar way we can prove that (34) holds. ■

Lemma 3.5. *Suppose that A1 holds and $\{x_k\}$ is generated by algorithm 1. Then there exists some positive constants m_2 and m_1 such that, for all $k > 0$,*

$$\|d_k\| \leq m_2, \tag{38}$$

and

$$\|c_k\| \leq m_3. \tag{39}$$

Proof. By (26), we have

$$\|d_k\| = \left\| -\frac{F(x_k)\, y_{k-1}^T s_{k-1}}{\|y_{k-1}\|^2} \right\| \leq \frac{\|F(x_k)\|\,\|s_{k-1}\|\,\|y_{k-1}\|}{m^2\|s_{k-1}\|^2} \leq \frac{\|F(x_k)\|M\|s_{k-1}\|}{m^2\|s_{k-1}\|}$$
$$\leq \frac{\|F(x_k)\|M}{m^2} \leq \frac{\|F(x_0)\|M}{m^2}. \tag{40}$$

Taking $m_2 = \frac{\|F(x_0)\|M}{m^2}$, leads to (38).

Also, from (11) and (24) we have,

$$\|c_k\| = \|-F(x_k)\| \leq \|F(x_k)\| \leq m_1. \tag{41}$$

Here the proof completes. ■

Moreover, we can deduce that for all k, (38) and (39) hold. Now we are going to establish the following global convergence theorem to show that, under some suitable conditions, there exists an accumulation point of $\{x_k\}$ which is a solution to (1).

Theorem 3.1. *Suppose that A1 holds and $\{x_k\}$ is generated by algorithm 1.* Assume further, for all $k > 0$, that

$$\alpha_k \geq c\frac{|F(x_k)^T d_k|}{\|d_k\|^2}, \tag{42}$$

where c is some positive constant. Then

$$\lim_{k\to\infty} \|F(x_k)\| = 0. \tag{43}$$

Proof. From lemma 3.5 we arrive at (38). Therefore by (34) and the boundedness of $\{\|d_k\|\}$, we have

$$\lim_{k\to\infty} \alpha_k \|d_k\|^2 = 0. \tag{44}$$

From (42) and (44), it follows that

$$\lim_{k\to\infty} |F(x_k)^T d_k| = 0. \tag{45}$$

On the other hand, (10) leads to

$$F(x_k)^T d_k = -\gamma_k^{-1}\|F(x_k)\|^2, \tag{46}$$

and

$$\|F(x_k)\|^2 = \|-F(x_k)^T d_k \gamma_k\| \leq |F(x_k)^T d_k \gamma_k|. \tag{47}$$

But

$$\gamma_k^{-1} = \frac{y_{k-1}^T s_{k-1}}{\|y_{k-1}\|^2} \geq \frac{m\|s_{k-1}\|^2}{\|y_{k-1}\|^2} \geq \frac{m\|s_{k-1}\|^2}{M^2\|s_{k-1}\|^2} = \frac{m}{M^2}.$$

then

$|\gamma_k^{-1}| \geq \frac{m}{M^2}$.

So from (47) we have,

$$\|F(x_k)\|^2 \leq |F(x_k)^T d_k|\left(\frac{M^2}{m}\right). \tag{48}$$

Thus

$$0 \leq \|F(x_k)\|^2 \leq |F(x_k)^T d_k|\left(\frac{M^2}{m}\right) \to 0. \tag{49}$$

Therefore

$$\lim_{k\to\infty} \|F(x_k)\| = 0. \tag{50}$$

At this point, the proof is completed. ■

4. Numerical Results

The performance of our method for solving nonlinear equations (1) is compared in this section, with a derivative-free CG method. Here also we study the global convergence when solving symmetric nonlinear equations [12]. It should be noted here that:

(i) A transformed double step length (TDS) stands for our method, for which we set the following:

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

(ii) A derivative-free CG (DFCG) is the method proposed in [12] and for it we have

$\omega_1 = \omega_2 = 10^{-4}$, $\alpha_0 = 0.01$, $r = 0.2$ and $\eta_k = \frac{1}{(k+1)^4}$.

Problem 1:

$$F(x) = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} x + (e_1^x - 1, \ldots, e_n^x - 1)^T. \quad x_0 = (0.5, 0.5, \ldots, 0.5)^T.$$

Problem 2:

$$F(x) = \begin{pmatrix} 2 & -1 & & & \\ 0 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} x + (sinx_1 - 1, \ldots, sinx_n - 1)^T. \quad x_0 = (1, 1, \ldots, 1)^T.$$

Problem 3:

$F_1(x) = x_1(x_1^2 + x_2^2) - 1,$
$F_i(x) = x_i(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2),$
$F_n(x) = x_n(x_{n-1}^2 + x_n^2),$
$i = 2, 3, \ldots, n-1.$
$x_0 = (0.01, 0.01, \ldots, 0.01)^T.$

Problem 4:
$F_{3i-2}(x) = x_{3i} - 2x_{3i-1} - x_{3i}^2 - 1,$
$F_{3i-2}(x) = x_{3i} - 2x_{3i-1} - x_{3i}^2 - 1,$
$F_{3i}(x) = e^{-x_{3i-2}} - e^{-x_{3i-1}}.$
$i = 1, \ldots, \frac{n}{3}.$
$x_0 = (0.1, 0.1, \ldots, 0.1)^T.$

Problem 5:
$F_i(x) = (1 - x_i^2) + x_i(1 + x_i x_{n-2} x_{n-1} x_n) - 2,$
$i = 1, 2, \ldots, n.$
$x_0 = (0.7, 0.7, \ldots, 0.7)^T.$

Problem 6:
$F_1(x) = x_1^2 - 3x_1 + 1 + cos(x_1 - x_2),$
$F_i(x) = x_1^2 - 3x_i + 1 + cos(x_i - x_{i-1}),$
$i = 1, 2, \ldots, n.$
$x_0 = (0.4, 0.4, \ldots, 0.4)^T.$

Problem 7:
$F_i(x) = x_i - 0.1x_{i+1}^2,$
$F_n(x) = x_n - 0.1x_1^2,$
$i = 1, 2, \ldots, n-1.$
$x_0 = (1,1,...,1)^T.$

Problem 8:
$F_i(x) = 0.i\ (1 - x_i)^2 - e^{-x_i^2},$
$F_n(x) = \frac{n}{10}(1 - e^{-x_n^2}),$
$i = 1, 2, \ldots, n-1.$
$x_0 = (-0.1, -0.1, \ldots, -0.1)^T.$

Problem 9:
$F_i(x) = 2x_i - sin|x_i|,$
$i = 1, 2, \ldots, n.$
$x_0 = (-0.1, -0.1, \ldots, -0.1)^T.$

Problem 10:
$F_1 = x_1 - e^{cos\left(\frac{x_1 + x_2}{n+1}\right)},$
$F_i = x_i - e^{cos\left(\frac{x_{i-1} + x_i + x_{i+1}}{n+1}\right)},$

$F_n = x_n - e^{cos\left(\frac{x_{n-1}+x_n}{n+1}\right)}$,
$i = 2,3,\ldots,n-1.$
$x_0 = (-2,-2,\ldots,-2)^T.$

The employed computational codes were written in Matlab 7.9.0 (R2009b) and run on a personal computer 2.00 GHz CPU processor and 3 GB RAM memory. We stopped the iterations if the total number of iterations exceeds 1000 or $\|F(x_k)\| \leq 10^{-4}$. We have tried the two methods on the previous ten test problems with different initial points and dimension (n values). Problems 1-7 are from [12] and problem 8 was arbitrarily constructed by us, while problems 9 and 10 are from [17].

Table 1: The numerical results for TDS and DFCG for problems 1 to 5.

		TDS			DFCG		
Problems	Dim	Iter	Time(s)	$\|F(x_k)\|$	Iter	Time(s)	$\|F(x_k)\|$
1	10	14	0.059241	2.79E-05	34	0.139560	8.37E-05
	100	15	0.071557	1.49E-05	38	0.189075	9.55E-05
	1000	16	0.490706	9.60E-05	53	2.471807	8.72E-05
	10000	15	1.473685	8.21E-05	54	8.348075	8.10E-05
2	10	11	0.054593	5.34E-05	49	0.175516	4.08E-05
	100	12	0.069415	4.08E-05	60	0.304922	8.65E-05
	1000	12	0.353096	6.48E-05	67	3.228993	8.22E-05
	10000	12	1.143396	6.76E-05	75	12.382064	9.61E-05
3	10	15	0.005725	2.51E-05	130	0.048701	9.10E-05
	100	16	0.007151	5.57E-05	124	0.065831	9.16E-05
	1000	16	0.029844	9.39E-05	129	0.213241	9.40E-05
	10000	13	0.113851	8.93E-05	127	1.574677	8.34E-05
4	10	7	0.006096	2.31E-05	62	0.043354	9.69E-05
	100	7	0.005472	6.67E-05	74	0.042591	5.19E-05
	1000	8	0.012189	9.65E-05	77	0.164359	4.90E-05
	10000	9	0.094071	6.65E-05	78	1.147510	6.30E-05
5	10	7	0.0021951	1.67E-05	11	0.008832	1.86E-05
	100	7	0.004831	5.29E-05	11	0.012392	5.88E-05
	1000	8	0.013615	7.90E-07	12	0.044482	2.70E-07
	10000	8	0.096588	2.50E-06	12	0.148847	8.55E-07

The numerical results of the two methods are reported in tables 1 and 2,where "Iter" and "Time" stand for the total number of all iterations and the CPU time in seconds, respectively, while $\|F(x_k)\|$ is the norm of the residual at the stopping point. From tables 1 and 2, we can easily observe that both of these methods attempt to solve the systems of nonlinear equations (1), but the better efficiency and effectiveness of our proposed algorithm was clear for it solves where DFCG fails. This is quite evident for instance with problem 6. In particular, the TDS method considerably outperforms the DFCG for almost all the tested problems, as it has the least number of iterations and CPU time, which are even much less than the CPU for the DFCG method. This is apparently due to the computation of step length in each iteration of the TDS as well as the approximation of the Jacobian through the acceleration parameter.

Table 2: The numerical results for TDS and DFCG for problems 6 to 10.

		TDS			DFCG		
Problems	Dim	Iter	Time(s)	$\|F(x_k)\|$	Iter	Time(s)	$\|F(x_k)\|$
6	10	6	0.005406	3.24E-06	—	—	—
	100	6	0.004294	1.02E-05	—	—	—
	1000	6	0.022208	3.24E-05	—	—	—
	10000	7	0.095078	4.73E-07	—	—	—
7	10	4	0.003252	6.59E-06	5	0.006986	5.23E-06
	100	4	0.002224	1.91E-05	5	0.059992	2.35E-05
	1000	4	0.006047	5.89E-05	5	0.303477	7.52E-05
	10000	5	0.059346	9.50E-07	6	5.610828	4.64E-08
8	10	6	0.003555	6.23E-05	10	0.003783	6.53E-05
	100	8	0.003634	8.56E-05	9	0.010739	5.72E-05
	1000	11	0.017507	4.88E-05	9	0.031811	9.15E-05
	10000	13	0.118862	6.76E-05	—	—	—
9	10	4	0.003177	8.12E-05	6	0.003853	3.53E-11
	100	6	0.005008	1.53E-06	6	0.003912	1.12E-10
	1000	6	0.009973	4.82E-06	6	0.012268	3.53E-10
	10000	6	0.064697	1.53E-05	6	0.085342	1.12E-09
10	10	7	0.004101	9.59E-05	13	0.006429	4.42E-05
	100	5	0.004108	1.72E-06	5	0.004818	2.78E-06
	1000	4	0.018428	1.35E-05	4	0.018432	4.78E-06
	10000	4	0.076609	1.81E-05	3	0.090856	4.32E-05

Figures (1-2) exhibit the better performance of our method relative to the number of iterations and CPU time, which were evaluated using the profiles of Dolan and Mor*é* [5]. That is, for each method, we plot the fraction $P(\tau)$ of the problems for which the method is within a

factor τ of the best time. The top curve is the method that solved most problems in a time that was within a factor τ of the best time.

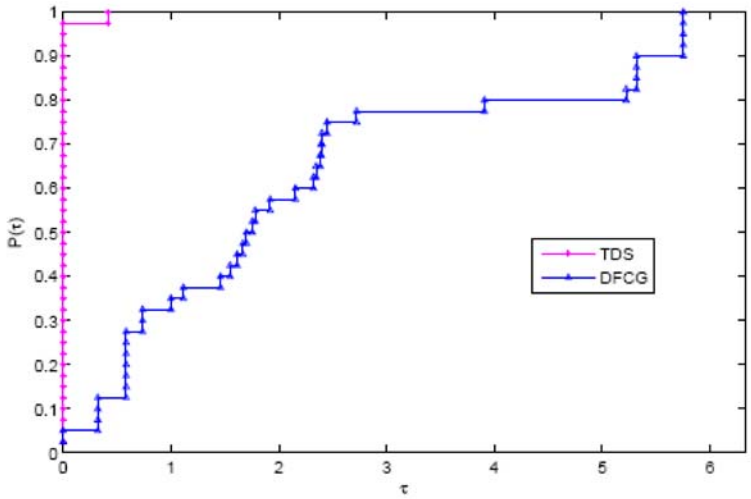

Figure 1: Performance Profiles of TDS and DFCG methods with respect to the number of iterations for problems 1-10

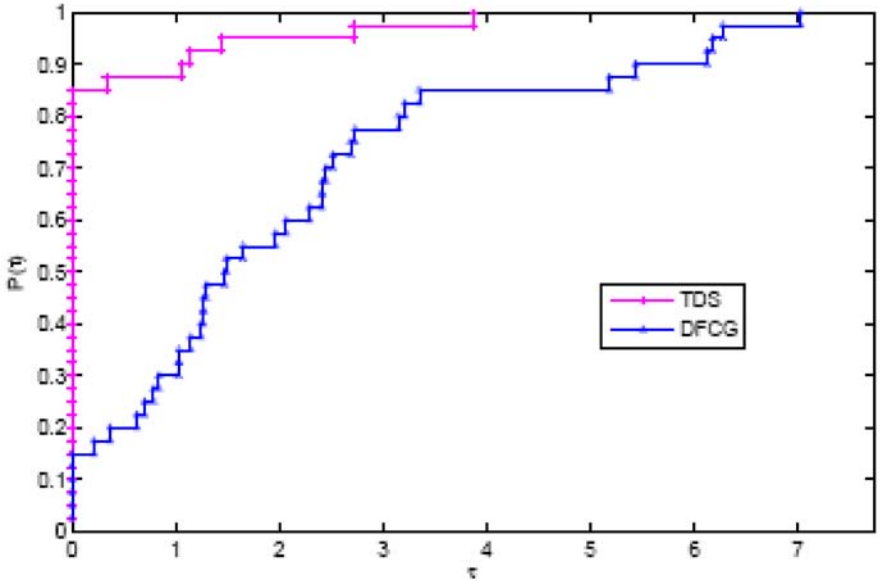

Figure 2: Performance Profiles of TDS and DFCG methods with respect to the CPU time (in seconds) for problems 1-10

5. Conclusion

In this paper we presented a transformed double step length (TDS) method for solving large-scale systems of nonlinear equations and compared its numerical performance with that of a derivative-free conjugate gradient (DFCG) method for symmetric systems of nonlinear equations [12]. We have proved the global convergence of our proposed method by using a backtracking type line search. Also the numerical results of the reported experiments demonstrate the good efficiency of our method.

References

[1] C. G. Broyden, A class of methods for solving nonlinear simultaneous equations, *Mathematics of Computation* **19**, (1965), 577-593.

[2] N. I. Dbaruranovic-milicic, A multi-step curve search algorithm in nonlinear optimization, *Yugoslav Journal of Operations Research* **18**, (2008), 47-52.

[3] D. W. Decker, and C.T. Kelley, Broyden's methods for a class of problems having singular Jacobian at root, *SIAM Journal of Numerical Analysis* **17**(1), (1985), 566-574.

[4] J. E. Dennis, and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Non-Linear Equations*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[5] E. Dolan, and J. More, Benchmarking optimization software with performance profiles, *Journal of Mathematical Programing* **91**(2), (2002), 201-213.

[6] D. Li, and M. Fukushima, A global and superlinear convergent Gauss-Newton based BFGS method for symmetric nonlinear equation, *SIAM Journal on Numerical Analysis* **37**, (1999), 152-172.

[7] K. Natasa, and L. Zorna, Newton-like method with modification of rigtht-hand vector, *Mathematics of Computation* **71**, (2001), 237-250.

[8] M. J. Petrovic, and P. S. Stanimirovic, Accelerated double direction method for solving unconstrained optimization problems, *Mathematical Problems in Engineering* **2014**, (2014), Article ID 965104, 8p.

[9] M. J. Petrovic, An accelerated double step size model in unconstrained optimization, *Journal of mathematics and Computation* **250**, (2015), 309-319.

[10] M. Raydan, On Barzilai and Borwein choice of step length for the gradient method, *IMA Journal of Numerical Analysis* **13**, (1993), 321-326.

[11] P. S. Stanimirovic, G. V. Milovanovic, M. J. Petrovic, and N. Z. Kontrec, A transformation of accelerated double step size method for unconstrained optimization, *Journal of Mathematical Problems in Engineering* **2015**, (2015), Article ID 283679, 8p.

[12] M.Y. Waziri and J. Sabiu, A derivative-free conjugate gradient method and its global convergence for symmetric nonlinear equations, *Journal of Mathematics and Mathematical Sciences* **2015** (2015), Article ID 961487, 8p.

[13] M.Y. Waziri, W. J. Leong, M. A. Hassan, and M. Monsi, A new Newton's method with diagonal Jacobian approximation for system of nonlinear equations, *Journal of Mathematics and Statistics* **6**(3), (2010), 246-252.

[14] M.Y. Waziri, W. J. Leong, and M. A. Hassan, Jacobian-free diagonal Newton's method for solving nonlinear systems with singular Jacobian, *Malaysian Journal of Mathematical Science* **5**, (2011), 241-255.

[15] M.Y. Waziri, W. J. Leong, M. A. Hassan, and M. Monsi, Jacobian computation-free Newton method for systems of non-linear equations, *Journal of Numerical Mathematics and Stochastics* **2**, (2010), 54-63.

[16] M.Y. Waziri, and Z. A. Majid, An enhanced matrix-free secant mehod via predictor-corrector modified line search strategies for solving systems of nonlinear equations, *Journal Mathematics and Mathematical Sciences* **2013**, (2013), Article ID 814587, 6p.

[17] Q-R. Yana, X-Z. Penga, and D-H. Li, A globally convergent derivative-free method for solving large-scale nonlinear monotone equations, *Journal of Computational and Applied Mathematics* 234, (2010), 649-657.

[18] G. Yuan, and X. Lu, A new backtracking inexact BFGS method for symmetric nonlinear equations, *Computers and Mathematics With Applications* **55**, (2008), 116-129.

[19] L. Zang, W. Zhou, and D. H. Li, Global convergence of modified Fletcher-Reeves conjugate gradient method with Armijo-type line search, *Numerische Mathematik* **164**(1), (2005), 277-289.

[20] W. Zhou, and D. Shen, An inexact PRP conjugate gradient method for symmetric nonlinear equations, *Numerical Functional Analysis and Optimization* **35**, (2014), 370-388.

Article history: Submitted January, 01, 2017; Revised May, 03, 2017; Accepted June, 06, 2017.