Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 7, No. 1, 2019, pp. 69-85

An efficient improvement of the Newton method for solving nonconvex optimization problems

Tayebeh Dehghan Niri*
Department of Mathematics, Yazd University,
P. O. Box 89195-74, Yazd, Iran.
E-mail: ta.dehghan18175@gmail.com,
T. Dehghan@stu.yazd.ac.ir

Mohammad Mehdi Hosseini
Department of Mathematics, Yazd University,
P. O. Box 89195-74, Yazd, Iran.
E-mail: hosse_m@yazd.ac.ir

Mohammad Heydari
Department of Mathematics, Yazd University,
P. O. Box 89195-74, Yazd, Iran.
E-mail: m.heydari@yazd.ac.ir

Abstract Newton method is one of the most famous numerical methods among the line search methods to minimize functions. It is well known that the search direction and step length play important roles in this class of methods to solve optimization problems. In this investigation, a new modification of the Newton method to solve unconstrained optimization problems is presented. The significant merit of the proposed method is that the step length α_k at each iteration is equal to 1. Additionally, the convergence analysis for this iterative algorithm is established under suitable conditions. Some illustrative examples are provided to show the validity and applicability of the presented method and a comparison is made with several other existing methods.

Keywords. Unconstrained optimization, Newton method, Line search methods, Global convergence.
2010 Mathematics Subject Classification. 65K05, 90C26, 90C30, 49M15.

1. Introduction

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable and smooth function. Consider the minimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.1}$$

and assume that the solution set of (1.1) is nonempty. Among the optimization methods, classical Newton method is well known for its fast convergence property. However, the Newton step size may not be a descent direction of the objective function or even not well defined when the Hessian is not a positive definite matrix.

Received: 26 March 2017 ; Accepted: 23 October 2018.
∗ Corresponding author.

There are many improvements of the Newton method for unconstrained optimization to achieve convergence. Zhou et al. [24] presented a new method for monotone equations, and showed its superlinear convergence under a local error-bound assumption that is weaker than the standard nonsingularity condition. Li et al. [11] obtained two regularized Newton methods for convex minimization problems in which the Hessian at solutions may be singular and showed that if f is twice continuously differentiable, then the methods possess local quadratic convergence under a local error bound condition without requiring isolated nonsingular solutions. Ueda and Yamashita [21] applied a regularized algorithm for nonconvex minimization problems. They presented a global complexity bound and analyzed the super linear convergence of their method. Polyak [16] proposed the regularized Newton method for unconstrained convex optimization. For any convex function, with a bounded optimal set, the RNM (regularized Newton method) generates a sequence converging to the optimal set from any starting point. Shen et al. [19] proposed a regularized Newton method to solve unconstrained nonconvex minimization problems without assuming the non-singularity of solutions. They also proved its global and fast local convergences under suitable conditions.

In this paper, we propose a new algorithm to solve unconstrained optimization problems. The organization of the paper is as follows: In section 2, we introduce a new regularized method to solve minimization problems. Section 3 presents the global convergence analysis of our algorithm. Some preliminary numerical results are reported in section 4 and some concluding remarks are presented in the final section.

2. Regularized Newton method

We consider the unconstrained minimization problem,

$$\min_{x\in\mathbb{R}^n} f(x), \tag{2.1}$$

where $f:\mathbb{R}^n \longrightarrow \mathbb{R}$ is twice continuously differentiable. We suppose that, for a given $x_0 \in \mathbb{R}^n$, the level set

$$L_0 = \{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}, \tag{2.2}$$

is compact. Gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ are denoted by $g(x)$ and $H(x)$, respectively. In general, numerical methods, based on line search to solve the problem (2.1) have the following iterative formula

$$x_{k+1} = x_k + \alpha_k p_k, \tag{2.3}$$

where x_k, α_k and p_k are current iterative point, a positive step size and a search direction, respectively. The success of linear search methods depends on the appropriate selections of step length α_k and direction p_k. Most line search methods require p_k to be a descent direction, since this property guarantees that the function f can be decreased along this direction.
For example, the steepest descent direction is represented by

$$p_k = -g_k, \tag{2.4}$$

and Newton-type direction uses

$$p_k = -H_k^{-1} g_k. \tag{2.5}$$

Generally, the search direction can be defined as

$$p_k = -B_k^{-1}\nabla f_k, \tag{2.6}$$

where B_k is a symmetric and nonsingular matrix. If B_k is not positive definite, or is close to being singular, then one can modify this matrix before or during the solution process. A general description of this modification is presented as follows [15].

Algorithm 1. (Line Search Newton with Modification):
For given initial point x_0 and parameters $\alpha > 0, \beta > 0$;
while $\nabla f(x_k) \neq 0$
Factorize the matrix $B_k = \nabla^2 f(x_k) + E_k$,
where $E_k = 0$ if $\nabla^2 f(x_k)$ is sufficiently positive definite;
otherwise, E_k is chosen to ensure that B_k is sufficiently positive definite;
Solve $B_k p_k = -\nabla f(x_k)$;
Set $x_{k+1} = x_k + \alpha_k p_k$,
where α_k satisfies the Wolfe, Goldstein, or Armijo backtracking conditions.
end while.

The choice of Hessian modification E_k is crucial to the effectiveness of the method.

2.1. **The regularized Newton method.** Newton method is one of the most popular methods in optimization and to find a simple root δ of a nonlinear equation $f(x)$, i.e., $f(\delta) = 0$ in case $f'(\delta) \neq 0$, by using

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \ k = 0, 1, \ldots, \tag{2.7}$$

that converges quadratically in some neighborhoods of δ [20, 3]. The modified Newton method for multiple root δ of multiplicity m, i.e., $f^{(j)}(\delta) = 0, \ j = 0, 1, \ldots, m-1$ and $f^{(m)}(\delta) \neq 0$, is quadratically convergent and it is written as

$$x_{k+1} = x_k - m\frac{f(x_k)}{f'(x_k)}, \ k = 0, 1, \ldots, \tag{2.8}$$

which requires the knowledge of the multiplicity m. If the multiplicity m is unknown, the standard Newton method has a linear convergence with a rate of $(m-1)/m$ [4]. Traub in [20] used a transformation $\mu(x) = \frac{f(x)}{f'(x)}$ instead of $f(x)$ to compute a multiple root of $f(x) = 0$. Then the problem of finding a multiple root is reduced to the problem of finding a simple root of the transformed equation $\mu(x)$, and thus any iterative method can be used to maintain its original convergence order. Applying the standard Newton method (2.7) to $\mu(x) = 0$, we can obtain

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - f(x_k)f''(x_k)}, \ k = 0, 1, \ldots. \tag{2.9}$$

This method can be extended to n-variable functions as

$$X_{k+1} = X_k - \left(\nabla f(X_k)\nabla f(X_k)^T - f(X_k)\nabla^2 f(X_k)\right)^{-1} f(X_k)\nabla f(X_k), \tag{2.10}$$

for $k = 0, 1, \ldots$.
In this section, we introduce a new search direction for the Newton method. The presented method is obtained by investigating the following parametric family of the iterative method

$$X_{k+1} = X_k - \Big(\beta \nabla f(X_k).\nabla f(X_k)^T - f(X_k)\nabla^2 f(X_k)\Big)^{-1} \theta f(X_k)\nabla f(X_k), \tag{2.11}$$

where $\theta, \beta \in \mathbb{R} - \{0\}$ are parameters to be determined and $k = 0, 1, \ldots$.
When $\theta = \beta = 2$, (2.11) reduces to the Halley method [12, 18], which is defined by

$$X_{k+1} = X_k - \Big(2\nabla f(X_k).\nabla f(X_k)^T - f(X_k)\nabla^2 f(X_k)\Big)^{-1} 2f(X_k)\nabla f(X_k), \tag{2.12}$$

for $k = 0, 1, \ldots$.

Remark 2.1. If $\beta = 0$ and $\theta = -1$, the proposed method reduces to the classical Newton method,

$$X_{k+1} = X_k - \left(\nabla^2 f(X_k)\right)^{-1} \nabla f(X_k), \ k = 0, 1, \ldots.$$

Now, we present a general algorithm to solve unconstrained optimization problems by using (2.11).

Algorithm 2. (Regularized Newton method):
Step 1. Given initial point x_0, $\tau > 0$, θ, β and ϵ.
Step 2. If $\|f_k g_k\| = 0$ stop.

Step 3. If $B_k = (\beta g_k g_k^T - f_k H_k)$ is a nonsingular matrix then
Solve $(\beta g_k g_k^T - f_k H_k)p_k = -\theta f_k g_k$;
else
Solve $(\beta g_k g_k^T - f_k H_k + \tau I)p_k = -\theta f_k g_k$;

Step 4. $x_{k+1} = x_k + p_k$.
Set $k := k + 1$ and go to Step 2.

We remind that $B_k = (\beta g_k g_k^T - f_k H_k)$ is a symmetric matrix. This algorithm is a simple regularized Newton method.

3. Global convergence

In this section, we study the global convergence of Algorithm 2. We first give the following assumptions.

Assumption 3.1.

(A1): The mapping f is twice continuously differentiable, below bounded and the level set

$$L_0 = \{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}, \tag{3.1}$$

is bounded.

(A2): $g(x) \in \mathbb{R}^{n\times 1}$ and $H(x) \in \mathbb{R}^{n\times n}$ are both Lipschitz continuous that is, there exists a constant $L > 0$ such that

$$\|g(x) - g(y)\| \le L\|x - y\|,\ x, y \in \mathbb{R}^n, \tag{3.2}$$

and

$$\|H(x) - H(y)\| \le L\|x - y\|,\ x, y \in \mathbb{R}^n. \tag{3.3}$$

(A3): $\beta \in \mathbb{R} - \{0\}$ and $2\beta(\nabla f_k^T(f_k\nabla^2 f_k)^{-1}\nabla f_k) \le 1$.

Theorem 3.2. *Suppose A is a nonsingular $N\times N$ matrix, U is $N\times M$, V is $M\times N$, then $A+UV$ is nonsingular if and only if $I+VA^{-1}U$ is a nonsingular $M\times M$ matrix. If this is the case, then*

$$(A+UV)^{-1} = A^{-1} - A^{-1}U(I+VA^{-1}U)^{-1}VA^{-1}.$$

This is the Sherman-Morrison-Woodbury formula [10, 9, 22]. See [10] for further generalizations.

Proposition 3.3. [10] *Let B be a nonsingular $n\times n$ matrix and let $u, v \in \mathbb{R}^n$. Then $B+uv^T$ is invertible if and only if $1+v^TB^{-1}u \neq 0$. In this case,*

$$(B+uv^T)^{-1} = \left(I - \frac{B^{-1}uv^T}{1+v^TB^{-1}u}\right)B^{-1}.$$

Lemma 3.4. *Suppose that Assumption 3.1 **(A1)** and **(A3)** hold. Then*

(I) $\left|\frac{\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k}{1+\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k}\right| \le 1,$

(II) $(\frac{-f_k}{\beta}\nabla^2 f_k + \nabla f_k\nabla f_k^T)^{-1} = \frac{-\beta}{f_k}(\nabla^2 f_k)^{-1}(I - \frac{\nabla f_k\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}}{1+\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k}).$

Proof. From Assumption 3.1 **(A3)**, we have

$$\beta(\nabla f_k^T(-f_k\nabla^2 f_k)^{-1}\nabla f_k) \ge -\frac{1}{2} \Longrightarrow \nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k \ge -\frac{1}{2}, \tag{3.4}$$

and hence

$$\left|\frac{\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k}{1+\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k}\right| \le 1.$$

According to Theorem 3.2 and Proposition 3.3, we set $B = \frac{-f_k}{\beta}\nabla^2 f_k$, $u = v = \nabla f_k$. From (3.4) we obtain

$$1 + v^TB^{-1}u = 1 + \nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k \ge \frac{1}{2}.$$

Therefore, the matrix $B+uv^T$ is invertible and we can get

$$(B+uv^T)^{-1} = (\tfrac{-f_k}{\beta}\nabla^2 f_k + \nabla f_k\nabla f_k^T)^{-1}$$
$$= (\tfrac{-f_k}{\beta}\nabla^2 f_k)^{-1} - (\tfrac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k(1+\nabla f_k^T(\tfrac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k)^{-1}\nabla f_k^T(\tfrac{-f_k}{\beta}\nabla^2 f_k)^{-1}$$
$$= \tfrac{-\beta}{f_k}(\nabla^2 f_k)^{-1}(I - \frac{\nabla f_k\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}}{1+\nabla f_k^T(\frac{-f_k}{\beta}\nabla^2 f_k)^{-1}\nabla f_k}).$$

□

Theorem 3.5. *Suppose that the sequence $\{x_k\}$ generated by Algorithm 2 is bounded. Then we have*

$$\lim_{k\to\infty} \|f_k g_k\| = 0.$$

Proof. First, we prove that p_k is bounded. Suppose that $\gamma = L\|(\nabla^2 f(x^*))^{-1}\|$. From the definitions of p_k in Algorithm 2, we have

$$\|p_k\| \leq |\theta|.\|(\nabla^2 f(x_k))^{-1}\|.\|\nabla f(x_k)\| \leq \frac{2|\theta|\gamma}{L}\|\nabla f(x_k)\|,$$

suppose that $\{x_k\} \subseteq \Lambda$ and $C = \sup_{x\in\Lambda}\{\|\nabla f_k\|\} < +\infty$, therefore

$$\|p_k\| \leq \frac{2|\theta|\gamma}{L}C,$$

which proves that p_k is bounded for all k. By Taylor Theorem, we have

$$f(x_k + p_k) = f(x_k) + \nabla f_k^T p_k + \frac{1}{2}p_k^T \nabla^2 f(x_k + tp_k)p_k$$

$$= f(x_k) + \nabla f_k^T p_k + O(\|p_k\|^2),$$

therefore by the boundedness of p_k

$$f(x_k + p_k) - f(x_k) - \sigma\nabla f_k^T p_k = (1-\sigma)\nabla f_k^T p_k + O(\|p_k\|^2)$$

$$= -(1-\sigma)\nabla f_k^T B_k^{-1} f_k \nabla f_k + O(\|p_k\|^2).$$

If $\nabla f_k^T B_k^{-1} f_k \nabla f_k > 0$, then

$$f(x_k + p_k) - f(x_k) - \sigma\nabla f_k^T p_k \leq 0,$$

therefore

$$f(x_k + p_k) \leq f(x_k) + \sigma\nabla f_k^T p_k. \tag{3.5}$$

Hence, the sequence $\{f(x_k)\}$ is decreasing. Since $f(x)$ is bounded below, it follows that

$$\lim_{k\to\infty} f_k = \underline{f},$$

where $\underline{f}$ is a constant. Now we prove that

$$\lim_{k\to\infty} \nabla f_k^T B_k^{-1} f_k \nabla f_k = 0. \tag{3.6}$$

With assuming $\nabla f_k^T B_k^{-1} f_k \nabla f_k > 0$ there exists a scalar $\epsilon > 0$ and an infinite index set Γ such that $\nabla f_k^T B_k^{-1} f_k \nabla f_k > \epsilon$ for all $k \in \Gamma$. According to (3.5), we obtain

$$f(x_k + p_k) - f(x_k) \leq \sigma\nabla f_k^T p_k. \tag{3.7}$$

Then

$$\underline{f} - f(x_0) = \sum_{k=0}^{\infty}(f(x_{k+1}) - f(x_k)) \leq \sum_{k\in\Gamma}(f(x_{k+1}) - f(x_k)) \leq \sum_{k\in\Gamma}\sigma\nabla f_k^T p_k$$

$$= -\sum_{k\in\Gamma}\sigma\nabla f_k^T B_k^{-1} f_k \nabla f_k.$$

This implies that $\nabla f_k^T B_k^{-1} f_k \nabla f_k \to 0$ as $k \to \infty$ and $k \in \Gamma$, which contradicts the fact $\nabla f_k^T B_k^{-1} f_k \nabla f_k > \epsilon, k \in \Gamma$. Hence, the whole sequence $\{\nabla f_k^T B_k^{-1} f_k \nabla f_k\}$ tends to zero.
Assumption 3.1 **(A2)** and the boundedness of $\{x_k\}$ show that

$$\lambda_{\max}\left(B_k^{-1} f_k\right) \leq \overline{\lambda}, \tag{3.8}$$

for all k, where $\overline{\lambda}$ is a positive constant. Therefore due to matrices property, $\lambda_{\min}(B_k^{-1} f_k) \geq \widehat{\lambda}$ for some constant $\widehat{\lambda} > 0$. Hence, $\nabla f_k^T B_k^{-1} f_k \nabla f_k \geq \widehat{\lambda} f_k \|\nabla f(x_k)\|^2$. Then from (3.6), $\|f_k g_k\| \longrightarrow 0$ as $k \to \infty$. □

4. Numerical results

In this section, we report some results on the following numerical experiments for the proposed algorithm. In addition, we have compared the effectiveness of the proposed method with the Improved Cholesky factorization, ([15], Chapter 6, Page 148) regularized Newton (RN) [7] and Halley method [12]. In Algorithm 2, we have used $\tau = 10^{-6}$ and in Improved Cholesky factorization (LDL^T), we have assumed $c_1 = 10^{-4}$, $\alpha_0 = 1, \rho = \frac{1}{2}, \delta = \sqrt{eps}$ and $\epsilon = 10^{-5}$. Furthermore, Improved Cholesky factorization uses the Armijo step size rule. N_f and N_g represent the number of the objective function and its gradient evaluations, respectively. All these algorithms are implemented in Matlab 12.0. The test functions are commonly used for unconstrained test problems with standard starting points and summary of them are given in Table 1 [1, 2, 13].

Table 1. Test problems [1, 2, 13].

No.	Name	No.	Name
1	Powell Singular	16	NONDQUAR
2	Extended Beale	17	ARWHEAD
3	HIMMELH	18	Broyden Tridiagonal
4	SINE	19	Extended DENSCHNB
5	FLETCHCR	20	Extended Trigonometric
6	LIARWHD	21	Extended Himmelblau
7	DQDRTIC	22	Extended Block Diagonal BD1
8	NONSCOMP	23	Full Hessian FH2
9	NONDIA	24	EG2
10	wood	25	EG3
11	Brown badly scaled	26	ENGVAL8
12	Griewank	27	Generalized Quartic
13	Extended Powell	28	Broyden Pentadagonal
14	Diagonal Double Bounded Arrow Up	29	Freudenstein and Roth
15	Diagonal full borded	30	INDEF

The numerical results are listed in Table 2. As we can see from Table 2, Algorithm 2 is more effective than the other three methods.

TABLE 2. Numerical results.

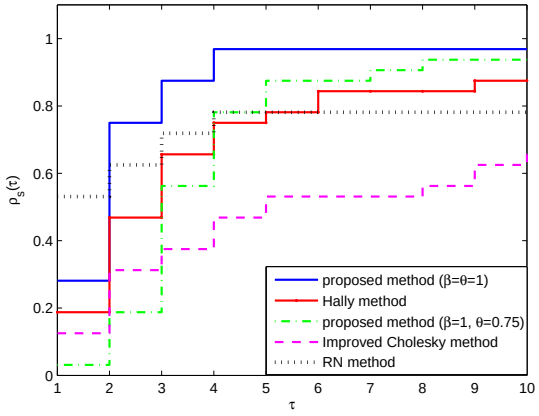
	Algorithm 2 $\beta = \theta = 1$ Proposed	Algorithm 2 $\beta = \theta = 2$ (Halley)	Algorithm 2 $\beta = 1, \theta = 0.75$ Proposed	Improved Cholesky method	RN method [7]
No./Dim	N_f/N_g/CPU f	N_f/N_g/CPU f	N_f/N_g/CPU f	N_f/N_g/CPU f	N_f/N_g/CPU f
1/4	6/6/ 0.479 3.82 e-7	9/9/ 0.761 1.96 e-5	8/8/ 0.666 2.61 e-6	15/16/ 2.881 4.38 e-9	2/5/ 0.079 5.47 e-58
1/400	7/7/ 8.303 2.044 e-9	11/11/ 13.988 5.55 e-5	10/10/ 12.489 4.09 e-5	17/18/ 29.863 1.709 e-8	2/5/ 5.496 1.16 e-41
2/50	5/5/ 0.964 1.90 e-8	9/9/ 1.848 5.55 e-5	9/9/ 1.862 2.19 e-6	44/9/ 6.851 7.77 e-13	4/9/ 1.742 1.12 e-23
2/500	5/5/ 11.576 1.86 e-7	10/10/ 26.378 1.32 e-5	10/10/ 26.414 1.43 e-6	44/9/ 50.450 8.058 e-12	4/9/ 22.787 1.09 e-22
3/300	5/5/ 2.880 −4.036 e-12	4/4/ 2.197 −3.467 e-12	16/16/ 11.031 1.48 e-8	4/5/ 3.634 -1.500 e+2	4/9/ 5.451 -1.500 e+2
4/60	11/11/ 1.959 2.26 e-14	6/6/ 0.999 1.91 e-11	29/29/ 5.322 -3.55 e-9	FAIL -	FAIL -
5/500	26/26/ 63.140 1.97 e-10	FAIL -	FAIL -	FAIL -	FAIL -
6/500	9/9/ 16.713 1.09 e-9	13/13/ 25.312 4.68 e-6	16/16/ 31.479 4.79 e-6	11/12/ 28.933 8.89 e-15	5/11/ 21.101 1.57 e-17
7/100	9/9/ 5.456 1.93 e-11	7/7/ 4.060 -8.77 e-11	28/28/ 17.875 5.05 e-11	FAIL -	FAIL -
8/500	FAIL -	11/11/ 29.460 3.04 e-5	8/8/ 20.431 9.82 e-6	8/9/ 26.587 1.50 e-15	9/19/ 48.852 NaN
9/500	6/6/ 10.646 1.02 e-7	13/13/ 25.466 7.83 e-7	11/11/ 21.149 3.082 e-7	6/7/ 15.367 2.41 e-19	4/9/ 15.588 3.57 e-21
10/4	47/47/ 1.106 5.93 e-9	43/43/ 1.015 2.38 e-6	33/33/ 0.768 1.04 e-6	61/40/ 1.674 8.75 e-20	29/59/ 0.954 3.046 e-26
11/2	12/12/ 0.246 4.93 e-30	996/996/ 59.632 5.46 e-9	30/30/ 0.509 3.15 e-9	FAIL -	11/23/ 0331 0
12/50	6/6/ 83.762 5.94 e-13	7/7/ 95.341 1.67 e-5	8/8/ 113.179 1.79 e-5	72/10/ 178.821 0	5/11/ 160.765 0
13/400	7/7/ 9.237 2.04 e-9	11/11/ 15.372 5.55 e-5	10/10/ 13.806 4.09 e-5	17/18/ 30.827 1.71 e-8	2/5/ 5.461 1.16 e-41
14/500	16/16/ 34.458 1.75 e-8	14/14/ 29.448 3.63 e-6	17/17/ 35.988 1.20 e-6	10/10/ 29.087 7.58 e-25	5/11/ 20.243 9.16 e-19

TABLE 3. Numerical results.

No./Dim	Algorithm 2 $\beta = \theta = 1$ Proposed N_f/N_g/CPU f	Algorithm 2 $\beta = \theta = 2$ (Halley) N_f/N_g/CPU f	Algorithm 2 $\beta = 1, \theta = 0.75$ Proposed N_f/N_g/CPU f	Improved Cholesky method N_f/N_g/CPU f	RN method [7] N_f/N_g/CPU f
15/50	16/16/ 4.149 6.87 e-5	25/25/ 6.643 5.15 e-4	45/45/ 12.078 4.41 e-5	59/48/ 18.162 2.43 e-5	FAIL -
16/500	4/4/ 7.488 8.92 e-6	FAIL -	7/7/ 14.789 3.27 e-4	16/17/ 44.758 2.67 e-9	6/13/ 27.645 2.61 e-9
17/500	6/6/ 11.029 1.47 e-8	10/10/ 19.780 4.12 e-6	8/8/ 15.386 8.60 e-7	5/6/ 47.632 -5.45 e-12	3/7/ 45.774 2.75 e-11
18/500	5/5/ 13.160 7.93 e-11	9/9/ 26.520 1.37 e-5	7/7/ 20.416 3.77 e-6	6/7/ 23.988 3.34 e-17	3/7/ 16.885 6.24 e-22
19/500	7/7/ 11.912 4.01 e-8	15/15/ 28.274 4.94 e-5	15/15/ 29.336 6.16 e-6	30/4/ 15.119 0	20/41/ 80.026 2.06 e-17
20/50	15/15/ 89.994 1.005 e-5	18/18/ 109.089 3.84 e-5	16/16/ 95.747 4.95 e-6	143/26/ 1068.135 1.13 e-5	51/103/ 535.304 1.64 e-12
21/500	6/6/ 10.798 2.09 e-11	11/11/ 21.869 2.26 e-5	13/13/ 26.277 3.34 e-6	6/7/ 15.867 7.03 e-15	8/17/ 32.247 4.54 e+4
22/500	9/9/ 17.792 2.17 e-8	13/13/ 26.894 6.76 e-6	12/12/ 24.488 3.54 e-6	39/12/ 41.977 5.88 e-22	8/17/ 33.078 7.51 e-24
23/500	19/19/ 836.167 -5.15 e-12	11/11/ 472.175 2.99 e-11	26/26/ 1165.607 3.39 e-9	FAIL -	FAIL -
24/500	7/7/ 12.783 -2.15 E-14	6/6/ 10.427 -9.52 E-15	21/21/ 43.176 2.55 E-9	FAIL -	13/27/ 52.795 -4.96 e+2
25/500	5/5/ 8.443 -2.81 e-14	5/5/ 8.310 7.92 e-14	19/19/ 38.897 2.46 e-9	FAIL -	8/17/ 32.303 -4.99 e+2
26/500	8/8/ 15.392 -3.18 e-12	FAIL -	6/6/ 10.872 -1.00 E-11	22/22/ 46.026 -3.96 e-9	11/10/ 22.604 -5.87 e+3
27/500	9/9/ 16.959 2.98 e-8	10/10/ 19.169 7.52 e-6	9/9/ 16.919 1.29 e-5	6/7/ 14.333 5.88 e-17	3/7/ 11.670 3.57 e-18
28/500	5/5/ 17.327 7.93 e-11	9/9/ 26.072 1.37 e-5	7/7/ 19.632 3.77 e-6	6/7/ 23.352 3.34 e-17	3/7/ 16.740 6.24 e-22
29/100	14/14/ 6.036 3.69 e-13	53/53/ 24.900 4.77 E-6	31/31/ 13.913 3.33 e-7	6/7/ 3.824 2.45 e+3	12/25/ 8.915 2.45 e+3
30/500	3/3/ 5.541 1.09 e-13	2/2/ 2.758 NaN	3/3/ 5.337 NaN	FAIL -	2/5/ 9.629 NaN

Recently, to compare iterative algorithms, Dolan and More' [8], proposed a new technique comparing the considered algorithms with statistical process by demonstrating performance profiles. In this process, it is known that the plot of the performance profile reveals all of the major performance characteristics, which is a common tool to graphically compare effectiveness as well as robustness of the algorithms. In this technique, one can choose a performance index as a measure of comparison among considered algorithms and can illustrate the results with performance profile. We use the three measures N_f, N_g and $CPUtime$ to compare these algorithms. Hence, we use these three indices for all of the presented algorithms separately. Figures 1, 2 and 3 show the performance of the mentioned algorithms relative to these metrics, respectively.

FIGURE 1. Performance profile for the number of the objective function evaluations.



4.1. **Systems of nonlinear equations.** In this part, we solve systems of nonlinear equations by using the proposed algorithm. Consider the following nonlinear system of equations

$$F(x) = 0, \tag{4.1}$$

where $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$ and $x \in \mathbb{R}^n$. This system can be extended as

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0, \\ f_2(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0. \end{aligned}$$

For solving (4.1) by proposed algorithm, we suppose that $f(x) = \sum_{i=1}^{n} f_i^2(x)$. Here, we have worked out our proposed method on the following test problems. In all problems,

FIGURE 2. Performance profile for the number of gradient evaluations.

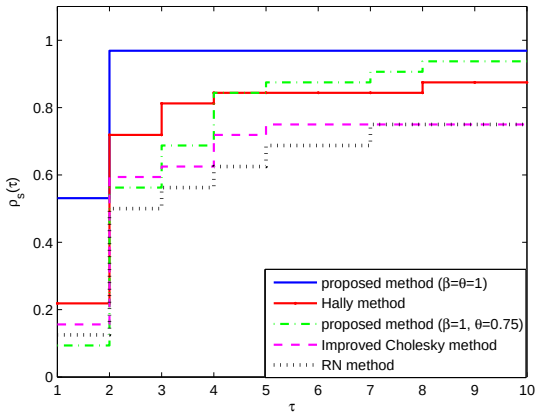
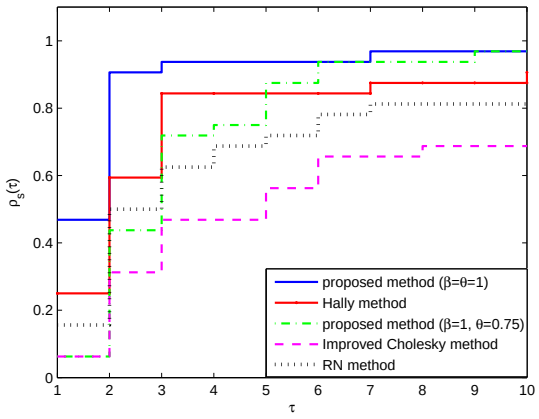


FIGURE 3. Performance profile for CPU time.



the stopping criterion is given by $\|f(x_k)\| < 10^{-8}$.
The numerical results of Examples 1,2 and 3 are given in Tables 3, 4 and 5, respectively.

Example 1. [14]:

$$F(X) = \begin{cases} x_1^3 - 3x_1x_2^2 - 1 = 0, \\ 3x_2x_1^2 - x_2^3 + 1 = 0, \end{cases}$$

$X_1^* = (-0.290514555507, 1.0842150814913)$,
$X_2^* = (1.0842150814913, -0.290514555507)$.

TABLE 4. Numerical results for Example 1.

	Algorithm 2 ($\beta = \theta = 1$)	Algorithm 2 ($\beta = 1, \theta = 0.75$)	Algorithm 2 ($\beta = \theta = 3$)	LM method [15]	Newton method
X_0	N_f N_g CPU time(s) E_k	N_f N_g CPU time(s) E_k	N_f N_g CPU time(s) E_k	N_f N_g CPU time(s) E_k	N_f N_g CPU time(s) E_k
$(1, -0.5)$	3 4 0.13 (. . ., 3.32 e-6)	7 8 0.25 (. . ., 1.46 e-5)	10 11 0.31 (. . ., 2.42 e-5)	16 15 0.51 (. . ., 2.24 e-5)	20 21 0.38 (. . ., 1.44 e-9)
$(0.5, 0)$	5 6 0.20 ($\cdots$, 2.59 e-6)	10 11 0.31 ($\cdots$, 1.31 e-5)	11 12 0.32 ($\cdots$, 2.46 e-5)	16 14 0.51 (. . . , 2.38 e-5)	FAIL - - -
$(1, 0)$	4 5 0.16 (. . . , 1.98 e-9)	8 9 0.24 (. . . , 7.37 e-6)	11 12 0.33 (. . . , 1.22 e-5)	16 15 0.52 (. . . , 2.30 e-5)	40 41 0.70 (. . . , 2.30 e-9)
$(0, 1)$	4 5 0.17 (1.98 e-9, . . .)	8 9 0.24 (7.37 e-6, . . .)	11 12 0.34 (1.22 e-5, . . .)	16 15 0.51 (2.30 e-5, . . .)	40 41 0.51 (2.30 e-5, . . .)

In this example, we define $E_k := (\|X_k - X_1^*\|, \|X_k - X_2^*\|)$.

Example 2. [5]:

$$F(X) = \begin{cases} 3x_1 - \cos(x_2 x_3) - .5 = 0, \\ x_1^2 - 81(x_2 + .1)^2 + \sin x_3 + 1.06 = 0, \\ e^{-x_2 x_3} + 20x_3 + \frac{10\pi - 3}{3} = 0, \end{cases}$$

$X^* = (0.5, 0, -0.5)$.

Example 3. [17]:

$$F(X) = \begin{cases} (x_1 - 5x_2)^2 + 40\sin^2(10x_3) = 0, \\ (x_2 - 2x_3)^2 + 40\sin^2(10x_1) = 0, \\ (3x_1 + x_2)^2 + 40\sin^2(10x_2) = 0, \end{cases}$$

$X^* = (0, 0, 0)$.

Also, consider the following systems of nonlinear equations in Table 6.
Example 4. Freudenstein and Roth function [13]:

TABLE 5. Numerical results for Example 2.

	Algorithm 2 ($\beta = \theta = 1$)	Algorithm 2 ($\beta = 1, \theta = 0.75$)	Algorithm 2 ($\beta = \theta = 3$)	LM method [15]	Newton method
X_0	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$
$(0.5, 0.1, -0.4)$	4 5 0.21 2.36 e-2	8 9 0.35 2.36 e-2	12 13 0.48 2.36 e-2	162 87 3.96 2.36 e-2	26 27 0.52 2.36 e-2
$(0.3, 0, -0.2)$	3 4 0.17 2.36 e-2	8 9 0.32 2.36 e-2	13 14 0.49 2.36 e-2	172 95 4.22 2.36 e-2	5 6 0.17 2.36 e-2
$(0.7, 0, 0)$	4 5 0.20 2.36 e-2	9 10 0.37 2.36 e-2	13 14 0.48 2.36 e-2	192 105 4.72 2.36 e-2	5 6 0.15 2.36 e-2
$(1, 2, 1)$	6 7 0.26 2.36 e-2	10 11 0.38 2.36 e-2	29 30 1.00 2.02 e-1	186 107 4.69 2.36 e-2	398 399 6.82 2.36 e-2

TABLE 6. Numerical results for Example 3.

	Algorithm 2 ($\beta = \theta = 1$)	Algorithm 2 ($\beta = 1, \theta = 0.75$)	Algorithm 2 ($\beta = \theta = 3$)	LM method [15]	Newton method
X_0	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$	N_f N_g CPU time(s) $\|X_k - X^*\|$
$(0, 0.1, -0.1)$	3 4 0.20 1.08 e-4	7 8 0.33 4.87 e-5	16 17 0.68 1.79 e-4	37 36 1.45 1.97 e-4	FAIL - - -
$(0.01, 0, -0.02)$	2 3 0.16 9.69 e-9	4 5 0.24 9.53 e-5	12 13 0.53 1.71 e-4	31 30 1.19 1.88 e-4	FAIL - - -
$(0.1, 0.1, 0.1)$	3 4 0.19 1.22 e-4	7 8 0.36 5.93 e-5	17 18 0.72 1.45 e-4	40 39 1.54 2.07 e-4	FAIL - - -

$$F(X) = \begin{cases} -13 + x_1 + ((5 - x_2)x_2 - 2)x_2 = 0, \\ -29 + x_1 + ((x_2 + 1)x_2 - 14)x_2 = 0, \end{cases}$$

$X_0 = (0.5, -2)$, $X^* = (5, 4)$.

Example 5. [23]:

$$F(X) = \begin{cases} x_1 + 1 - e^{x_2} = 0, \\ x_1 + \cos x_2 - 2 = 0, \end{cases}$$

$X_0 = (1.5, 1.2)$, $X^* = (1.340191857555588340..., 0.850232916416951327...)$.

Example 6. [6]:

$$F(X) = \begin{cases} (x_1 - 1)^4 e^{x_2} = 0, \\ (x_2 - 2)^2 (x_1 x_2 - 1) = 0, \\ (x_3 + 4)^6 = 0, \end{cases}$$

$X_0 = (1, 2, 0)$, $X^* = (1, 2, -4)$.

Example 7. Wood function [13]:

$$F(X) = \begin{cases} 10(x_2 - x_1^2) = 0, \\ 1 - x_1 = 0, \\ 90^{1/2}(x_4 - x_3^2) = 0, \\ 1 - x_3 = 0, \\ 10^{1/2}(x_2 + x_4 - 2) = 0, \\ 10^{-1/2}(x_2 - x_4) = 0, \end{cases}$$

$X_0 = (-3, -1, -3, -1)$, $X^* = (1, 1, 1, 1)$.

Example 8. [23]:

$$F(X) = \begin{cases} x_1^2 + x_2^2 + x_2^3 - x_3 - x_3^2 = 0, \\ 2x_1 + x_2^2 - x_3 = 0, \\ 1 + x_1 - x_2 x_3 = 0, \end{cases}$$

$X_0 = (-1.3, -0.8, -2.4)$,

$X^* = (-0.717018454826653767, -0.203181240635058422, -1.392754293107306018)$.

Example 9. $n = 16$, $1 \le i \le n - 1$ [23]:

$$F(X) = \begin{cases} x_i \sin(x_{i+1}) - 1 = 0, \\ x_n \sin(x_1) - 1 = 0, \end{cases}$$

$X_0 = (-0.85, \ldots, -0.85)$.

$X^* = (-1.114157140871930087, \ldots, -1.114157140871930087)$.

TABLE 7. Numerical results for Examples 4-9.

	Algorithm 2 ($\beta=\theta=1$)	Algorithm 2 ($\beta=1,\theta=0.75$)	Algorithm 2 ($\beta=\theta=3$)	LM method [15]	Newton method
Example	N_f N_g $\|X_k-X^*\|$	N_f N_g $\|X_k-X^*\|$	N_f N_g $\|X_k-X^*\|$	N_f N_g $\|X_k-X^*\|$	N_f N_g $\|X_k-X^*\|$
4	13 14 0.36 1.08 e-8	30 31 0.74 3.19 e-5	102 103 2.33 6.58 e-5	2719 755 43.91 8.29 e-5	143 144 2.31 4.92 e-9
5	4 5 0.16 2.22 e-8	7 8 0.21 5.80 e-5	11 12 0.28 5.90 e-5	199 142 4.25 1.75 e-4	16 17 0.23 4.61e-9
6	1 2 0.08 0	FAIL - - -	FAIL - - -	FAIL - - -	FAIL - - -
7	55 56 1.23 3.36 e-6	34 35 0.80 8.56 e-5	39 40 0.91 7.09 e-5	29304 6115 336.35 1.66 e-4	FAIL - - -
8	7 8 0.24 6.10 e-7	9 10 0.31 3.08 e-5	12 13 0.37 3.77 e-5	95 78 2.02 1.22 e-4	28 29 0.39 4.39 e-9
9	2 3 0.23 1.54 e-5	8 9 0.51 2.08 e-5	11 12 0.64 4.34 e-5	27 26 1.31 7.01 e-5	7 8 0.29 1.76 e-9

5. Conclusions

In this paper, we proposed a regularized Newton method for unconstrained minimization problems, and analyzed its global convergence. Convex and nonconvex problems can be solved using the presented algorithm. We also tested our algorithm on some unconstrained problems with small and medium dimensions. This algorithm does not require to calculate the step length at each iteration, and we keep $\alpha_k = 1$ as a constant. The numerical results and comparisons with some algorithms confirm the efficiency and robustness of our algorithm. The obtained numerical results showed that our proposed method in most cases was faster and more accurate than the other three methods (Improved Cholesky factorization [15], regularized Newton (RN) [7] and Halley method [12]). Moreover, we solved several nonlinear equations by Algorithm 2 with different parameters θ and β. By comparing the results of the proposed

algorithm and the other two methods (Newton method and Levenberg-Marquardt (LM) algorithm), the performance of Algorithm 2 in solving nonlinear equations systems is also shown.

Acknowledgment

The authors are grateful for the valuable comments and suggestions of referees, which improved this paper.

References

[1] N. Andrei, *Test functions for unconstrained optimization*, 8-10, Averescu Avenue, Sector 1, Bucharest, Romania. Academy of Romanian Scientists, 2004.
[2] N. Andrei, *An unconstrained optimization test functions collection*, Adv. Model. Optim., *10* (2008), 147–161.
[3] M. Aslam Noor, K. Inayat Noor, S. T. Mohyud-Din, and A. Shabbir, *An iterative method with cubic convergence for nonlinear equations*, Appl. Math. Comput., *183* (2006), 1249–1255.
[4] K. E. Atkinson, *An Introduction to Numerical Analysis*, 2nd ed., John Wiley and Sons, Singapore, 1988.
[5] R. L. Burden and J. D. Faires, *Numerical analysis (Seventh Edition)*. Thomson Learning, Inc. Aug., 2001.
[6] J. E. Dennis and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equation*, 1996.
[7] T. Dehghan Niri, M. M. Hosseini, and M. Heydari, *On The Convergence of an Efficient Algorithm For Solving Unconstrained Optimization Problems*, SAUSSUREA, *6* (2016), 342–359.
[8] E. Dolan and J. J. More , *Benchmarking optimization software with performance profiles*, Math. Program., *91* (2002), 201–213.
[9] W. J. Duncan, *Some devices for the solution of large sets of simultaneous linear equations (with an appendix on the reciprocation ofp artitioned matrices)*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science., *35* (1944), 660–670.
[10] C. T. Kelley, *Iterative methods for linear and nonlinear equations*, North Carolina State University, 1995.
[11] D. H. Li, M. Fukushima, L. Qi, and N. Yamashita, *Regularized Newton methods for convex minimization problems with singular solutions*, Comput. Optim. Appl., *28* (2004), 131–147.
[12] Y. Levina and Adi Ben-Israelb, *Directional Halley and Quasi-Halley Methods in n Variables*, Inherently Parallel Algorithms in Feasibility and Optimization and their Applications., *8* (2001), 345–367.
[13] J. J. More, B. S. Grabow and K. E. Hillstrom, *testing unconstrained optimization software*, ACM, Trans. Math. software, *7* (1981), 17–41.
[14] G. H. Nedzhibov, *A family of multi-point iterative methods for solving systems of nonlinear equations*, J. Comput. Appl. Math., *222* (2008), 244–250.
[15] J. Nocedal and S. Wright, *Numerical optimization*, 2nd edn. Springer, New York, 2006.
[16] R. A. Polyak, *Regularized Newton method for unconstrained convex optimization*, Math. Program., *120* (2009), 125–145.
[17] Z. Sun and K. Zhang, *Solving nonlinear systems of equations based on social cognitive optimization*, Comput. Eng. Appl., *44* (2008), 42–46.
[18] F. A. Shah and M. Aslam Noor,*Higher order iterative schemes for nonlinear equations using decomposition technique*, Appl. Math. Comput., *266* (2015), 414–423.
[19] Ch. Shen, Ch. Xiongda, and y. Liang, *A regularized Newton method for degenerate unconstrained optimization problems*, Optim. Lett., *6* (2012), 1913–1933.
[20] J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice Hall, Englewood, 1964.
[21] K. Ueda and N. Yamashita, *Convergence properties of the regularized Newton method for the unconstrained nonconvex optimization*, Appl. Math. Optim., *62* (2010), 27–46.

[22] M. A. Woodbury, *Inverting modified matrices*, Memorandum Report 42, Statistical Research Group, Princeton NJ, 1950.
[23] X. Xiao and H. Yin, *A new class of methods with higher order of convergence for solving systems of nonlinear equations*, Appl. Math. Comput., *264* (2015), 300–309.
[24] G. Zhou and K. C. Toh, *Superlinear convergence of a Newton-type algorithm for monotone equations*, J. Optim. Theory Appl., *125* (2005), 205–221.