American Journal of Operations Research, 2011, 1, 73-83
doi:10.4236/ajor.2011.13011 Published Online September 2011 (http://www.SciRP.org/journal/ajor)

Higher Order Iteration Schemes for Unconstrained Optimization

Yangyang Shi[1], Pingqi Pan[2]
[1]*Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands*
[2]*Department of Mathematics, Southeast University, Nanjing, China*
E-mail: *shiyang*1983@gmail.com, *panpq@seu.edu.cn*
Received August 3, 2011; *revised August* 20, 2011; *accepted September* 19, 2011

Abstract

Using a predictor-corrector tactic, this paper derives new iteration schemes for unconstrained optimization. It yields a point (predictor) by some line search from the current point; then with the two points it constructs a quadratic interpolation curve to approximate some ODE trajectory; it finally determines a new point (corrector) by searching along the quadratic curve. In particular, this paper gives a global convergence analysis for schemes associated with the quasi-Newton updates. In our computational experiments, the new schemes using DFP and BFGS updates outperformed their conventional counterparts on a set of standard test problems.

Keywords: Unconstrained Optimization, Iteration Scheme, ODE Method, Quasi-Newton Update, Convergence Analysis

1. Introduction

Consider the unconstrained optimization problem

$$\min f(x)\, x \in R^n, \tag{1}$$

where $f: R^n \to R$ is twice continuously differentiable.

Let x_k be the *k*-th iteration point. We will denote values of $f(x)$ and its gradient at x_k by f_k and ∇f_k, respectively.

Optimization problems are usually solved by iteration methods. The line search widely used in unconstrained optimization is a kind of iteration scheme for updating iterates. Such a scheme, by which one obtains the next iterate x_{k+1} from a current iterate x_k, is of the following form:

$$x_{k+1} = x_k + \alpha p_k, \tag{2}$$

where p_k and α are termed search direction and stepsize, respectively. p_k is usually determined as a descent direction with respect to the objective $f(x)$, and α by exact or inexact line searches, so that the objective value decreases after the iteration.

For instance, the famous Newton method uses the scheme with search direction

$$p_k = -\left(\nabla^2 f_k\right)^{-1} \nabla f_k,$$

where $\nabla^2 f_k$ is the Hessian matrix of $f(x)$ at x_k, and stepsize $\alpha = 1$.

The quasi-Newton methods are reliable and efficient in solving the unconstrained optimization problems. Saving explicit calculations of the second order derivatives and solution of a system of linear equations, quasi-Newton methods achieved a great degree of popularity since the first paper of Davidon [1,2]. He used

$$p_k = -H_k \nabla f_k,$$

where H_k is some approximation to the inverse Hessian matrix $\left(\nabla^2 f_k\right)^{-1}$.

The next approximate inverse Hessian matrix H_{k+1}, is obtained by updating H_k by rank-one or rank-two matrix. To this end, all quasi-Newton updates require H_{k+1} for satisfying the so-called quasi-Newton equation:

$$H_{k+1} y_k = s_k, \tag{3}$$

where $y_k = \nabla f_{k+1} - \nabla f_k$ and $s_k = x_{k+1} - x_k$.

Various quasi-Newton updates were proposed in the past. The important modification of Davidon's work by Fletcher and Powell [3] (the DFP algorithm) was the first and successful one. It was then surpassed by the BFGS update (as accepted as the best quasi-Newton method) [4-8] proposed independently by Broyden, Fletcher, Goldfarb and Shanno. These updates theoretically guarantee all H_k to be positive definite; therefore, the asso-

ciated p_k is a descent direction, and the objective decreases if α is determined by some line search.

There are other iteration schemes that appear differently from the conventional ones. The so-called ODE methods use the following initial value problem:

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = p(x) \\ \mathrm{d}(0) = x_0 \end{cases} \tag{4}$$

Assume that $p(x)$ satisfies certain conditions, and hence the preceding defines a trajectory.

Arrow, Huwicz and Uzawa [9] used $p(x) = -\nabla f(x)$ and $p(x) = -\left(\nabla^2 f(x)\right)^{-1} \nabla f(x)$. The associated trajectories might be called steepest descent curve and Newton curve respectively [10]. In this way, in fact, one could obtain many curves corresponding to existing unconstrained optimization methods.

Pan [11-13] generalized the steepest descent curve and Newton curve by setting $p(x) = -\phi(x)A(x)$, where $\phi(x)$ is called ratio factor and $A(x)$ direction matrix. He suggested some concrete ratio factors and direction matrices, and showed that under certain conditions, the objective value decreases strictly along the associated trajectory, the limit point of which is just an optimum.

ODE methods treat the optimization problem in the view of trajectory. They use numerical methods to approximately calculate associated trajectory, and finally approach the limit point of the trajectory. When Euler's approach is applied in the ODE method, standard iteration schemes are obtained. In fact the standard iteration schemes are originally derived in the direction of decreasing the objective function value instead of trajectory. Euler's approach is only of the first order precision. So it is possible to apply higher order approach to mimic the trajectory to get higher order iteration scheme than the standard one.

In this paper, we derive new iteration schemes along this line. In view of the importance of DFP and BFGS methods, we will focus on iteration schemes with respect to these methods.

The paper is organized as follows. Section 2 derives new iteration schemes. Section 3 offers the convergence analysis. Section 4 reports encouraging computational results with a set of problems.

2. Higher Order Iteration Scheme

Assume that x_k is the current iterate. The next iterate x_{k+1} will be determined by approximately following the trajectory, defined by (4). Let $\tilde{x}_{k+1}$ be a predictor. Introduce notation

$$p_k = p_k(x_k), \quad \tilde{p}_{k+1} = p(\tilde{x}_{k+1}).$$

We construct a quadratic interpolation curve, locally approximating the trajectory as follows:

$$x(t) = a_k t^2 + b_k t + c_k, \tag{5}$$

where a_k, b_k, c_k satisfy the following conditions:

$$a_k t_k^2 + b_k t_k + c_k = x_k, \tag{6a}$$

$$2a_k t_k + b_k = p_k, \tag{6b}$$

$$a_k \tilde{t}_{k+1}^2 + b_k \tilde{t}_{k+1} + c_k = \tilde{x}_{k+1}, \tag{6c}$$

$$2a_k \tilde{t}_{k+1} + b_k = \tilde{p}_{k+1}. \tag{6d}$$

Set $\tilde{t}_{k+1} = 0$, so $x(0) = \tilde{x}_{k+1}$. From (6a)-(6d), it is easily to draw that $b_k = \tilde{p}_{k+1}$, $c_k = \tilde{x}_{k+1}$ and

$$\frac{\tilde{p}_{k+1} + p_k}{2} t_k = x_k - \tilde{x}_{k+1}, \tag{7a}$$

$$a_k t_k = \frac{p_k - \tilde{p}_{k+1}}{2}. \tag{7b}$$

Pre-multiplying the both sides of (7a) by $(x_k - \tilde{x}_{k+1})^T$, we obtain an approximate t_{k-1}, furthermore, have an approximate solution of (6a)-(6d).

$$a_k = \frac{(p_k - \tilde{p}_{k+1})(x_k - \tilde{x}_{k+1})^T (p_k + \tilde{p}_{k+1})}{4\left\|(x_k - \tilde{x}_{k+1})\right\|^2}, \tag{8}$$

$$b_k = \tilde{p}_{k+1}, \; c_k = \tilde{x}_{k+1}, \tag{9}$$

where $\left\|(x_k - \tilde{x}_{k+1})\right\|$ denotes 2-norm of the vector $(x_k - \tilde{x}_{k+1})$.

The unconstrained optimization problem (1) can get a approximate solution by solving the following one-dimension minimization problem:

$$\min \varphi(t) = f\left(a_k t^2 + b_k t + c_k\right), \; t \geq 0. \tag{10}$$

To solve such problem, we apply the inexact line search rule, furthermore, we modify the sufficient decent condition

$$f(x_k + tp_k) \leq f(x_k) + \rho t p_k^T \nabla f_k, \tag{11}$$

in this way:

$$f\left(a_k t^2 + b_k t + c_k\right) \leq f(c_k) + \rho t b_k^T \nabla f(c_k), \tag{12}$$

where $\rho \in (0,1)$.

2.1. Modified Inexact Line Search Algorithm

The conventional backtracking inexact line search [14] operates in this way. At the beginning we set $t = \hat{t}$. The algorithm will stop if t satisfies the sufficient decent condition. Otherwise, the algorithm will continue with $t \leftarrow \alpha t$.

A modified backtracking inexact line search algorithm

was obtained by applying the expression (12) as the sufficient decent condition in backtracking line search algorithm.

Subalgorithm 2.1 modified backtracking inexact line search

Step 0. Given $\hat{t}$, ρ, $\alpha \in (0,1)$, ε; set $t = \hat{t}$.

Step 1. Set $a_k = 0$ if $a_k^T \nabla f_k > -c b_k^T \nabla f_k$, where constant $c \geq \frac{1}{\hat{t}}$.

Step 2. If

$$f\left(a_k t^2 + b_k t + c_k\right) \leq f\left(c_k\right) + \rho t b_k^T \nabla f\left(c_k\right), \qquad (13)$$

go to step 4; otherwise, go to step 3.

Step 3. $t \leftarrow \alpha t$ and go to step 2.

Step 4. Terminate with $t_k = t$.

2.2. Higher Order Iteration Schemes

The higher order iteration schemes, firstly obtain the predictor $\tilde{x}_{k+1}$ from the current point x_k by inexact line search rule following the direction p_k. Then construct the quadratic interpolation curve by the relevant information of x_k and $\tilde{x}_{k+1}$, and calculate x_{k+1} satisfying modified inexact line search rule (12). The overall steps of the higher order iteration schemes are organized as follows.

Algorithm 2.1 Higher order iteration schemes

Step 0. Given initial point x_0, ρ, $\alpha \in (0,1)$ and ε, set $k := 0$;

Step 1. If $\|\nabla f_k\| < \varepsilon$, stop.

Step 2. compute the predictor

- Call backtracking inexact line search algorithm to obtain t_k.
- Compute $\tilde{x}_{k+1} = x_k + t_k p_k$.

Step 3. If $\left\|\nabla f\left(\tilde{x}_{k+1}\right)\right\| < \varepsilon$, stop.

Step 4. Compute a new iteration point.

- Compute a_k, b_k, c_k by (8) and (9).
- Call subalgorithm 2.1 to get $\tilde{t}_{k+1}$.
- Compute $x_{k+1} = a_k \tilde{t}_{k+1}^2 + b_k \tilde{t}_{k+1} + c_k$.

Step 5. Set $k := k+1$ and go to step 1.

2.3. An Extension of the Higher Order Iteration Scheme

The higher order iteration schemes vary with different $p(x)$. In this paper we extend the higher order iteration schemes to BFGS method, and set $p_k = -H_k \nabla f_k$. We get the predictor $\tilde{x}_{k+1}$ from x_k, satisfying the backtracking inexact line search rule, in the direction p_k. From x_k and $\tilde{x}_{k+1}$, we compute $\tilde{p}_{k+1} = -\tilde{H}_{k+1} \nabla \tilde{f}_{k+1}$. Then by searching along the curve, we obtain the new point x_{k+1}. The overall steps of the variant of the iteration schemes are organized as follows.

Algorithm 2.2 higher order iteration schemes using BFGS update

Step 0. Given initial point x_0; set $k := 0$, ρ, $\alpha \in (0,1)$, N, $H_0 = I$, ε and ε;

Step 1. Compute the predictor.

- Call backtracking inexact line search algorithm to obtain t_k.
- Compute $\tilde{x}_{k+1} = x_k - t_k H_k \nabla f_k$ and $\tilde{f}_{k+1} = f\left(\tilde{x}_{k+1}\right)$.

Step 2. If $\left\|\nabla f\left(\tilde{x}_{k+1}\right)\right\| < \varepsilon$, stop.

Step 3. Compute $\tilde{H}_{k+1}$.

- If $\tilde{y}_k^T \tilde{s}_k \leq \varepsilon$ then $\tilde{H}_{k+1} = I$; otherwise

$$\tilde{H}_{k+1} = H_k + \left(1 + \frac{\tilde{y}_k^T H_k \tilde{y}_k}{\tilde{s}_k^T \tilde{y}_k}\right) \frac{\tilde{s}_k \tilde{s}_k^T}{\tilde{s}_k^T \tilde{y}_k} - \frac{\tilde{s}_k \tilde{y}_k^T H_k + H_k \tilde{y}_k \tilde{s}_k^T}{\tilde{s}_k^T \tilde{y}_k}, \qquad (14)$$

where $\tilde{y}_k = \nabla \tilde{f}_{k+1} - \nabla f_k$, $\tilde{s}_k = -t_k H_k \nabla f_k$.

Step 4. Compute the new iteration point.

- Compute a_k, b_k, c_k by (8) and (9).
- Call subalgorithm 2.1 to get $\tilde{t}_{k+1}$.
- Compute $x_{k+1} = a_k \tilde{t}_{k+1}^2 + b_k \tilde{t}_{k+1} + c_k$.

Step 5. If $\|\nabla f_{k+1}\| < \varepsilon$, stop.

Step 6. Update H_k.

- If $k+1(modN) = 0$ or $y_k^T s_k \leq \varepsilon$ then $H_{k+1} = I$; otherwise

$$H_{k+1} = H_k + \left(1 + \frac{y_k^T H_k y_k}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k} - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{s_k^T y_k}, \qquad (15)$$

where $y_k = \nabla f_{k+1} - \nabla f_k$, $s_k = x_{k+1} - x_k$.

- If $\left(\frac{\nabla f_{k+1}^T H_{k+1} \nabla f_{k+1}}{\|\nabla f_{k+1}\| \|H_{k+1} \nabla f_{k+1}\|}\right) < \varepsilon$ or $\frac{\|\nabla f_{k+1}^T H_{k+1}\|}{\|\nabla f_{k+1}\|} < \varepsilon$, then set $H_{k+1} = I$.

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

Note:

1) In this paper, I denotes identity matrix.

2) In step 3, we adopt a strategy to make sure that the curvature condition $\tilde{y}_k^T \tilde{s}_k > \varepsilon$ is hold. So $\tilde{H}_{k+1}$ is positive and $\tilde{F}_{k+1}$ is a decent direction.

3) In step 4, we call the subalgorithm (2.1), in which we set the parameter $c = \frac{1}{\varepsilon}$.

4) In step 6, we adopt a restart technique that if $k+1$ is the integral multiple of the N or $y_k^T s_k \leq \varepsilon$, we restart with $H_{k+1} = I$. Clearly, the BFGS method is a kind of conjugate direction method, so the restart technique can reduce the accumulation of the roundoff errors.

5) We only report the variant using BFGS update. we also derive a variant of the higher order iteration schemes by using the DFP updating formula instead of the BFGS

updating formula in step 3 and step 6.

3. The Global Convergence of the Higher Order Iteration Schemes

Definition 3.1 [11] *Curve* $x(t)$ *where* $t \in (0, \beta)$, *if for any* $\beta > 0$, $t \in (0, \beta)$, $x(t)$ *is contained in the domain of* $f(x)$ *and* $f(x(t))$ *is strictly monotone decreasing in* $(0, \beta)$, *then* $x(t)$ *is a decent trajectory of* $f(x)$ *at* $x(0) = x_0$. *And if* $\lim_{t \to \beta_-} x(t)$ *is exist and equal the minimization point of* $f(x)$, *then the curve* $x(t)$ *is normal decent trajectory of* $f(x)$ *at* x_0.

Pan proved the global convergence of the ODE methods with ratio factor and direction matrix [11]. In this paper we only consider the situation that ratio factor is 1 and direction matrix is identity matrix. So we draw the theorem as follows,

Theorem 3.1 [11] *Given* $x_0 \in \mathbb{R}^n$, *assume the level set*

$$\mathcal{L} = \left\{ x : x \in \mathbb{R}^n, f(x) \le f(x_0) \right\}$$

is bounded close set, and $f(x)$ is twice continuously differentiable in the set $\mathcal{L}$ and $\nabla f(x_0) \ne 0$, then the right segment trajectory of the the ordinary equations (4) is the decent curve of $f(x)$ at x_0 and the limit point of the trajectory is the stationery point of $f(x)$. If $f(x)$ is convex, then the right segment trajectory is normal decent curve of $f(x)$ at x_0.

We use the quadratic interpolation curve (5) to approximate the trajectory. However, when $a_k^T \nabla f_k > 0$, the iteration scheme may not be decent in the local region of the predictor $\tilde{x}_{k+1}$. So we apply the strategy of step 1 in subalgorithm (2.1) to keep the iteration decent.

Theorem 3.2 *Given constant* $\hat{t}$ *and* $c \ge \frac{1}{\hat{t}}$. *In subalgorithm* (2.1), *if* b_k *is decent direction satisfying*

$$b_k^T \nabla f_k \le 0,$$

then for any $0 < t \le \frac{1}{c}$, the condition

$$\left(a_k t^2 + b_k t \right)^T \nabla f_k \le 0,$$

is hold.

Proof. If $a_k^T \nabla f_k \le 0$, clearly, the conclusion holds.

Otherwise $a_k^T \nabla f_k > 0$, from algorithm (2.1) step 1, if

$$a_k^T \nabla f_k > -c b_k^T \nabla f_k,$$

then set $a_k = 0$, so the conclusion holds. If

$$a_k^T \nabla f_k \le -c b_k^T \nabla f_k,$$

then

$$\left(a_k t^2 + b_k t \right)^T \nabla f_k \le t \left(\frac{1}{c} a_k + b_k \right)^T \nabla f_k \le 0. \qquad \square$$

Theorem 3.3 *Consider the algorithm* (2.1), *where* p_k *and* $\tilde{p}_k$ *are decent direction, and* p_k *satisfying the condition*

$$\frac{-p_k^T \nabla f_k}{\|p_k\| \|\nabla f_k\|} \ge c_1 \tag{16}$$

and

$$\|p_k\| \ge c_2 \|\nabla f_k\| \tag{17}$$

where $c_1 > 0$ and $c_2 > 0$ are constants. Suppose that $f(x)$ is bounded below and continuously differentiable in the level set

$$\mathcal{L} = \left\{ x : x \in \mathbb{R}^n, f(x) \le f(x_0) \right\}, \tag{18}$$

where x_0 is the starting point. And the gradient ∇f is Lipschitz continuous on $\mathcal{L}$; namely, there exists L such that

$$\|\nabla f(x) - \nabla f(y)\| \le L \|x - y\|. \tag{19}$$

Then for some k, $\nabla f_k = 0$ is hold, otherwise,

$$\lim_{k \to +\infty} \|\nabla f_k\| = 0. \tag{20}$$

Proof. Consider the situation that for all k, $\nabla f_k \ne 0$. Then from the algorithm (2.1), we have that

$$\tilde{f}_{k+1} \le f_k + \rho t_k p_k^T \nabla f_k, \tag{21}$$

and

$$f_{k+1} \le \tilde{f}_{k+1} + \rho \tilde{t}_{k+1} \tilde{p}_{k+1}^T \nabla \tilde{f}_{k+1} \le \tilde{f}_{k+1}, \tag{22}$$

By (21) and (22), we have

$$f_k - \tilde{f}_{k+1} \ge -\rho t_k p_k^T \nabla f_k, \tag{23}$$

and

$$\tilde{f}_{k+1} - f_{k+1} \ge 0. \tag{24}$$

With (23) and (24) , we obtain

$$f_k - f_{k+1} \ge -\rho t_k p_k^T \nabla f_k. \tag{25}$$

By summing this expression over all indices less than or equal to k, we obtain

$$f_0 - f_{k+1} \ge -\rho \sum_{j=0}^{k} t_k p_k^T \nabla f_k. \tag{26}$$

Since f is bounded below, we have that $f_0 - f_{k+1}$ is less than some positive constant, for all k. Hence, by taking limits in (26), we obtain

$$\sum_{j=0}^{\infty} -t_k p_k^T \nabla f_k < +\infty. \tag{27}$$

In standard inexact line search algorithem, we know that if the initiate trial $\hat{t}$ does not satisfy the condition (11), then $\frac{t_k}{\alpha}$ violate the condition. So

$$f\left(x_k + \frac{t_k}{\alpha} p_k\right) > f_k + \rho \frac{t_k}{\alpha} p_k^T \nabla f_k. \tag{28}$$

By the Lipschitz condition (37), we have

$$\begin{aligned} \tilde{f}_{k+1} - f_k &= f(x_k + t_k p_k) - f_k \\ &= t_k p_k^T \nabla f_k + \int_0^{t_k} \left[\nabla f(x_k + s p_k) - \nabla f_k\right]^T p_k \mathrm{d}s \\ &\le t_k p_k^T \nabla f_k + \int_0^{t_k} sL \|p_k\|^2 \mathrm{d}s \\ &= t_k p_k^T \nabla f_k + \left(\frac{1}{2}\right) L t_k^2 \|p_k\|^2 \\ &\le \rho t_k p_k^T \nabla f_k, \text{ for all } 0 < t_k \le \frac{2(\rho - 1) p_k^T \nabla f_k}{L\|p_k\|^2} \end{aligned} \tag{29}$$

It follows from (28) and (29) that

$$\frac{t_k}{\alpha} > \frac{2(\rho - 1) p_k^T \nabla f_k}{L\|p_k\|^2}. \tag{30}$$

If initiate trial $\hat{t}$ satisfy the condition (11), then $t = \hat{t}$. Furthermore, from (16) and (17), we have

$$t_k \ge \min\left\{\frac{2\alpha(\rho - 1) p_k^T \nabla f_k}{L\|p_k\|^2}, \hat{t}\right\} > 0. \tag{31}$$

It follows from (27) that

$$\sum_{j=0}^{\infty} \min\left\{\frac{2\alpha(\rho - 1)|\, p_k^T \nabla f_k\,|^2}{L\|p_k\|^2}, -\hat{t} p_k^T \nabla f_k\right\} < +\infty \tag{32}$$

It follows from (16) and (17)

$$\sum_{j=0}^{\infty} \min\left\{\frac{2\alpha(1-\rho)}{L} c_1^2 \|p_k\|^2, \hat{t} c_1 c_2 \|\nabla f_k\|^2\right\} < +\infty \tag{33}$$

By (33), we obtain that

$$\sum_{j=0}^{\infty} \|\nabla f_k\|^2 < +\infty. \tag{34}$$

This implies that

$$\lim_{k \to +\infty} \|\nabla f_k\| = 0. \tag{35}$$

The theorem (3.3) analyzes the global convergence of the iteration scheme based on ODE, similarly, we obtain the global convergence of the variant iteration scheme using BFGS update formula.

Theorem 3.4 *Consider the algorithm* (2.2), *suppose that f is bounded below in* $\mathbb{R}^n$ *and continuously differentiable in the level set*

$$\mathcal{L} = \{x | f(x) \le f(x_0)\}, \tag{36}$$

where x_0 is the starting point. And the gradient ∇f is Lipschitz continuous on $\mathcal{L}$; namely, there exists L such that

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|. \tag{37}$$

Then for some k, $\nabla f_k = 0$ is hold, otherwise,

$$\lim_{k \to +\infty} \|\nabla f_k\| = 0. \tag{38}$$

Proof. Consider the situation that for all k, $\nabla f_k \ne 0$. Then from the algorithm (2.2), we have that

$$f_{k+1} \le \tilde{f}_{k+1} - \rho \tilde{t} \left(\tilde{H}_{k+1} \nabla \tilde{f}_{k+1}\right)^T \nabla \tilde{f}_{k+1} \tag{39}$$

and

$$\tilde{f}_{k+1} \le f_k - \rho t \left(H_k \nabla f_k\right)^T \nabla f_k. \tag{40}$$

The step 3 implies that

$$\left(\tilde{H}_{k+1} \nabla \tilde{f}_{k+1}\right)^T \nabla \tilde{f}_{k+1} > 0 \tag{41}$$

By combining the condition (39), (40) and (41), we have

$$f_k - f_{k+1} \ge \rho t \left(H_k \nabla f_k\right)^T \nabla f_k. \tag{42}$$

From the step 8 in the algorithm (2.2), we have that

$$\frac{\nabla f_{k+1}^T H_{k+1} \nabla f_{k+1}}{\|\nabla f_{k+1}\| \|H_{k+1} \nabla f_{k+1}\|} \ge \varepsilon, \tag{43}$$

and

$$\frac{\|\nabla f_{k+1}^T H_{k+1}\|}{\|\nabla f_{k+1}\|} \ge \varepsilon. \tag{44}$$

From the theorem (3.3), we conclude that

$$\lim_{k \to +\infty} \|\nabla f_k\| = 0. \tag{45}$$

is hold. □

4. Computational Results

In this section, we report computational results showing that the variant iteration schemes using BFGS and DFP update formula outperformed the BFGS method and DFP method on two sets of test functions. The first set of 20 functions were from [15], and the second from [16], which can be obtained from http://www.ici.ro/camo/neculai/ansoft.htm.

4.1. Test Codes

In this section, the following four codes are tested:

- *DFP*: the DFP method.
- *BFGS*: the BFGS method.
- *HDFP*: the higher order iteration schemes using DFP update.
- *HBFGS*: the higher order iteration schemes using BFGS update.

To have the competitions fair and easy, all the codes were implemented with the same parameters: $\rho = 10^{-4}$, $\alpha = 0.5$, $N = 15$, $H_0 = I$, $\varepsilon = 10^{-6}$, $\varepsilon = 10^{-12}$ and $\hat{t} = 1$; The detail results of the BFGS and HBFGS are showed below. And the performance of DFP and HDFP are only demonstrated in the overall results table.

Compiled using Matlab 7.0.4, the four codes operated under a Windows XP system Home Edition Version 2002 on an Asus PC with Genuine Intel(R) Centrino-Duo T2300 processor 1.66 GHz and 1.00 GB memory.

4.2. Result for 20 Small Scale Functions

The first set of test problems included the 20 problems. Numerical results obtained are listed in **Table 1**, where numbers of function value computation and gradient computation are listed in columns labeled "f" and "∇f", respectively. And CUP-time required for solving each problem are listed in columns labeled 'Time' and its unit is second. "-" denotes that the algorithm does not get a correct solution in upper bound iteration number.

Table 1 serves as an comparison between the BFGS and HBFGS. It shows that the computation numbers of function value andgradient vectors of HBFGS are fewer than that of BFGS. However, the HBFGS costs 0.11 seconds more than the BFGS, because the HBFGS has to compute a_k. Although the computation of a_k is much less compare with that of function value, it affects the CPU-time, especially, for small scale problems. So the HBFGS is competitive with BFGS on the 20 small scale

Table 1. Statistics of first 20 functions.

Problem	BFGS			HBFGS		
	Time	f	∇f	Time	f	∇f
Rosenbrock	0.02	62	36	0.06	133	44
Freudenstein and Roth	0.00	29	14	0.03	48	16
Powell badly scaled	0.17	796	303	0.02	257	33
Brown badly scaled	-	-	-	-	-	-
Beale	0.03	23	16	0.03	35	18
Jennrich and Sampson	0.00	7	2	0.02	7	2
Helical vally	0.05	77	35	0.06	166	44
Bard	0.05	430	34	0.03	65	22
Guassian	0.00	6	5	0.02	20	10
Meyer	-	-	-	-	-	-
Gulf research and develop	0.05	1	2	0.00	1	2
Box three-dimensional	0.04	51	41	0.05	86	44
Powell singular	0.05	65	36	0.06	157	58
Wood	0.05	88	32	0.06	194	44
Kowalik and Osborne	0.05	45	42	0.05	71	38
Brown and Dennis	0.34	2546	263	0.20	1225	104
Biggs EXP6	0.03	21	19	0.05	31	20
Watson	0.39	79	42	0.47	163	54
Extended Rosenbrock	0.13	244	74	0.25	832	130
Broyden banded	0.09	160	45	0.19	647	112
total	1.53	4730	1041	1.64	4138	795

problems.

4.3. Result for 50 Middle Scale Functions

The second test set of 50 problems consist of 43 functions with 100 variables, 3 functions with 200 variables and 4 functions with 300 variables. The problems with "*" have 300 independent variables, and with "**" have 200 independent variables.

Table 2 shows that compared with BFGS, the computation of the function value and the gradient vectors and CPU-time of HBFGS decrease by 52.65%, 52.08% and 36.01%, respectively. In summary the HBFGS method are faster and have less computation than the BFGS method.

4.4. Result for 50 Large Scale Functions

The second test set of 50 problems consist of 46 functions with 500 variables, 4 functions with 300 variables. The problems with "*" have 300 independent variables.

Table 3 shows that the HBFGS's CPU-time, computation numbers of function value and gradient vectors are less than the BFGS by 949.10 seconds, 38808 and 3957, respectively.

4.5. Statistics of the Ratio

The **Table 4** gives overall comparison of HDFP, HBFGS and DFP, BFGS. In **Table 4**, "Time" denotes the run time ratio, "f" denotes function value computation number ratio and "∇f" denotes gradient computation number ratio.

Table 4 shows that the HDFP outperforms the DFP with the average CPU-time ratio 1.58, function computation ratio 1.67 and gradient computation ratio 1.71. And the HBFGS defeats the BFGS with the average CPU-time ratio 1.23, function computation ratio 1.57 and gradient computation ratio 1.56.

4.6. Summary of the Tests

As the tests show, although the higher order iteration schemes add the computation of a_k, it has less computation of function value and gradient vector. For large scale problems, the computation of a_k is much less than that of function value.

5. Concluding Remarks

We gave a new iteration scheme based on ODE, proved

Table 2. Statistics of middle scale 50 functions.

Problem	BFGS			HBFGS		
	Time	f	∇f	Time	f	∇f
Strictly Convex1	0.05	7	8	0.06	21	12
Strictly Convex2	0.30	202	97	0.19	147	60
Extended Freudenstein and Roth	0.08	21	12	0.13	45	18
Extended Trigonometric	0.17	237	57	0.39	388	124
Extended White and Holst	0.19	75	36	0.22	118	38
Extended Beale	0.08	20	15	0.09	45	20
Extended Penalty	0.09	85	26	0.13	141	38
Perturbed Quadratic	0.80	1729	279	0.53	1095	170
Diagonal2	0.28	112	113	0.27	97	96
Diagonal1	5.58	10756	1998	1.03	1701	342
Diagonal3	5.77	10380	2001	1.03	1758	326
Hager	0.16	138	63	0.13	107	40
Generalized Tridiagonal-1	0.47	460	104	0.44	439	90
Extended Tridiagonal-1	0.19	30	28	0.17	29	24
Extended Three Exponential Terms	0.06	12	9	0.05	33	12

Generalized Tridiagonal-2	0.67	1261	189	0.77	1340	192
Diagonal4	0.03	9	4	0.03	28	6
Diagonal5	0.03	5	6	0.03	12	8
Extended Himmelblau	0.06	19	10	0.06	56	18
Generalized PSC1	0.73	369	313	0.28	137	104
Extended PSC1	0.06	22	15	0.09	41	22
Extended Powell**	1.45	148	78	0.98	111	50
Extended Block Diagonal BD1	0.05	14	13	0.06	30	16
Extended Maratos	0.19	146	67	0.25	264	90
Extended Cliff	0.17	71	44	0.09	65	22
Quadratic Diagonal Perturbed	0.11	57	40	0.13	65	42
Extended Wood**	0.34	81	25	0.64	134	44
Scaled Quadratic SQ1**	6.61	3163	460	5.70	3205	360
Quadratic Function QF1	0.52	1015	193	0.42	784	140
Extended Quadratic Penalty QP1	0.06	41	21	0.06	45	16
Extended Quadratic Penalty QP2	0.08	49	26	0.11	82	34
A Quadratic Function QF2	0.92	2352	324	0.58	1377	186
Extended EP1	0.03	12	5	0.05	31	8
Extended Tridiagonal-2	0.11	75	40	0.11	83	38
BDQRTIC	1.70	5044	560	0.69	1619	206
TRIDIA	1.75	5288	570	0.88	2336	258
ARWHEAD	1.69	749	121	1.81	748	114
NONDIA (Shanno-78)	0.06	88	19	0.19	378	58
NONDQUAR	5.25	2170	1999	2.86	1207	1040
DQDRTIC	0.05	49	13	0.14	234	42
Extended Rosenbrock	0.13	74	38	0.16	153	52
EG2	0.05	21	12	0.08	57	24
DIXMAANA*	0.48	14	11	0.75	27	16
DIXMAANB*	0.69	19	16	0.94	31	20
Almost Perturbed Quadratic	0.80	1744	280	0.47	963	148
Tridiagonal Perturbed Quadratic	1.25	1686	267	0.83	1088	168
DIXMAANC*	0.67	21	16	0.98	40	20
DIXMAANE*	8.31	176	174	4.93	109	102
Partial Perturbed Quadratic	0.97	1618	260	0.64	1019	158
Broyden Tridiagonal	0.64	1030	174	0.66	1015	160
total	50.97	52964	11251	32.31	25078	5392

Table 3. Statistics of large scale 50 functions on BFGS and HBFGS.

Problem	BFGS			HBFGS		
	Time	f	∇f	Time	f	∇f
Strictly Convex1	1.58	7	8	2.13	22	12
Strictly Convex2	383.11	8094	2001	346.83	6694	1714
Extended Freudenstein and Roth	3.14	21	12	5.83	48	20
Extended Trigonometric	12.84	411	66	48.61	706	242
Extended White and Holst	9.36	75	36	11.09	123	40
Extended Beale	3.39	20	15	4.83	45	20
Extended Penalty	116.31	6041	580	32.44	309	17
Perturbed Quadratic	81.64	3428	398	86.69	3508	400
Diagonal2	67.25	353	354	38.95	205	194
Diagonal1	423.53	15033	2001	315.34	10818	1448
Diagonal3	418.64	16311	2001	323.20	11385	1432
Hager	33.78	626	176	58.16	1511	284
Generalized Tridiagonal-1	19.14	414	95	19.84	441	94
Extended Tridiagonal-1	7.98	35	32	6.41	30	24
Extended Three Exponential Terms	2.02	13	10	2.48	32	12
Generalized Tridiagonal-2	28.84	977	145	41.16	1365	196
Diagonal4	0.42	9	4	1.28	29	8
Diagonal5	0.91	5	6	1.27	12	8
Extended Himmelblau	1.70	19	10	4.02	62	20
Generalized PSC1	110.98	672	584	23.94	166	120
Extended PSC1	3.14	25	16	4.75	41	22
Extended Powel**	1.56	148	78	1.02	111	50
Extended Block Diagonal BD1	2.66	15	14	2.91	30	16
Extended Maratos	12.27	141	65	17.20	241	86
Extended Cliff	9.16	71	44	4.72	65	22
Quadratic Diagonal Perturbed	24.92	343	131	19.22	258	96
Extended Wood	0.36	81	25	0.66	134	44
Scaled Quadratic SQ1	6.59	3163	460	5.67	3205	360
Quadratic Function QF1	67.39	2626	344	78.47	2949	378
Extended Quadratic Penalty QP1	3.36	40	19	8.72	374	42
Extended Quadratic Penalty QP2	6.42	76	35	9.84	133	50
A Quadratic Function QF2	110.25	5324	556	103.45	4822	492
Extended EP1	0.69	12	5	1.39	31	8

Extended Tridiagonal-2	7.16	72	39	8.30	83	42
BDQRTIC	401.72	21393	2001	216.27	10251	1010
TRIDIA	403.97	23427	2001	160.38	8716	750
ARWHEAD	1.70	749	121	1.77	748	114
NONDIA (Shanno-78)	16.36	310	86	44.33	913	218
NONDQUAR	423.55	2203	2001	600.22	3070	2555
DQDRTIC	2.53	51	14	7.42	199	38
Extended Rosenbrock	7.20	74	38	10.33	153	52
EG2	11.64	486	59	5.80	280	34
DIXMAANA*	0.48	14	11	0.73	27	16
DIXMAANB*	0.67	19	16	0.95	31	20
Almost Perturbed Quadratic	71.27	3120	361	75.47	3187	360
Tridiagonal Perturbed Quadratic	89.06	3623	420	87.52	3468	392
DIXMAANC*	0.66	21	16	0.94	40	20
DIXMAANE*	8.11	176	174	4.98	109	102
Partial Perturbed Quadratic	82.22	3446	392	93.06	3728	420
Broyden Tridiagonal	40.69	1206	203	44.28	1303	208
total	3544.33	125019	18279	2995.23	86211	14322

Table 4. Statistics of the ratio.

Problem	DFP/HDFP			BFGS/HBFGS		
	Time	f	∇f	Time	f	∇f
small 20 functions	2.49	2.66	2.63	0.93	1.14	1.31
middle 50 functions	1.03	1.29	1.23	1.58	2.11	2.09
large scale 50 functions	1.23	1.07	1.27	1.18	1.45	1.28
average	1.58	1.67	1.71	1.23	1.57	1.56

the global convergence of this scheme and variant method using DFP and BFGS update formula. In particular, this iteration has a class of variant methods using different directions as the right-hand side vectors of (4). From our experiments, we can safely conclude that this iteration scheme improved the BFGS and DFP method on the test data sets.

6. References

[1] W. C. Davidon, "Variable Metric Method for Minimization," Technical Report ANLC5990 (Revised), Argonne National Laboratory, Argonne, 1959.

[2] W. C. Davidon, "Variable Metric Method for Minimization," *SIAM Journal on Optimization*, Vol. 1, No. 1, 1991, pp. 1-17. doi:10.1137/0801001

[3] R. Fletcher and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," *The Computer Journal*, Vol. 6, No. 2, 1963, pp. 163-168.

[4] C. G. Broyden, "The Convergence of a Class of Double Rank Minimization Algorithms 2. The New Algorithms," *IMA Journal of Applied Mathematics*, Vol. 6, No. 3, 1970, pp. 222-231. doi:10.1093/imamat/6.3.222

[5] R. Fletcher, "A New Approach to Variable Matric Algorithm," *The Computer Journal*, Vol. 13, No. 3, 1970, pp. 317-322. doi:10.1093/comjnl/13.3.317

[6] C. G. Broyden, "The Convergence of a Class of Double Rank Minimization Algorithms 1. General Consideration," *IMA Journal of Applied Mathematics*, Vol. 6, No. 1,

1970, pp. 76-90. doi:10.1093/imamat/6.1.76

[7] D. Goldfarb, "A Family of Variable Metric Methods Derived by Variational Means," *Mathematics of Computation*, Vol. 24, No. 109, 1970, pp. 23-26. doi:10.1090/S0025-5718-1970-0258249-6

[8] D. F. Shanno, "Conditioning of Quasi-Newton Methods for Function on Minimization," *Mathematics of Computation*, Vol. 24, No. 111, 1970, pp. 647-656. doi:10.1090/S0025-5718-1970-0274029-X

[9] K. J. Arrow, L. Hurwicz and H. Uzawa, "Studies in Linear and Nonlinear Programming," Stanford University Press, Palo Alto, 1958.

[10] F. H. Branin and S. K. Hoo, "A Method for Finding Multiple Extreme of a Function of *n* Variables," In: F. A. Lootsman, Ed., *Numerical Method for Nonlinear Optimization*, Academic Press, Cambridge, 1972.

[11] P. Q. Pan, "Differential Equation Methods for Unconstrained Optimization," *Nanjing University Journal of Computational Mathematics*, in Chinese, Vol. 4, 1982, pp. 338-349.

[12] P. Q. Pan, "New ODE Methods of Equality Constrained Optimization (1): Equations," *Journal of Computational Mathematics*, Vol. 10, No. 1, 1992, pp. 77-92.

[13] P. Q. Pan, "New ODE Methods for Equality Constrained Optimization (2): Algorithm," *Journal of Computational Mathematics*, Vol. 10, No. 2, 1992, pp. 129-146.

[14] J. Nocedal and S. J. Wright, "Numerical Optimization," Science Press, Beijing, 2006.

[15] J. J. More, B. S. Garbow and K. E. Hillstrome, "Testing Unconstrained Optimization Software," *ACM Transactions on Mathematical Software*, Vol. 7, No. 1, 1981, pp. 17-41. doi:10.1145/355934.355936

[16] N. Andrei, "Unconstrained Optimization Test Function," *Advanced Modeling and Optimization*, Vol. 10, No. 1, 2008, pp. 147-161.

