Master's Thesis

A regularized limited memory BFGS method for box-constrained minimization problems

Guidance

Professor Nobuo YAMASHITA

Kouki HAMA

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University



February 2016

Abstract

A regularized limited memory BFGS (RL-BFGS) method is one of the efficient solution methods for large-scale unconstrained minimization problems, and it adjusts a certain regularization parameter for global convergence. Some limited numerical results indicate that the RL-BFGS method is better than the well known limited memory BFGS method with line search. However, we cannot directly apply the RL-BFGS method for the constrained minimization problem.

In this paper, we propose a RL-BFGS method for solving box-constrained minimization problems. The proposed method exploits a subspace technique given by Ni and Yuan. We show that the proposed method has global convergence under usual conditions. Moreover, we report some numerical experiments that compare the proposed method with the state-of-art method L-BFGS-B, for the box-constrained minimization problems. The results show that through the proposed method is inferior to the well-coded L-BFGS-B for most of test problems, it can get optimal solutions of some problems that cannot be solved by L-BFGS-B.

Contents

1	Introduction	1
2	A regularized limited memory BFGS Update	2
3	The RL-BFGS method for box-constrained minimization problems	4
4	Convergence analysis	6
5	Numerical results	10
6	Conclusion	14
A	Details of the numerical experiments	16

1 Introduction

A regularized limited memory BFGS (RL-BFGS) method [16] is one of the efficient solution methods for the following large-scale unconstrained minimization problems.

minimize
$$f(x)$$
,
subject to $x \in \mathbb{R}^n$, (1)

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable function. The RL-BFGS is based on the limited memory BFGS (L-BFGS) method [12]. While the usual L-BFGS [12] exploits the line search technique, the RL-BFGS controls a certain regularization parameter for global convergence. Some limited numerical results indicate that the RL-BFGS method is better than the L-BFGS method with line search [16]. However, we cannot directly apply the RL-BFGS method for constrained minimization problems.

In this paper, we propose a RL-BFGS method for solving box-constrained minimization problems.

minimize
$$f(x)$$
,
subject to $l \le x \le u$. (2)

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, $l, u \in \mathbb{R}^n$ are vectors such that $l_i < u_i$ for all *i*. We observe that box-constrained minimization problems are important, not only by themselves, but also because they are used as subproblems in penalty and augmented Lagrangian method.

The proposed method exploits a subspace technique given by Ni and Yuan [11] for the global convergence. The subspace method divides a set of variables into 3 sets: an active set, a near active set and the other one . Then a search direction $d \in \mathbb{R}^n$ is given as $d_i = 0$ for *i* in the activeset, $d_i = -\frac{\partial f(x_k)}{\partial x_i}$ for *i* in the near active set, and d_i is computed by some Newton-type method for *i* in the other set [11]. The technique is very simple and it is applicable for the RL-BFGS method. We present how to apply the technique to the RL-BFGS method, in paticular how to control the regularized parameter. We show that the proposed method has global convergence under usual conditions.

The paper is organized as follows. In Section 2, we introduce the regularized limited memory BFGS update given in [16]. In Section 3, we propose that a regularized limited memory BFGS method for box-constrained minimization problems. In Section 4, we show its global convergence under some conditions. In Section 5, we discuss improvement in order to implement the code and some numerical results are proposed, and Section 6 we conclude the paper.

Notations

Throughout the paper, we use the following notations. For a vector $x \in \mathbb{R}^n$, ||x|| denotes the Euclidean norm defined by $||x|| := \sqrt{x^T x}$. We define the feasible set region of problems (2) as

$$\Omega := \{ x \in \mathbb{R}^n | l \le x \le u \}.$$

The projection $P_{\Omega} : \mathbb{R}^n \to \mathbb{R}^n$ of $x \in \mathbb{R}^n$ onto Ω can be written as $P_{\Omega}[x]$, where each is given by the projection onto Ω , which is given by

$$(P_{\Omega}[x])_i = \begin{cases} x_i \text{ if } l_i \leq x_i \leq u_i, \\ l_i \text{ if } x_i < l_i, \\ u_i \text{ if } x_i > u_i. \end{cases}$$

Moreover, the gradient of the objective function is denoted by $(g_1(x), \dots, g_n(x))^T = g(x) := \nabla f(x)$.

2 A regularized limited memory BFGS Update

In this section, we give a brief explonation about the RL-BFGS update proposed in [16]. Let x^k be the *k*-th iterative point, and let H_k be an approximation of the inverse Hessian matrix $(\nabla^2 f(x^k))^{-1}$. Recall that H_k is used to compute search direction $d^k = -H_k g(x^k)$. As the L-BFGS, the RL-BFGS method used pair of (s^k, y^k) given by

$$s^k := x^{x+1} - x^k,$$

 $y^k := g^{k+1} - g^k,$

to construct the approximate Hessian H_k . At each iteration k, we perform the following BFGS update. The matrix H_k is transformed by BFGS update

$$H_{k+1} = \left(I - \frac{s^{k}(y^{k})^{\mathrm{T}}}{(s^{k})^{\mathrm{T}}y^{k}}\right) H_{k}\left(I - \frac{y^{k}(s^{k})^{\mathrm{T}}}{(s^{k})^{\mathrm{T}}y^{k}}\right) + \frac{s^{k}(s^{k})^{\mathrm{T}}}{(s^{k})^{\mathrm{T}}y^{k}}.$$
(3)

Note that H_{k+1} is usually dense matrix even if H_k or $\nabla^2 f(x^k)$ is sparse. If, we define the following matrix

$$V_k = I - \frac{s^k (y^k)^{\mathrm{T}}}{(s^k)^{\mathrm{T}} y^k},$$

$$S_k = \frac{s^k (s^k)^{\mathrm{T}}}{(s^k)^{\mathrm{T}} y^k},$$

and pferform the BFGS update *m* times, we can get following equations :

$$H_{k+1} = V_k H_k V_k + S_k$$

= $V_k V_{k-1} H_{k-1} V_{k-1} V_k + S_k + V_k S_{k-1} V_k$
= $V_k V_{k-1} \cdots V_{k-m+1} H_{k-m+1} V_{k-m+1} \cdots V_{k-1} V_k + S_k + V_k S_{k-1} V_k + \cdots$

Therefore, we can get H_{k+1} by using the matrix H_{k-m+1} , and vectors s^{k-m+1}, \ldots, s^k and y^{k-m+1}, \ldots, y^k that should be are stored in the memory. The L-BFGS method confines H_{k-m+1} to a diagonal matrix, e.g., the identity matrix. Thus it requires a little memory. Moreover, it calculates $d^k = -H_k g^k$ with O(mn) computations [12].

The RL-BFGS method calculates a search direction d^k as an approximate solution of linear equations $(H_k^{-1} + \mu I)d = -g^k$, where H_k is the matrix generated by the L-BFGS method and μ is a regularized parameter. Note that it is not easy to solve the linear equations exactly. Therefore, the paper [16] considers $H_k^{-1} + \mu I$ as the approximate Hessian of $f(x) + \frac{\mu}{2} ||x||^2$. Then we may construct a $\bar{H}_k(\mu) = (H_k^{-1} + \mu I)^{-1}$ by the usual L-BFGS update with gradients of $f(x) + \frac{\mu}{2} ||x||^2$. In other words, we adopt the following vector $\hat{y}^k(\mu)$ instead of y^k .

$$\hat{y}^{k}(\mu) = (g^{k+1} + \mu x^{k+1}) - (g^{k} + \mu x^{k}) = y^{k} + \mu s^{k}.$$

Consequently, a search direction $d^k(\mu) = -\bar{H}_k(\mu)\nabla f(x^k)$ is constructed as follows [16].

RL-BFGS update at *k***th iteration with parameter** μ

[Step 0] Set $p \leftarrow \nabla g^k$. [Step 1] Repeat the following process for $i = k - 1, k - 2, \dots, k - t$:

$$\begin{array}{rccc} r^i & \leftarrow & \tau^i (s^i)^{\mathrm{T}} p, \\ p & \leftarrow & p - r^i (y^i + \mu s^i) \end{array}$$

where $\tau^i = ((s^i)^T (y^i + \mu s^i))^{-1}$. [Step 2] Set $q \leftarrow \hat{H}_k^{(0)}(\mu)p$. [Step 3] Repeat the following process for $i = k - t, k - t + 1, \dots, k - 1$:

$$\begin{array}{rcl} \beta & \leftarrow & \tau^i (y^i + \mu s^i)^{\mathrm{T}} q, \\ q & \leftarrow & q + (r^i - \beta) s^i. \end{array}$$

[Step 4] Return the search direction $d^k(\mu) = -q$.

Note that we do not have to store $\hat{y}^k(\mu)$ for the update due to the definition of $\hat{y}^k(\mu)$. When a regularized parameter is changed, we get $\hat{y}^k(\mu)$ from s^k and y^k immediately. Note that, there is a relationship between $d^k(\mu)$ generated by the above algorithm and μ :

$$\begin{aligned} &d^k(\mu) \quad \to \quad -H_k g^k \quad \text{if} \ \mu \to 0, \\ &d^k(\mu) \quad \to \quad -\frac{1}{\mu} g^k \quad \text{if} \ \mu \to \infty. \end{aligned}$$

These directions correspond to the steepest descent direction and the L-BFGS directions, respectively. More precisely, if μ is large (small), then $d^k(\mu)$ tends to point out the L-BFGS (steepest descent) direction.

The paper [16] shows the following property of $H_k(\mu)$, which we will use to show the global convergence of the proposed method.

Theorem 2.1. [16] Suppose that ∇f is Lipschitz continuous. Suppose also that f is strongly convex. Then there exist positive constants γ_1, γ_2 and γ_3 such that

$$\frac{\gamma_1}{1+\gamma_2\mu} \|v\|^2 \le v^{\mathrm{T}} \bar{H}_k(\mu) v \le \frac{\gamma_3}{1+\mu} \|v\|^2 \quad \forall v \in \mathbb{R}^n.$$

$$\tag{4}$$

3 The RL-BFGS method for box-constrained minimization problems

In this section, we propose a RL-BFGS method for box-constrained minimization problems (2). The proposed method exploits the subspace technique [11], and it gets a search direction without solving any subproblem. The technique first estimates an active set of the box-constraints. Then we apply the gradient method for coordinates corresponding to the estimated active set , and use the Newton type method for the other coodinages. The proposed method adopt the RL-BFGS update in Section 2 as the Newton type method.

Now we explain the details of the proposed method. Let $A^k(x^k)$ be the active sets estimated in the k-th iteration, and let $B^k(x^k)$ be the complement set of $A^k(x^k)$. In this paper we estimate $A^k(x^k)$ as in [11], that is, $A^k(x^k)$ and $B^k(x^k)$ are defined as follows.

$$A^{k}(x^{k}) = \{i : l_{i} \leq x_{i} \leq l_{i} + \epsilon \text{ or } u_{i} - \epsilon \leq x_{i}^{k} \leq u_{i}\}.$$

$$B^{k}(x^{k}) = \{1, \dots, n\} \setminus (A^{k}(x^{k})) = \{i : l_{i} + \epsilon < x_{i}^{k} < u_{i} - \epsilon\},$$
(5)

where ϵ is a threshold for the estimation, and it sholud be sufficiently small so that

$$0 < \epsilon < \min_i \frac{1}{3}(u_i - l_i),$$

which implies

$$l_i + \epsilon < u_i - \epsilon$$
 for $i = 1, \ldots, n$

The active set $A^k(x^k)$ is not necessarily the best one. Thus we further divide the set into several parts taking into account of the following Karush-Kuhn-Tucker conditions of problem (2).

$$g(x) = \sum_{i=1}^{n} \lambda_i e_i + \sum_{i=1}^{n} \rho_i e_i, \lambda_i [x_i - l_i] = 0, \rho_i [u_i - x_i] = 0, \lambda_i \ge 0, \ \rho_i \le 0 (i = 1, \dots, n),$$

where $\lambda_i, \rho_i (i = 1, ..., n)$ are Lagrange multipliers. The conditions are equivalent to

$$g_i(x)(l_i + u_i - 2x_i) \ge 0 \text{ if } x_i = l_i \text{ or } x_i = u_i$$

$$g_i(x) = 0 \text{ otherwise.}$$
(6)

Then, we divide the set $A^k(x^k)$ into the following six sets.

$$\begin{aligned}
A_{1L}^{k}(x^{k}) &= \{i : x_{i}^{k} = l_{i} \text{ and } g_{i}^{k} \ge 0\} \\
A_{2L}^{k}(x^{k}) &= \{i : l_{i} \le x_{i}^{k} \le l_{i} + \epsilon \text{ and } g_{i}^{k} < 0\} \\
A_{3L}^{k}(x^{k}) &= \{i : l_{i} < x_{i}^{k} \le l_{i} + \epsilon \text{ and } g_{i}^{k} \ge 0\} \\
A_{1U}^{k}(x^{k}) &= \{i : x_{i}^{k} = u_{i} \text{ and } g_{i}^{k} \le 0\} \\
A_{2U}^{k}(x^{k}) &= \{i : u_{i} - \epsilon \le x_{i}^{k} \le u_{i} \text{ and } g_{i}^{k} > 0\} \\
A_{3U}^{k}(x^{k}) &= \{i : u_{i} - \epsilon \le x_{i}^{k} < u_{i} \text{ and } g_{i}^{k} \le 0\}
\end{aligned}$$
(7)

The variables x_i^k for $i \in A_{1L}(x) \cup A_{1U}(x)$ are just located at border of $[l_1, u_i]$, and satisfy (6). Therefore, we may suppose that they are optimal, and hence $d_i^k(\mu)$ is set to 0. The x_i^k for $i \in A_{3L}(x) \cup A_{3U}(x)$ are close to the border, and almost satisfy (6). Thus such variables might be active at a solution, and hence we adopt the projected gradient method for them. The other variables are supposed to be inactive at a solution. Thus we use the RL-BFGS method for them.

Summing up the above discussion, we propose the following $d^{k}(\mu)$ as a search direction.

$$d_{i}^{k}(\mu) = \begin{cases} 0 & \text{if } i \in A_{1L}^{k}(x^{k}) \cup A_{1U}^{k}(x^{k}) \\ -\min\left\{\frac{2(x_{i}^{k}-l_{i})}{g_{i}^{k}}, \alpha_{k}\right\} \frac{1}{1+\mu}g_{i}^{k} & \text{if } i \in A_{3L}^{k}(x^{k}) \\ -\min\left\{\frac{2(x_{i}^{k}-u_{i})}{g_{i}^{k}}, \alpha_{k}\right\} \frac{1}{1+\mu}g_{i}^{k} & \text{if } i \in A_{3U}^{k}(x^{k}) \\ -(\bar{H}_{C_{k}}(\mu))g_{C_{k}}^{k} & \text{if } i \in C^{k} \end{cases}$$
(8)

where C_k is defined by

$$C_k = B^k(x^k) \cup A^k_{2L}(x^k) \cup A^k_{2U}(x^k),$$
(9)

and α_k is a scaling parameter. Moreover, $\bar{H}_{C_k}(\mu)$ is a $|C_k| \times |C_k|$ matrix that is a submatrix of $\bar{H}_k(\mu)$ with respect to $(i, j) \in C_k \times C_k$. We can directly use a submatrix of the $n \times n$ full matrix $\bar{H}_k(\mu)$ constituted by (s^j, y^j) (j = k - m, ..., k - 1). However, when $|C_k|$ is much smaller than n, using the full matrix might be time consuming. In this case we may construct $\bar{H}_{C_k}(\mu)g_{C_k}^k$ by the RL-BFGS update with subvectors $(s_{C^k}^j, y_{C_k}^j)$ of (s^j, y^j) .

The proposed method generates a sequence $\{x^k\}$ as $x^{k+1} = P_{\Omega}[x^k + d^k(\mu)]$. However the objective function value might increase, that is, $f(x^{k+1}) > f(x^k)$, for some μ . Thefore we need to tune μ for global convergence. To this end, we apply the idea of the trust-region method as the RL-BFGS method [16] for unconstrained minimization problems. That is to say, we use the ratio of reduction of the objective function value to that of the model function value. We define $r_k(\hat{d}^k(\mu), \mu)$ by

$$r_k(\hat{d}^k(\mu),\mu) = \frac{f(x^k) - f(x^k + \hat{d}^k(\mu))}{f(x^k) - q_k(\hat{d}^k(\mu),\mu)},$$
(10)

where $\hat{d}^k(\mu)$ is

$$\hat{d}^{k}(\mu) = P_{\Omega}[x^{k} + d^{k}(\mu)] - x^{k}.$$
(11)

and $q_k : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is a model function of f at x^k defined by

$$q_k(\hat{d}^k(\mu),\mu) = f(x^k) + \frac{1}{2}(g^k)^T \hat{d}^k(\mu).$$
(12)

Note that, when μ is large, $\hat{d}^k(\mu) = d(\mu)$ as shown in the next section. Then, when $C_k = \{1, 2, ..., n\}$, we have

$$q_k(\hat{d}^k(\mu),\mu) = f(x^k) + (g^k)^T \hat{d}^k(\mu) + \frac{1}{2} \hat{d}^k(\mu) \bar{H}_k(\mu)^{-1} \hat{d}^k(\mu),$$

which is a usual quadratic model function.

If the ratio $r_k(\hat{d}^k(\mu), \mu)$ is large, that is to say, the reduction of the objective function f is sufficiently large as compared to that of the model function, we adopt $d^k(\mu)$ and decrease the parameter μ . If the ratio $r_k(\hat{d}^k(\mu), \mu)$ is small, we need to increase μ and compare $d^k(\mu)$ again.

We now propose the following method.

The Regularized Limited Memory BFGS method

[Step 0] Choose the parameters μ_0 , μ_{\min} , ψ_1 , ψ_2 , η_1 , η_2 , *m* such that $0 < \mu_{\min} \le \mu_0$, $0 < \psi_1 \le 1 < \psi_2$, $0 < \eta_1 < \eta_2 \le$, m > 0.

Choose an initial point $x_0 \in \mathbb{R}^n$ and set k := 0.

[Step 1] If some termination condition holds, then stop. Otherwise, go to Step 2.

[Step 2] Determine $d^k(\mu)$.

[Step 2-1] Compute $d^k(\mu)$ by (8).

[Step 2-2] Compute $\hat{d}^k(\mu)$ and $r_k(\hat{d}^k(\mu), \mu)$ by (10) and (11), respectively.

[Step 2-3] If $r_k(\hat{d}^k(\mu), \mu) < \eta_1$ or $(g^k)^T \hat{d}^k(\mu) > 0$, then $\mu := \psi_2 \mu$ and go to Step 2-1. Otherwise go to Step 3.

[Step 3] If $\eta_1 \le r_k < \eta_2$, μ is not changed, if $r_k \ge \eta_2$, then $\mu := \min[\mu_{\min}, \psi_1 \mu]$. [Step 4] Set $x_{k+1} = x_k + \hat{d}^k(\mu)$ and k = k + 1 and go to Step 1.

Step 3-3 checks not only $r_k(\hat{d}^k(\mu),\mu)$ but also $(g^k)^T \hat{d}^k(\mu)$. This is because $\hat{d}^k(\mu)$ is not necessarily a descent direction of f at x^k . Note that if $(g^k)^T \hat{d}^k(\mu) > 0$, then the inequality $r_k(\hat{d}^k(\mu),\mu) > 0$ implies that $f(x^{k+1}) > f(x^k)$.

4 Convergence analysis

In this section, we show global convergence of the proposed algorithm.

We need the following assumptions to show the global convergence.

Assumption 4.1.

(i) The objective function f is twice continuously defferentiable;

(ii) There exist positive constants γ_1, γ_2 and γ_3 .

$$\frac{\gamma_1}{1 + \gamma_2 \mu} \|v\|^2 \le v^T \bar{H}_{C_k}(\mu) v \le \frac{\gamma_3}{1 + \mu} \|v\|^2 \quad \forall v \in \mathbb{R}^{|C_k|}$$
(13)

for all k.

- (iii) The feasible set Ω is bounded.
- (iv) There exist positive constants α_{\min} and α_{\max} such that $\alpha_k \in [\alpha_{\min}, \alpha_{\max}]$, where α_k is a scaling parameter used in $d^k(\mu)$.

Assumption 4.1 (ii) implies that $\bar{H}_{C_k}(\mu)$ is positive definite. Theorem 2.1 indicates that the matrix \bar{H}_{C_k} generated by the RL-BFGS update satisfies this assumption. Assumption 4.1 (iii) can be replaced with the condition that $\{x^k\}$ is bounded.

We first show the equivalence between the KKT condition (6) and $d^k(\mu) = 0$.

Lemma 4.1. Let x^k and $d^k(\mu)$ be given by the proposed method with $\mu \in (0,\infty)$. Then x^k satisfies (2) if and only if $d^k(\mu) = 0$.

Proof. We first assume that x^k satisfies (2). If $x_i = l_i$, then $g_i^k \ge 0$, and hence $i \in A_{1L}$. Therefore we have $d_i^k(\mu) = 0$. Similarly, when $x_i = u_i$, $d_i^k(\mu) = 0$. Finally, when $x_i \in (l_i, u_i)$, we have $g_i^k = 0$. From the definition of $d^k(\mu)$ we have $d^k(\mu)_i = 0$. Hence, we obtain $d^k(\mu) = 0$. Now, suppose the contrary, that is, $d^k(\mu_k) = 0$. If $i \in A_{1L} \cup A_{1U}$, then (12) holds for *i*. If

 $i \notin A_{1L} \cup A_{1U}$, then we have

$$\frac{1}{1+\mu}g_i^k = 0$$

$$\bar{H}_{C_k}(\mu)g_{C^k}^k = 0$$

by (8). Since $\bar{H}_k(\mu)$ is positive definite and $\mu \in (0, \infty)$, we have $g_i^k = 0$, that is, (6) holds.

Next we show the upper bound of the direction $d^k(\mu)$.

Lemma 4.2. There exists constants $\gamma_4, \gamma_5, \gamma_6$ and γ_7 such that

$$\begin{aligned} \|d^{k}(\mu)\| &\leq -\frac{\gamma_{4}}{1+\gamma_{5}\mu} \\ \|d^{k}(\mu)\|^{2} &\leq -\frac{\gamma_{6}}{1+\gamma_{7}\mu} (d^{k}(\mu))^{\mathrm{T}} g^{k} \end{aligned}$$

Proof. Let $D_k = A_{2L}^k(x^k) \cup A_{2U}^k(x^k) \cup A_{3L}^k(x^k) \cup A_{3U}^k(x^k)$. Then we have

$$(g^{k})^{\mathrm{T}} d^{k}(\mu) \leq -(g_{C_{k}}^{k})^{\mathrm{T}} \bar{H}_{C_{k}}(\mu) g_{C_{k}}^{k} - \frac{\tau_{1}}{1+\mu} \|g_{D_{k}}^{k}\|^{2}$$

$$\leq -\frac{\gamma_{1}}{1+\gamma_{2}\mu} \|g_{C_{k}}^{k}\|^{2} - \frac{\tau_{1}}{1+\mu} \|g_{D_{k}}^{k}\|^{2},$$
(14)

where $\tau_1 = \min{\{\epsilon, \alpha_{\min}\}}$. Moreover, we have

$$\|d^{k}(\mu)\|^{2} \leq \frac{\gamma_{3}^{2}}{(1+\mu)^{2}} \|g_{C_{k}}^{k}\|^{2} + \frac{\tau_{2}}{(1+\mu)^{2}} \|g_{D_{k}}^{k}\|^{2},$$
(15)

where $\tau_2 = \max{\{\epsilon, \alpha_{\max}\}}^2$. It then follows from (14) that there exist positive constants γ_6 and γ_7 such that

$$\|d^{k}(\mu)\|^{2} \leq -\frac{\gamma_{6}}{1+\gamma_{7}\mu} (g^{k})^{\mathrm{T}} d^{k}(\mu),$$
(16)

which is the second inequality of the lemma.

The inequality (15) yields

$$\|d^k(\mu)\|^2 \le \frac{\gamma_3^2 + \tau^2}{(1+\mu)^2} \|g^k\|^2.$$

Since Ω is bounded from Assumption 4.1 (i) and (iii), there exists C_g such as $C_g = \max_{x \in \Omega} ||g^k||^2$. Therefore there exists γ_4 and γ_5 such that

$$\|d^k(\mu)\| \le \frac{\gamma_4}{1 + \gamma_5 \mu}$$

Note that this lemme shows that $(g^k)^T d^k(\mu) < 0$. However it does not implies that $(g^k)^T \hat{d}^k(\mu) < 0$. The next lemma shows when $\hat{d}(\mu) = d(\mu)$.

Lemma 4.3. There exisits $\bar{\mu}$ such that $\hat{d}^k(\mu) = d^k(\mu)$ for all $\mu \in [\bar{\mu}, \infty)$. Then $(\hat{d}^k(\mu))^T g^k < 0$ for all $\mu \in [\bar{\mu}, \infty)$.

Proof. We consider the case when $i \in A_{3L}^k(x^k)$. We have

$$\min\left\{\frac{x_i^k - l_i}{(g^k)_i}, \alpha_k\right\} \frac{1}{1 + \mu} (g^k)_i \le x_i - l_i.$$

Therefore, we have $\hat{d}^k(\mu)_i = d^k(\mu)_i$. Similarly, when $i \in A^k_{3U}(x^k)$, we have $\hat{d}^k(\mu)_i = d^k(\mu)_i$. Next, we consider the case when $i \in C_k$. From $\sqrt{n} ||d^k(\mu)|| \ge ||d^k(\mu)||_{\infty}$ and Lemma 4.2,

Next, we consider the case when $i \in C_k$. From $\sqrt{n} ||d^k(\mu)|| \ge ||d^k(\mu)||_{\infty}$ and Lemma 4.2, we have

$$\|d^k(\mu)\|_{\infty} \le \sqrt{n} \|d^k(\mu)\| \le \frac{\sqrt{n\gamma_4}}{(1+\gamma_5\mu)}$$

Therefore, when $\mu \ge \frac{\sqrt{n\gamma_4}}{\gamma_5\varepsilon} - \frac{1}{\gamma_5}$, in other words, when $\mu \ge \frac{\sqrt{n\gamma_4}}{\gamma_5\varepsilon} - \frac{1}{\gamma_5}$, we have

$$\|d^{\kappa}(\mu)\|_{\infty} \leq \epsilon,$$

where ϵ is the threshold used in (5). Therefore, we have $\hat{d}^{k}(\mu)_{i} = d^{k}(\mu)_{i}$. Consequently, if we set $\bar{\mu} = \max\left\{0, \frac{\sqrt{n\gamma_{4}}}{\gamma_{5}\varepsilon} - \frac{1}{\gamma_{5}}\right\}$, we have $\hat{d}^{k}(\mu) = d^{k}(\mu)$ for $\mu \in [\bar{\mu}, \infty)$. Moreover, from Lemma 4.2, we obtain $0 > (g^{k})^{\mathrm{T}} d^{k}(\mu) = (g^{k})^{\mathrm{T}} \hat{d}^{k}(\mu)$. Using the lemmas, we show that the proposed method is well-defined, that is, Step 2 stops within finite iterations.

Lemma 4.4. There exists a positive constant $\hat{\mu}$ such that $(g^k)^T \hat{d}(\mu) < 0$ and $r_k(q_k(\mu), \mu) > \eta_1$ for all $\mu \in [\hat{\mu}, \infty)$. Therefore the proposed method is well-defined.

Proof. First note that

$$\|d^{k}(\mu)\|^{2} \ge \|\hat{d}^{k}(\mu)\|^{2}.$$
(17)

Note also that ∇f is Lipschitz continuous on Ω from Assumption 4.1 (i), that is, there exists L such that

$$\|\nabla f(x) - \nabla f(y)\| \le L \|x - y\| \quad \forall x, y \in \Omega.$$

It then follows from the Taylor's theorem that

$$f(x + \hat{d}_{k}(\mu)) = f(x) + \int_{0}^{1} \nabla f(x + \tau \hat{d}_{k}(\mu))^{T} \hat{d}_{k}(\mu) d\tau$$

$$= f(x) + \nabla f(x)^{T} \hat{d}_{k}(\mu) + \int_{0}^{1} (\nabla f(x_{k} + \tau \hat{d}(\mu)) - \nabla f(x_{k}))^{T} \hat{d}_{k}(\mu) d\tau$$

$$\geq f(x) + \nabla f(x_{k})^{T} \hat{d}^{k}(\mu) - \frac{L}{2} \|\hat{d}_{k}(\mu)\|^{2}.$$
 (18)

Then we have

$$\begin{split} f(x^{k}) &- f(x^{k} + \hat{d}^{k}(\mu)) - \eta_{1} \left(f(x^{k}) - q_{k}(\hat{d}^{k}(\mu)), \mu \right) \right) \\ &\geq -(g^{k})^{T} \hat{d}^{k}(\mu) - \frac{\eta_{1}}{2} (g^{k})^{T} \hat{d}^{k}(\mu) - \frac{L}{2} \| \hat{d}_{k}(\mu) \|^{2} \\ &= \frac{\eta_{1} - 2}{2} \nabla f(x_{k})^{T} \hat{d}_{k}(\mu) - \frac{L}{2} \| \hat{d}_{k}(\mu) \|^{2} \\ &\geq -\frac{2 - \eta_{1}}{2} \nabla f(x_{k})^{T} \hat{d}_{k}(\mu) - \frac{L\gamma_{4}^{2}}{(1 + \gamma_{5}\mu)^{2}} \\ &\geq \frac{2 - \eta_{1}}{2} \frac{\gamma_{6}}{1 + \gamma_{7}\mu} - \frac{L\gamma_{4}^{2}}{(1 + \gamma_{5}\mu)^{2}}, \end{split}$$

where the second and the last inequalities follow from Lemma 4.2. This inequality implies $r_k(\hat{d}(\mu), \mu) = \frac{f(x) - f(x + \hat{d}(\mu))}{f(x) - q(\hat{d}(\mu), \mu)} \ge \eta_1$ if

$$\frac{2-\eta_1}{2}\frac{\gamma_6}{1+\gamma_7\mu} - \frac{L\gamma_4^2}{(1+\gamma_5\mu)^2} \ge 0.$$
 (19)

It is easy to see that there exist $\tilde{\mu}$ such that (19) holds for $\mu \in [\tilde{\mu}, \infty]$.

Letting $\hat{\mu} = \max{\{\tilde{\mu}, \bar{\mu}\}}$ where $\bar{\mu}$ is given by Lemma 4.3, we have the lemma.

Now we show the main theorem of the paper.

Theorem 4.1. Suppose that Assumption 4.1 holds. Then any accumulation point of $\{x^k\}$ generated by the proposed method is a Karush-Kuhn-Tucker point of problem (2).

Proof. Lemma 4.4 says that the inner loops of Algorithm Step 3 must terminate when $\mu \ge \hat{\mu}$. Therfore we have $\mu \le \psi_2 \hat{\mu}$, where ψ_2 is the constant used in Step 2-3. It then follows from Lemma 4.2 that

$$\begin{aligned} f(x^{k}) - f(x^{k+1}) &= f(x^{k}) - f(x^{k} + \hat{d}^{k}(\mu)) \\ &\geq \eta_{1}(f(x^{k}) - q(\hat{d}^{k}(\mu), \mu))) \\ &= \eta_{1}(\frac{-1}{2}g_{k}^{T}\hat{d}_{k}(\mu)) \\ &\geq \frac{\eta_{1}}{2}\frac{\gamma_{6}}{1 + \gamma_{7}\mu}\|\hat{d}^{k}(\mu)\|^{2} \\ &\geq \frac{\eta_{1}}{2}\frac{\gamma_{6}}{1 + \gamma_{7}\psi_{2}\hat{\mu}}\|\hat{d}^{k}(\mu)\|^{2} \\ &= \gamma_{8}\|\hat{d}^{k}(\mu_{k})\|^{2}, \end{aligned}$$

where $\gamma_8 = \frac{\eta_1}{2} \frac{\gamma_6}{1+\gamma_7\psi_2\hat{\mu}}$ and $\mu_k \in [\mu_{\min}, \psi_2\hat{\mu}]$ is the parameter μ accepted at the *k*-th iteration. Summing up the inequalities, we have

$$f(x_0) - f(x^k) = \sum_{j=0}^{k-1} (f(x^j) - f(x^{j+1}))$$

$$\geq \gamma_8 \sum_{j=0}^{k-1} ||d^k(\mu_k)||^2.$$

Since $f(x) > -\infty$ from Assumption 4.1 (i) and (iii), we have

$$\lim_{k\to\infty}\|d^k(\mu_k)\|=0.$$

Moreover, since $\{\mu_k\}$ and $\{x^k\}$ are bounded, and since C^k is finite set, any accumulation point of $\{x^k\}$ satisfies (6) from Lemma 4.1.

5 Numerical results

In this section, some numerical results are reported. At first, we discuss some implementation issues for the proposed method. It uses the following initial matrix in each iteration:

$$\bar{H}_k^{(0)}(\mu) = \frac{\alpha_k}{1 + \alpha_k \mu} I.$$

where the scaling parameter α_k is the following:

$$\alpha_k = \frac{(s^{k-1})^{\mathrm{T}} y^{k-1}}{\|y^{k-1}\|^2}.$$

We observe that this idea is already used in [1, 2, 12, 14, 15].

We implemented our proposed algorithm with FORTRAN77 and the experiments were carried out in machine with Intel Core i7 1.7GHz of CPU and 8GB of memory. We considered 75 box-constrained problems from CUTEst collection [8]. We used the original L-BFGS-B algorithm coded by Nocedal [13] without changing the default settings. Let n_f be the number of function evaluations. In the RL-BFGS method, we set $\gamma_1 = 0.1$, $\gamma_2 = 10.0$, $\psi_1 = 0.01$, $\psi_2 = 0.9$, $\mu_{\min} = 1.0 \times 10^{-3}$, m = 5. We choose $\epsilon = \min\{\frac{1}{1000}, \min\{u_i - l_i | i = 1, \dots, n\}\}$. The program is stopped when n_f exceeds 300000, or when the following inequality is satisfied :

$$||P_{\Omega}(x_k - \nabla f(x_k)) - x_k||_{\infty} \le 10^{-5}$$

However, when n_f is larger than 10000 we consider that the proposed program failed. Moreover, for the L-BFGS-B method, it is considered as failure when the line search is failed.

We now compare the rate of success of both methods observe that the proposed one can solve more problems than L-BFGS-B method from Table 1. Moreover, from Table 2, we can

Algorithm	The rate of success (%)
L-BFGS-B method	73.3
Proposed method	72.0

Table 1: The rate of successes at L-BFGS-B method and RL-BFGS method

Table 2:	The number	of successes a	t L-BFGS-B	method and	RL-BFGS	S method
			1			

		RL-	BFGS	
		Solvable	Unsolvable	Total
L-BFGS-B	Solvable	50	5	55
	Unsolvable	4	16	20
	Total	54	21	75

see that some problems were solved only with L-BFGS-B and others were solved only with the proposed method.

Next, we compare the algorithms by using a distribution function proposed in [6] which is often called performance profile. In order to compare the CPU-times of both methods, we first define two scalars, that depend on a parameter $\tau \ge 1$ as follows. Between the number of problems that was solved by both methods, let $\zeta_1(\tau)$, $(\zeta_2(\tau))$ be the number of such that the time spent by RL-BFGS (L-BFGS-B) method is lower than or equal to τ multiplied by the minimum of the CPU times of both method. Then let us define

$$F_{\text{RLBFGS}}^{time}(\tau) = \frac{\zeta_1(\tau)}{50},$$
$$F_{\text{LBFGSB}}^{time}(\tau) = \frac{\zeta_2(\tau)}{50}.$$

Note that the denominator 50 of the above function represents the number of problems solved by both method, and note that $F_{\text{RLBFGS}}^{time}(\tau), F_{\text{LBFGSB}}^{time}(\tau)$ are close to 1 is considered to be superior than the other algorithm. These function are used to generate the performance profile of Figure 1.



Figure 1: CPU-time

Note that it shows that RL-BFGS method is faster than L-BFGS-B method, Table 1 show that a

RL-BFGS method faster than L-BFGS-B method. Now, let us define $\hat{\zeta}(\tau)$ and $\hat{\zeta}(\tau)$ as before, except that we replace the CPU time with the number of function evaluations. Thus we consider the function below

$$F_{\text{RLBFGS}}^{n_f}(\tau) = \frac{\hat{\zeta}_1(\tau)}{50},$$
$$F_{\text{LBFGSB}}^{n_f}(\tau) = \frac{\hat{\zeta}_2(\tau)}{50},$$



Figure 2: Function evaluation

which generate the performance profile of Figure 2. In this case, it shows that RL-BFGS requires more function evaluations than L-BFGS-B method.

6 Conclusion

In this paper, we proposed the regularized limited memory BFGS method for box-constrained minimization problems and we showed its global convergence of under some appropriate assumptions. Moreover, from the numerical experiments, we saw that our proposed method solves some problems which cannot solved by L-BFGS-B, and the reciprocal also holds.

Acknowledgments

First of all, I would like to express sincere appreciation to Professor Nobuo Yamashita. Although I sometimes troubled him due to my greenness, he always kindly looked after me and gave me plenty of precise advice. It is an honor to have studied under him. I am grateful to Assistant Professor Ellem Hidemi Fukuda. She was so kind that she gave me plenty of precise advice. Finally, I would like to thank all members of Yamashita Laboratory, my friends and my family for their encouraging words.

References

- J. Barzilai, J. M. Borwein . "Two-point step size gradient methods". *IMA Journal of Numer*ical Analysis, 1988, 8.1: 141-148.
- [2] J.V. Burke, A. Wiegmann, L. Xu. "Limited memory BFGS updating in a trust-region framework." SIAM Journal on Optimization, 2008.
- [3] R. H. Byrd, P. Lu, J. Nocedal, C. Zhu. "A limited memory algorithm for bound constrained optimization." SIAM Journal on Scientific Computing, 1995, 16.5: 1190-1208.
- [4] A. R. Conn, N. Gould, A. Sartenaer, P. L. Toint. "Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints." SIAM Journal on Optimization, 1993, 3.1: 164-221.
- [5] A. R. Conn, N. Gould, P. L. Toint. "Testing a class of methods for solving minimization problems with simple bounds on the variables." *Mathematics of Computation*, 1988, 50.182: 399-430.
- [6] E. D. Dolan, J. J. More, "Benchmarking optimization software with performance profiles", *Mathematical Programming 91*, 2002, 91.2: 201-213.
- [7] R. Fletcher. "Practical Methods of Optimization." John Wiley and Sons, 2013.
- [8] N. Gould, D. Orban, P. L. Toint. "CUTEr and SifDec: A constrained and unconstrained testing environment, revisited." ACM Transactions on Mathematical Software (TOMS), 2003, 29.4: 373-394.
- [9] L. Grippo, F. Lampariello, S. Lucidi. "A nonmonotone line search technique for Newton's method." SIAM Journal on Numerical Analysis, 1986, 23.4: 707-716.

- [10] D.C. Liu, J. Nocedal. "On the limited memory BFGS method for large scale optimization." *Mathematical programming*, 1989, 45.1-3: 503-528.
- [11] Q. Ni, Y. Yuan. "A subspace limited memory quasi-Newton algorithm for large-scale nonlinear bound constrained optimization." *Mathematics of Computation of the American Mathematical Society*, 1997, 66.220: 1509-1520.
- [12] J. Nocedal. "Updating quasi-Newton matrices with limited storage." *Mathematics of computation*, 1980, 35.151: 773-782.
- [13] J. Nocedal. "Software for Large-scale Bound-constrained Optimization", Available at : http://www.ece.northwestern.edu/nocedal/lbfgsb.html, Jan. 2016.
- [14] M. Raydan. "The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem." *SIAM Journal on Optimization*, 1997, 7.1: 26-33.
- [15] D. F. Shanno, K. H. Phua. "Matrix conditioning and nonlinear optimization." *Mathematical Programming*, 1978, 14.1: 149-160.
- [16] S. Sugimoto and N. Yamashita."A regularized limited memory BFGS method for unconstrained minimization problems", Technical Report 2014-001, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 2014.
- [17] W. Sun. "Nonmonotone trust region method for solving optimization problems." *Applied Mathematics and Computation*, 2004, 156.1: 159-174.

Details of the numerical experiments

0.000090.00198 0.00006 0.00009 0.00005 0.00006 0.00004 5.92275 0.00004 8.07919 0.00005 0.000110.1135 0.016480.00081 0.30166 0.00322 0.00008 0.00004 0.00005 0.00007 0.000020.00059 CPUTime (Sec.) 0.0001 0.82301 0.00027 0.0001 0.00002 Projection – g 9.61E-06 9.53E-06 6.51E-06 2.50E-06 4.16E-06 2.24E-07 1.86E-09 1.99E-09 9.89E-06 9.78E-06 3.28E-08 0.00E+00 0.00E+00).00E+00 8.68E-06 9.06E-05 2.37E-07 9.51E-06 9.82E-06 0.00E+00 1.00E-05 6.76E-07 5.25E-06 3.00E-06 2.00E-06 8.98E-06 6.56E-06 5.96E-08 2.46E-02 1.56E-02 0.00E+00 -5.49E-05 8.72E-03 2.00E+00 2.25E+06 7.38E-08 3.32E+00 9.60E-10 5.57E-03 2.39E-09 3.63E-12 5.04E-02 **RL-BFGS** ·1.03E+00 ·1.13E+00 2.44E-04 2.68E-14 2.67E+00 1.00E+00 6.81E+02 0.00E+00 -3.63E-01 6.21E+01 4.58E+01 3.28E+01 3.83E-11 1.91E+00 575 512 11008 17 178 28 49 15 1349 163 6 250 225 13 79 300001 2 25 54 23 35 26 17 n_f 0.00043 7.70973 0.00394 0.00122 0.00205 0.00092 0.06242 0.10476 0.00211 0.00329 0.0026 0.00783 0.032086.3477 32.93183 0.01123 0.00171 0.00134 0.00043 0.00078 0.00059 0.00089 0.00161 0.00283 0.00101 0.00051 76000.0 0.20585 CPUTime (Sec.) Projection - g5.52E-06 9.99E-06).00E+00 5.87E-06 2.75E-04 3.63E-06 .81E+09 0.00E+00 9.04E-06 1.94E-06 8.17E-06 3.58E-06 4.66E-06 7.35E-06 7.86E-07 3.54E-07 2.02E-08 1.41E-07 1.99E-08 2.59E-13 4.46E-06).00E+00 0.00E+00 9.80E-01 6.94E-17 2.46E-07 9.98E-06 9.69E-08 L-BFGS-B .97E-03 1.52E-02 0.00E+00 -5.52E-05 -3.63E-01 1.03E+001.86E-02 2.85E+11 2.25E+06 1.14E-08 -1.13E+00 3.32E+00 1.00E-11 5.57E-03 8.71E-11 2.39E+01 4.54E-14 4.58E+01 4.94E+003.28E+01 1.68E-21 1.98E-14 1.20E-33 2.67E+00 .00E+00 .91E+00 6.81E+02 0.00E+00 26 10519 14 20003 111 1019 23 222 113 5427 69 40 31 12 16 26 n_f 1851 2003 100 5000 10000 25 20 10 2 1999 50 63 4 2 5000 5000 CHEBYQAD CHENHARK BQPGAUSS BQPGASIM **HIMMELP1** INVERSE DECONVB HATFLDC **CVXBQP1** HATFLDB BQP1VAR CAMEL6 HATFLDA **HS3MOD BIGGSB1** LOGROS BDEXP HART6 Problem HS110 HS25 HS38 HS45 EG1 HS1 HS2 HS5 HS3 HS4

Table 3: The details of the comparison of L-BFGS-B and RLBFGS

Table 3: The details of the comparison of L-BFGS-B and RLBFGS

	CPUTime (Sec.)	0.09922	0.0178	0.00021	0.00199	124.35575	126.45333	0.04395	0.00001	0.96878	0.20863	0.00183	1.81026	0.00031	0.00206	0.00045	0.07498	0.82584	0.01142	0.00034	0.60145	0.0001	0.00606	0.04715	0.24058	0.87061	0.90979	0.00019	0.02	0.04237	0.15235	0.78296
	Projection – g	8.36E-06	1.04E-06	3.67E-09	0.00E+00	9.90E-01	5.79E-03	7.35E-06	4.99E-08	1.41E-04	9.48E-06	8.79E-06	1.06E-04	2.86E-06	9.83E-06	1.59E-06	9.99E-06	1.13E-05	8.69E-06	8.04E-06	8.50E-06	9.26E-06	5.78E-06	4.50E-06	9.68E-06	1.17E-03	5.93E-04	4.50E-06	9.65E-06	9.04E-06	9.34E-06	2.94E-05
RL-BFGS	f^*	1.14E+03	-4.57E+03	3.36E-18	-1.99E+10	-1.32E+10	-6.56E+09	3.49E-08	6.25E+00	1.18E+04	8.99E-02	3.45E+00	8.33E-02	3.65E+03	1.71E-02	6.23E-01	2.67E-02	2.27E+03	2.05E-02	4.23E+00	6.83E-05	2.42E+03	4.06E-02	6.84E+00	1.83E-04	1.34E-01	6.32E-02	8.73E+01	4.18E-02	5.63E-02	3.44E-04	1.07E+01
	n_f	1273	17	158	0	300001	300001	70	б	300001	46671	415	300001	102	654	128	17083	300001	2839	118	139864	31	1667	16704	51552	300001	300001	93	6603	14211	47248	300001
	CPUTime (Sec.)	0.31995	0.04821	0.00367	0.10606	0.28994	0.2424	0.10024	0.00053	0.002	0.05949	1.37184	0.87338	0.00222	0.06627	0.00343	0.89134	0.03409	0.03346	0.00339	0.84141	0.00531	0.02512	0.00227	0.9161	0.3273	0.89022	0.00188	0.31328	0.03957	0.87166	0.18442
	Projection – g	6.46E-06	9.92E-07	2.32E-15	0.00E+00	9.90E + 00	9.90E + 00	4.63E-06	0.00E+00	3.74E-06	5.17E-06	3.03E-05	4.56E-03	7.43E-06	7.22E-06	8.76E-06	4.74E-04	8.22E-02	2.20E-06	8.09E-06	2.59E-03	1.50E-10	7.20E-06	2.86E-05	2.71E-03	1.42E-04	4.25E-03	7.67E-09	5.77E-06	9.87E-06	4.41E-04	7.95E-05
L-BFGS-B	f^*	1.14E+03	-4.57E+03	1.34E-32	-1.99E+10	0.5507751 + 306	Infinity	6.44E-12	6.25E+00	1.18E + 04	8.99E-02	3.45E+00	1.08E-01	3.65E+03	1.71E-02	6.23E-01	2.66E-02	2.27E+03	2.04E-02	4.23E+00	3.08E-02	2.42E+03	4.06E-02	6.84E+00	6.57E-02	9.48E-02	6.13E-02	8.73E+01	2.58E-02	5.59E-02	2.65E-02	1.05E+01
	n_f	1719	15	88	7	09	61	37	Э	100	754	199058	14261	32	889	48	14843	602	432	49	14770	58	355	145	14927	5063	11258	37	4394	504	14652	2881
	n	8	5000	7	10000	10000	10000	5000	8	4	9	4	8	4	9	4	8	4	9	4	8	4	9	4	8	8	9	4	8	9	8	9
	Problem	MAXLIKA	MCCORMCK	MDHOLE	NCVXBQP1	NCVXBQP2	NCVXBQP3	NONSCOMP	OSLBQP	PALMER1	PALMER1A	PALMER1B	PALMER1E	PALMER2	PALMER2A	PALMER2B	PALMER2E	PALMER3	PALMER3A	PALMER3B	PALMER3E	PALMER4	PALMER4A	PALMER4B	PALMER4E	PALMER5A	PALMER5B	PALMER5D	PALMER5E	PALMER6A	PALMER6E	PALMER7A

Table 3: The details of the comparison of L-BFGS-B and RLBFGS

	CPUTime (Sec.)	0.16781	0.00323	0.04816	0.00389	0.0006	0.00003	9.94603	132.03308	0.00123	0.0001	0.00002	0.01454	0.37475	0.00243	0.00994
	Projection - g	9.96E-06	9.83E-06	8.69E-06	7.04E-07	1.37E-08	2.91E-07	9.49E-06	9.49E-01	0.00E+00	7.94E-06	2.95E-06	5.64E-06	4.95E-06	7.59E-06	9.21E-06
RL-BFGS	f^*	1.02E+01	7.40E-02	6.35E-03	-7.50E-01	3.99E-07	2.41E+00	5.00E-04	-2.65E+11	-1.25E+09	-7.50E-01	4.34E-11	-9.99E+04	6.49E-12	2.59E+00	4.30E-05
	n_f	50626	1343	14396	10	12	11	35539	300001	Э	14	15	71	291	1064	3416
	CPUTime (Sec.)	0.57811	0.01711	0.83396	0.00566	0.00233	0.00168	5.17058	2.80767	0.06484	0.00146	0.00049	0.01688	0.24571	0.00474	0.00682
	Projection – g	4.49E-05	8.61E-06	1.75E-04	0.00E+00	2.83E-07	2.29E-06	5.88E-03	1.38E+00	0.00E+00	7.52E-06	2.37E-20	3.09 E-06	5.37E-06	4.25E-05	3.70E-06
L-BFGS-B	f^*	1.02E+01	7.40E-02	6.34E-03	-7.50E-01	3.99E-07	2.41E+00	6.69E-03	-2.65E+11	-1.25E+09	-7.50E-01	2.81E-41	-9.99E+04	2.50E-13	2.59E+00	6.67E-13
	n_f	8292	252	14162	б	С	11	11016	3215	2	11	9	30	157	116	67
	и	8	9	8	5000	500	4	610	5000	5000	8	5	1000	6	ю	ŝ
	Problem	PALMER7E	PALMER8A	PALMER8E	PENTDI	PROBPENL	PSPDOC	QR3DLS	QRTQUAD	QUDLIN	S368	SIMBQP	SINEALI	SPECAN	WEEDS	YFIT