A regularized limited-memory BFGS method for unconstrained minimization problems

Shinji SUGIMOTO and Nobuo YAMASHITA[*]

August 5, 2014

Abstract

The limited-memory BFGS (L-BFGS) algorithm is a popular method of solving large-scale unconstrained minimization problems. Since L-BFGS conducts a line search with the Wolfe condition, it may require many function evaluations for ill-posed problems. To overcome this difficulty, we propose a method that combines L-BFGS with the regularized Newton method. The computational cost for a single iteration of the proposed method is the same as that of the original L-BFGS method. We show that the proposed method has global convergence under the usual conditions. Moreover, we present numerical results that show the robustness of the proposed method.

1 Introduction

In this paper, we consider the following unconstrained minimization problem:

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \quad x \in \mathbb{R}^n, \end{aligned} \tag{1.1}$$

where f is a smooth function from $\mathbb{R}^n$ into $\mathbb{R}$.
We focus on the large-scale case. Standard solution methods for (1.1), such as the steepest decent method, Newton's method, and the BFGS method [5, 12], are not suitable for large-scale problems. This is because the steepest decent method generally converges slowly, while Newton's method needs to compute the Hessian matrix and solve a linear equation at each iteration. Furthermore, the BFGS method requires $O(n^2)$ memory to store the approximate Hessian of f.

A popular method for large-scale problems (1.1) is the limited-memory BFGS method (L-BFGS) proposed by Nocedal [9, 11]. The L-BFGS method

[*]Graduate School of Informatics, Kyoto University, Yoshida-honmachi, Kyoto, 606-8501, Japan. E-mail: nobuo@i.kyoto-u.ac.jp

uses m vector pairs, i.e., (s_{k-i}, y_{k-i}), $i = 0, \cdots, m-1$, to compute a search direction d_k, where

$$\begin{aligned} s_k &= x_k - x_{k-1}, \\ y_k &= \nabla f(x_k) - \nabla f(x_{k-1}), \end{aligned}$$

and k denotes the iteration number. Whereas standard BFGS requires $O(n^2)$ memory, the L-BFGS algorithm needs only $O(mn)$ memory. Moreover, given $\nabla f(x_k)$, we can compute the search direction $d_k = -H^k \nabla f(x_k)$ in $O(mn)$ time. The L-BFGS method needs a line search to satisfy the Wolfe condition to guarantee global convergence. However, the line search scheme may require a large number of function evaluations for ill-posed problems. Since the main burden of the L-BFGS method is function evaluation, it is preferable to reduce the number of function evaluations as much as possible.

The trust region method (TR-method) can guarantee global convergence without a line search. It is known that the TR-method needs fewer function evaluations than the line search [2, 3, 10]. In fact, the L-BFGS method combined with the TR-method [2] produces good performance for many benchmark problems in terms of the number of function evaluations. However, the TR-method must solve the nonconvex subproblem

$$\begin{aligned} &\text{minimize} \;\; f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T H_k^{-1} d \\ &\text{subject to} \;\; \|d\| \le \Delta_k \end{aligned} \tag{1.2}$$

in each step, where Δ_k is a positive parameter called the trust region radius. It takes a considerable amount of time to solve (1.2), even if H_k is given by the L-BFGS scheme.

The regularized Newton method proposed by Ueda and Yamashita [19, 20, 21] implicitly solves (1.2). It computes the search direction d_k by solving the following linear equation:

$$(\nabla^2 f(x_k) + \mu_k I)d = -\nabla f(x_k), \tag{1.3}$$

where $\mu_k > 0$ is called the regularized parameter. If μ_k is equal to the Lagrange multiplier in the KKT condition of (1.2), then the search direction d given by (1.2) is the solution to (1.2). Note that the linear equations (1.3) are much simpler than the subproblem (1.2) of the TR-method. The regularized Newton method [19] controls the parameter μ_k instead of computing the step length to guarantee global convergence. However, since the regularized Newton method in [19] is based on Newton's method, it must compute the Hessian matrix of f. Until now, using the regularized Newton method with the L-BFGS method has not been considered.

In this paper, we propose a novel approach that combines the L-BFGS method with the regularized Newton method. We call this the regularized L-BFGS method. It might be natural to adopt a solution of the linear equation $(B_k + \mu I)d = -\nabla f(x)$ as the search direction, where B_k is the inverse of H_k

computed by the L-BFGS method. However, we cannot determine B_k explicitly. Therefore, we construct an inverse approximate matrix of $(B_k + \mu I)^{-1}$ by the L-BFGS method using $(s_k, \hat{y}_k(\mu))$, where $\hat{y}_k(\mu) = y_k + \mu s_k$. The term μs_k in $\hat{y}_k(\mu)$ plays the role of regularization. Then, the search direction d_k can be computed in $O(mn)$ time, as for the conventional L-BFGS method. We also control the regularized parameter μ_k for global convergence, as in the regularized Newton method. To this end, we use the idea of updating the trust region radius Δ_k in the TR-method. We show that the proposed algorithm ensures global convergence. Furthermore, we conduct numerical experiments using the CUTEst test environment [7].

The paper is organized as follows. In Section 2, we describe the proposed regularized L-BFGS method that controls the regularized parameter at each iteration. In Section 3, we demonstrate its global convergence under certain conditions, and in Section 4, we discuss some implementation issues for practical use. Section 5 presents a series of numerical results, and Section 6 concludes the paper.

Throughout the paper, we use the following notation. For a vector $x \in \mathbb{R}^n$, $\|x\|$ denotes the Euclidean norm defined by $\|x\| := \sqrt{x^T x}$. For a symmetric matrix $M \in \mathbb{R}^{n\times n}$, we denote the maximum and minimum eigenvalues of M as $\lambda_{\max}(M)$ and $\lambda_{\min}(M)$. Moreover, $\|M\|$ denotes the l_2 norm of M defined by $\|M\| := \sqrt{\lambda_{\max}(M^T M)}$. If M is a symmetric positive-semidefinite matrix, then $\|M\| = \lambda_{\max}(M)$. Furthermore, if the matrix $M \in \mathbb{R}^{n\times n}$ is positive-(semi)definite, the minimum eigenvalue of M is positive (non-negative), i.e., $\lambda_{\min}(M) > (\geq)\ 0$.
Next, we give a definition of Lipschitz continuity.

Definition 1.1 *Let S be a subset of $\mathbb{R}^n$ and $f : S \to \mathbb{R}$.*

(i) *The function f is said to be Lipschitz continuous on S if there exists a positive constant L_f such that*

$$|f(x) - f(y)| \leq L_f\|x - y\|, \ \forall x, y \in S.$$

(ii) *Suppose that the function f is differentiable. ∇f is said to be Lipschitz continuous on S if there exists a positive constant L_g such that*

$$|\nabla f(x) - \nabla f(y)| \leq L_g\|x - y\|, \ \forall x, y \in S.$$

The constants L_f and L_g are called Lipschitz constants.

2 The regularized L-BFGS method

In this section, we propose a regularized L-BFGS method that controls the regularized parameter at each iteration. In the following, x_k denotes the k-th iterative point, B_k denotes the approximate Hessian of $f(x_k)$, and $H_k^{-1} = B_k$.

We consider combining the L-BFGS method with the regularized Newton method (1.3). For this purpose, we may replace the Hessian $\nabla^2 f(x_k)$ in equation (1.3) with the approximate Hessian B_k. That is, we define a search direction d_k as a solution of

$$(B_k + \mu I)^{-1} d_k = -\nabla f(x_k). \tag{2.1}$$

However, since the L-BFGS method updates H_k, it is not easy to construct B_k explicitly. Furthermore, even if we obtain B_k, it takes a considerable amount of time to solve the linear equation (2.1) in large-scale cases.

Now, we may regard $B_k+\mu I$ as an approximation of $\nabla^2 f(x)+\mu I$. Since B_k is the approximate Hessian of $f(x_k)$, the matrix $B_k+\mu I$ is an approximate Hessian of $f(x)+\frac{\mu}{2}\|x\|^2$. The L-BFGS method uses the vector pair (s_k, y_k) to construct the approximate Hessian, where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Note that y_k consists of the gradients of f. Therefore, when we compute the approximate Hessian of $f(x) + \frac{\mu}{2}\|x\|^2$, we use the gradients of $f(x) + \frac{\mu}{2}\|x\|^2$. That is, we adopt the following $\hat{y}_k(\mu)$ instead of y_k:

$$\hat{y}_k(\mu) = (\nabla f(x_{k+1}) + \mu x_{k+1}) - (\nabla f(x_k) + \mu x_k) = y_k + \mu s_k.$$

Let $\hat{H}_k(\mu)$ be the matrix constructed by the L-BFGS method with vector pair $(s_k, \hat{y}_k(\mu))$ and an appropriate initial matrix $\hat{H}_k^{(0)}(\mu)$. Then, the search direction $d_k = -\hat{H}_k(\mu)\nabla f(x_k)$ is calculated in $O(mn)$ time, which is the same as for the original L-BFGS.

Note that the conditions that $s_k^T \hat{y}_k(\mu) > 0$ and $\hat{H}_k^{(0)}$ is positive-definite are necessary for the positive-definiteness of $\hat{H}_k(\mu)^{-1}$. When the condition $s_k^T \hat{y}_k(\mu) > 0$ is not satisfied, we may replace $\hat{y}_k(\mu)$ by $\tilde{y}_k(\mu)$:

$$\tilde{y}_k(\mu) = y_k + \left(\max\left\{0, \frac{-s_k^T y_k}{||s_k||^2}\right\} + \mu \right) s_k.$$

Then, the condition $s_k^T \tilde{y}_k(\mu) > 0$ is satisfied, because

$$s_k^T \tilde{y}_k(\mu) = \max\{0, s_k^T y_k\} + \mu\|s_k\|^2 > 0.$$

In the following, $\hat{H}_k(\mu)$ is the matrix constructed by the L-BFGS method using the initial matrix $\hat{H}_k^{(0)}(\mu)$ and the vector pairs $(s_{k-i}, \hat{y}_{k-i}(\mu)),\ i = 1, \cdots, m$, and the search direction is $d_k(\mu) = -\hat{H}_k(\mu)\nabla f(x_k)$. The L-BFGS method uses $\gamma_k I$ as the initial matrix $H_k^{(0)}$. Since $(B_k^{(0)})^{-1} = H_k^{(0)}$ and $\hat{H}_k^{(0)}$ is an approximation of $(B_k^{(0)} + \mu I)^{-1}$, we can set the initial matrix $\hat{H}_k^{(0)}(\mu)$ as

$$\hat{H}_k^{(0)}(\mu) = (B_k^{(0)} + \mu I)^{-1} = \left(\frac{1}{\gamma_k} + \mu\right)^{-1} I = \frac{\gamma_k}{1 + \gamma_k \mu} I.$$

The proposed method sets $x_{k+1} = x_k + d_k(\mu)$ without a step length. This controls the parameter μ to guarantee the global convergence, as in [19]. Note that the regularized parameter μ and $d_k(\mu)$ have the following relations:

$$\begin{cases} d_k(\mu) \to -B_k^{-1}\nabla f(x_k) & (\mu \to 0) \\ d_k(\mu) \to -\frac{1}{\mu}\nabla f(x_k) & (\mu \to \infty). \end{cases}$$

These directions correspond to the steepest decent direction and the L-BFGS direction, respectively. Figure 2.1 shows the relation between these search directions. Namely, a large μ gives the descent direction of f, while a small μ gives the L-BFGS direction.

We exploit the idea of updating the trust region radius in the TR-method to control μ to find an appropriate search direction, that is, we use the ratio of the reduction in the objective function value to that of the model function value. We define a ratio function $r_k(d_k(\mu),\mu)$ by

$$r_k(d_k(\mu),\mu) = \frac{f(x_k) - f(x_k + d_k(\mu))}{f(x_k) - q_k(d_k(\mu),\mu)}, \tag{2.2}$$

where $q_k : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is the following model function at x_k:

$$q_k(d_k(\mu),\mu) = f(x_k) + \nabla f(x_k)^T d_k(\mu) + \frac{1}{2} d_k(\mu)^T \hat{H}_k(\mu)^{-1} d_k(\mu).$$

Note that we do not have to compute the matrix $\hat{H}_k(\mu)^{-1}$ explicitly in $q_k(d_k,\mu)$. Since $d_k(\mu) = -\hat{H}_k(\mu)\nabla f(x_k)$, we have $d_k(\mu)^T\hat{H}_k(\mu)^{-1}d_k(\mu) = -d_k(\mu)^T\nabla f(x_k)$. If the ratio $r_k(d_k(\mu),\mu)$ is large, i.e., the reduction in the objective function f is sufficiently large compared to that of the model function, we adopt $d_k(\mu)$ and decrease the parameter μ. On the other hand, if $r_k(d_k,\mu)$ is small, i.e., $f(x_k) - f(x_k + d_k)$ is small, we increase μ and compute $d_k(\mu)$ again.

Based on the above ideas, we propose the following regularized L-BFGS method.

Algorithm 2.1 Regularized L-BFGS method

Step 0 Choose the parameters $\mu_0, \mu_{\min}, \gamma_1, \gamma_2, \eta_1, \eta_2, m$ such that $0 < \mu_{\min} \le \mu_0, 0 < \gamma_1 \le 1 < \gamma_2, 0 < \eta_1 < \eta_2 \le 1$, and $m > 0$. Choose an initial point $x_0 \in \mathbb{R}^n$ and an initial matrix $\hat{H}_0^{(0)}$. Set $k := 0$.

Step 1 If some stopping criteria are satisfied, then terminate. Otherwise, go to Step 2.

Step 2

Step 2-0 Set $l_k := 0$ and $\bar{\mu}_{l_k} = \mu_k$.

Step 2-1 Compute $d_k(\bar{\mu}_{l_k})$ using the L-BFGS method.

Step 2-2 Compute $r_k(d_k(\bar{\mu}_{l_k}),\bar{\mu}_{l_k})$. If $r_k(d_k(\bar{\mu}_{l_k}),\bar{\mu}_{l_k}) < \eta_1$, then update $\bar{\mu}_{l_k+1} = \gamma_2\bar{\mu}_{l_k}$, set $l_k = l_k + 1$, and go to Step 2-1. Otherwise, go to Step 3.

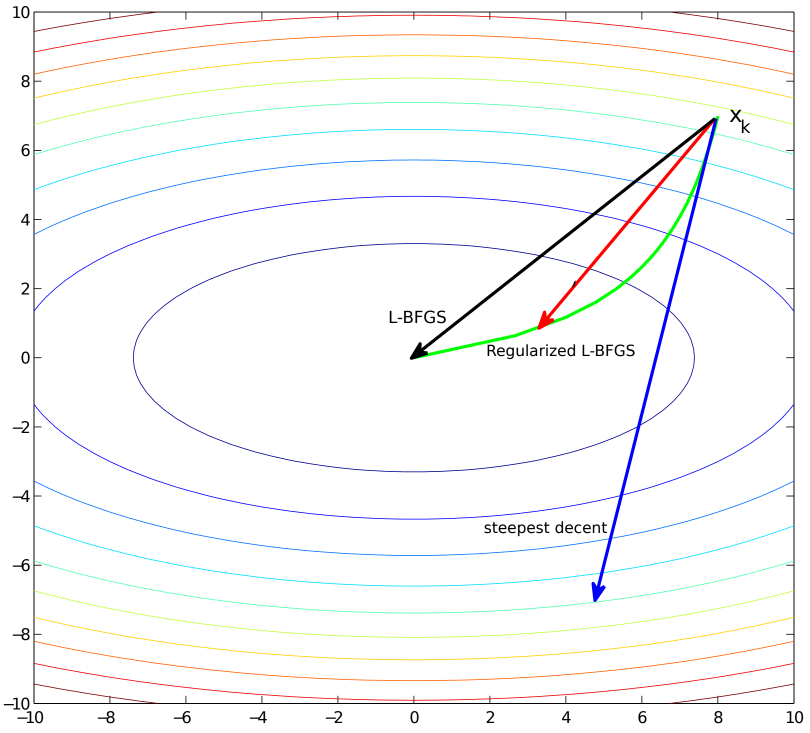

Figure 2.1: Implementations of μ and $d_k(\mu)$.

Step 3 If $\eta_1 \le r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) < \eta_2$, then update $\mu_{k+1} = \bar{\mu}_{l_k}$.
If $r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) \ge \eta_2$, then update $\mu_{k+1} = \max[\mu_{\min}, \gamma_1 \bar{\mu}_{l_k}]$.
Update $x_{k+1} = x_k + d_k(\bar{\mu_{l_k}})$. Set $k = k+1$, and go to Step 1.

In Step 2-1, we compute $d_k(\mu)$ from $(s_k, \hat{y}_k(\mu))$ by the L-BFGS updating scheme in [11]. The details of Step 2-1 are as follows.

Algorithm 2.2 Regularized L-BFGS update at kth iteration with parameter μ

Step 0 Set $p \leftarrow \nabla f(x_k)$.

Step 1 Repeat the following process with $i = k-1, k-2, \cdots, k-t$:

$$\begin{array}{rcl} r_i & \leftarrow & \tau_i s_i^T p \\ p & \leftarrow & p - r_i(y_k + \mu s_k), \end{array}$$

where $\tau_i = (s_i^T(y_k + \mu s_k))^{-1}$.

Step 2 Set $q \leftarrow \hat{H}_k^{(0)}(\mu)p$.

Step 3 Repeat the following process with $i = k-t, k-t+1, \cdots, k-1$:

$$\begin{array}{rcl} \beta & \leftarrow & \tau_i(y_k + \mu s_k)^T q \\ q & \leftarrow & q + (r_i - \beta)s_i. \end{array}$$

Step 4 Obtain the search direction as $d_k(\mu) = -q$.

Note that, since μ_k may vary in each iteration, it is impractical to store $y_k(\mu)$ for all μ_k. Fortunately, Algorithm 2.2 uses y_k and s_k. Thus, we do not have to store $\hat{y}_k(\mu)$. When the regularized parameter changes, we get $\hat{y}_k(\mu)$ from s_k and y_k immediately.

3 Global convergence

In this section, we show the global convergence of the proposed algorithm. To this end, we need the following assumptions.

Assumption 3.1

(i) *The objective function f is twice continuously differentiable.*

(ii) *The level set of f at the initial point x_0 is compact, i.e., $\Omega = \{x \in \mathbb{R}^n | f(x) \le f(x_0)\}$ is compact.*

(iii) *There exist positive constants* M_1 *and* M_2 *such that*

$$M_1\|z\|^2 \le z^T\nabla^2 f(x)z \le M_2\|z\|^2, \ \forall x \in \Omega.$$

(iv) *There exists a minimum* $f_{\min}$ *of* f*.*

(v) *There exists a constant* $\underline{\gamma}$ *such that* $\gamma_k \ge \underline{\gamma} > 0$ *for all* k*.*

The above assumptions are the same as those for the global convergence of the original L-BFGS method.

Under these assumptions, we have the following properties. First, we define

$$G(x) = \nabla^2 f(x), \quad G_k = G(x_k), \quad \bar{G}_k = \int_0^1 G(x_k + \tau s_k)d\tau.$$

It then follows from Taylor's theorem that

$$f(x_k + d_k(\mu)) = f(x_k) + \nabla f(x_k)^T d_k(\mu) + \frac{1}{2}\int_0^1 d_k(\mu)^T G(x_k + \tau d_k(\mu))d_k(\mu)d\tau.$$

Furthermore, since $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, we have

$$y_k = \bar{G}_k s_k. \tag{3.1}$$

$$\hat{y}_k(\mu) = y_k + \mu s_k = (\bar{G}_k + \mu I)s_k. \tag{3.2}$$

It follows from Assumption 3.1 (iii) that $\lambda_{\min}(\bar{G}_k) \ge M_1$ and $\lambda_{\max}(\bar{G}_k) \le M_2$. Therefore, we have that

$$\begin{array}{rcl} M_1\|s\|^2 & \le & s_k^T y_k \le M_2\|s_k\|^2, \\ \dfrac{1}{M_2}\|y_k\|^2 & \le & s_k^T y_k \le \dfrac{1}{M_1}\|y_k\|^2, \\ (M_1+\mu)\|s_k\|^2 & \le & s_k^T \hat{y}_k(\mu) \le (M_2+\mu)\|s_k\|^2. \end{array} \tag{3.3}$$

Since the sequence $\{x_k\}$ is included in the compact set Ω, and f is twice continuously differentiable under Assumption 3.1 (ii), there exists a positive constant L_f such that

$$\|\nabla f(x_k)\| \le L_f, \quad \text{for all } k. \tag{3.4}$$

Now, we investigate the behavior of the eigenvalues of $\hat{B}_k(\mu)$, which is the inverse of $\hat{H}_k(\mu)$. Note that the matrix $\hat{B}_k(\mu)$ is constructed by the BFGS formula with vector pairs $(s_k, \hat{y}_k(\mu))$ and initial matrix $\hat{B}_k^{(0)}(\mu) = \hat{H}_k^{(0)}(\mu)^{-1}$. Thus, we have

$$\begin{aligned} &\hat{B}_k(\mu) = \hat{B}_k^{(\tilde{m}_k)}(\mu) \\ &\hat{B}_k^{(l+1)}(\mu) = \hat{B}_k^{(l)}(\mu) - \frac{\hat{B}_k^{(l)}(\mu)s_{j_l}s_{j_l}^T\hat{B}_k^{(l)}(\mu)}{s_{j_l}^T\hat{B}_k^{(l)}(\mu)s_{j_l}} + \frac{y_{j_l}y_{j_l}^T}{y_{j_l}^T s_{j_l}}, \quad l = 0, \cdots, \tilde{m}_k - 1 \end{aligned} \tag{3.5}$$

where $\tilde{m}_k = \min\{k+1, m\}$ and $j_l = k - \tilde{m} + l$. Note that these expressions are used in [4, 9].

We now focus on the trace and determinant of $\hat{B}_k(\mu)$. First, we show that the trace of $\hat{B}_k^{(l)}(\mu)$ is $O(\mu)$.

Lemma 3.1 *Suppose that Assumption 3.1 holds. Then,*

$$\mathrm{tr}(\hat{B}_k^{(l)}(\mu)) \leq M_3 + (2m+n)\mu, \quad l = 0, \cdots, \ \tilde{m}_k$$

where

$$M_3 = \frac{n}{\underline{\gamma}} + mM_2.$$

Proof We have from Assumption 3.1, (3.1), and (3.3) that

$$\begin{aligned}
\frac{\|\hat{y}_k(\mu)\|^2}{s_k^T \hat{y}_k(\mu)} &= \frac{\|y_k\|^2 + 2\mu s_k^T y_k + \mu^2 \|s_k\|}{s_k^T y_k + \mu \|s_k\|^2} \\
&= \frac{\|y_k\|^2 + \mu s_k^T y_k}{s_k^T y_k + \mu \|s_k\|^2} + \frac{\mu(s_k^T y_k + \mu \|s_k\|^2)}{s_k^T y_k + \|s_k\|^2} \\
&\leq \frac{\|y_k\|^2 + \mu s_k^T y_k}{s_k^T y_k} + \mu \\
&\leq \frac{\|y_k\|^2}{\frac{1}{M_2}\|y_k\|^2} + 2\mu \\
&= M_2 + 2\mu. \qquad (3.6)
\end{aligned}$$

From the updating formula (3.5) of matrix $\hat{B}_k(\mu)$,

$$\mathrm{tr}(\hat{B}_k^{(l)}(\mu)) = \mathrm{tr}(\hat{B}_k^{(0)}(\mu)) + \sum_{t=0}^{l-1} \left(-\frac{\|\hat{B}_k^{(t)}(\mu) s_{j_t}\|^2}{s_{j_t}^T \hat{B}_k^{(t)}(\mu) s_{j_t}} + \frac{\|\hat{y_{j_t}}(\mu)\|^2}{s_{j_t}^T \hat{y}_{j_t}(\mu)} \right),$$

where

$$\frac{\|\hat{B}_k^{(t)}(\mu) s_{j_t}\|^2}{s_{j_t}^T \hat{B}_k^{(t)}(\mu) s_{j_t}} \geq 0.$$

It then follows from (3.6) that

$$\begin{aligned}
\mathrm{tr}(\hat{B}_k^{(l)}(\mu)) &\leq \mathrm{tr}(\hat{B}_k^{(0)}(\mu)) + \sum_{t=0}^{l-1} \frac{\|\hat{y_{j_t}}(\mu)\|^2}{s_{j_t}^T \hat{y}_{j_t}(\mu)} \\
&\leq \mathrm{tr}(\hat{B}_k^{(0)}(\mu)) + l(M_2 + 2\mu) \\
&= n\left(\frac{1}{\gamma_k} + \mu\right) + l(M_2 + 2\mu) \\
&\leq n\left(\frac{1}{\underline{\gamma}} + \mu\right) + m(M_2 + 2\mu) \\
&\leq M_3 + (2m+n)\mu.
\end{aligned}$$

This completes the proof. □

The next lemma gives a lower bound for the determinant of $\hat{B}_k(\mu)$.

Lemma 3.2 *Suppose that Assumption 3.1 holds. Then,*

$$\det(\hat{B}_k(\mu)) \geq M_4 \mu^n,$$

where

$$M_4 = \left(\frac{1}{2m+n}\right)^m.$$

Proof Note that the determinant of the approximate matrix updated by the BFGS updating scheme has the following property [14, 15]:

$$\det(\hat{B}_k^{(l+1)}(\mu)) = \det(\hat{B}_k^{(l)}(\mu)) \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{s_{j_l}^T \hat{B}_{k-1}^{(l)}(\mu) s_{j_l}}.$$

Then, we have

$$\begin{aligned}
\det(\hat{B}_k(\mu)) &= \det(\hat{B}_k^{(0)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{s_{j_l}^T \hat{B}_{k-1}^{(0)}(\mu) s_{j_l}} \\
&= \det(\hat{B}_k^{(0)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{s_{j_l}^T s_{j_l}} \frac{s_{j_l}^T s_{j_l}}{s_{j_l}^T \hat{B}_k^{(l)}(\mu) s_{j_l}} \\
&\geq \det(\hat{B}_k^{(0)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{s_{j_l}^T s_{j_l}} \frac{s_{j_l}^T s_{j_l}}{\lambda_{\max}(\hat{B}_k^{(l)}(\mu)) s_{j_l}^T s_{j_l}} \\
&= \det(\hat{B}_k^{(0)}(\mu)) \prod_{l=0}^{\tilde{m}-1} \frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{\|s_{j_l}\|^2} \frac{1}{\lambda_{\max}(\hat{B}_k^{(l)}(\mu))}.
\end{aligned}$$

Since $B_k^{(0)}(\mu)$ is symmetric positive-definite, Lemma 3.1 implies that $\lambda_{\max}(\hat{B}_k^{(l)}(\mu)) \geq M_3 + (2m+n)\mu$. Furthermore, we have $\frac{s_{j_l}^T \hat{y}_{j_l}(\mu)}{\|s_{j_l}\|^2} \geq M_1 + \mu$ from (3.3). Therefore, it follows that

$$\begin{aligned}
\det(\hat{B}_k(\mu)) &\geq \det(\hat{B}_k(\mu)^{(0)}) \left(\frac{M_1+\mu}{M_3+(2m+n)\mu}\right)^{\tilde{m}} \\
&= \det\left(\frac{1+\gamma_k\mu}{\gamma_k} I\right) \left(\frac{M_1+\mu}{M_3+(2m+n)\mu}\right)^{\tilde{m}} \\
&\geq \left(\frac{1}{\gamma_k}+\mu\right)^n \left(\frac{1}{2m+n}\right)^{\tilde{m}} \\
&\geq \left(\frac{1}{2m+n}\right)^m \mu^n \\
&= M_4 \mu^n.
\end{aligned}$$

This completes the proof. □

From the above two lemmas, we have $\lambda_{\max}(\hat{H}_k(\mu)) \to 0$ as $\mu \to \infty$.

Lemma 3.3 *Suppose that Assumption 3.1 holds. Then, for all* $k \geq 0$*,*

$$\lambda_{\max}(\hat{H}_k(\mu)) \leq M_5 \frac{1}{\mu}, \ \forall \mu \in [\mu_{\min}, \infty),$$

where

$$M_5 = \frac{1}{M_4 n^{n-1}} \left(\frac{M_3}{\mu_{\min}} + (2m+n) \right)^{n-1}.$$

Furthermore, $\lim_{\mu \to \infty} \lambda_{\max}(\hat{H}_k(\mu)) = 0$.

Proof We have from Lemmas 3.1 and 3.2 that

$$\begin{array}{rcl} \operatorname{tr}(\hat{B}_k(\mu)) & \leq & M_3 + (2m+n)\mu, \\ \det(\hat{B}_k(\mu)) & \geq & M_4 \mu^n. \end{array}$$

Since $\hat{B}_k(\mu)$ is symmetric positive-definite, we have

$$\begin{array}{rcl} \operatorname{tr}(\hat{B}_k(\mu)) & \geq & n\lambda_{\min}(\hat{B}_k(\mu)) \\ \det(\hat{B}_k(\mu)) & \leq & \lambda_{\min}(\hat{B}_k(\mu)) \{\lambda_{\max}(\hat{B}_k(\mu))\}^{n-1}. \end{array}$$

Therefore, we have

$$\begin{array}{rcl} \lambda_{\min}(\hat{B}_k(\mu)) & \geq & \dfrac{\det(\hat{B}_k(\mu))}{\{\lambda_{\max}(\hat{B}_k(\mu))\}^{n-1}} \\ & \geq & \dfrac{M_4 \mu^n}{\{\frac{1}{n}(M_3 + (2m+n)\mu)\}^{n-1}}. \end{array}$$

It then follows from Assumption 3.1 (v) that

$$\begin{array}{rcl} \lambda_{\max}(\hat{H}_k(\mu)) & = & \dfrac{1}{\lambda_{\min}(\hat{H}_k^{-1}(\mu))} \\ & = & \dfrac{1}{\lambda_{\min}(\hat{B}_k(\mu))} \\ & \leq & \dfrac{\{\frac{1}{n}(M_3 + (2m+n)\mu)\}^{n-1}}{M_4 \mu^n} \\ & = & \dfrac{1}{M_4 n^{n-1}} \left(\dfrac{M_3 + (2m+n)\mu}{\mu} \right)^{n-1} \dfrac{1}{\mu}. \end{array} \tag{3.7}$$

Since $\mu \geq \mu_{\min}$, we have

$$\frac{M_3 + (2m+n)\mu}{\mu} = \frac{M_3}{\mu} + (2m+n) \leq \frac{M_3}{\mu_{\min}} + (2m+n).$$

It then follows from (3.7) that

$$\begin{aligned}\lambda_{\max}(\hat{H}_k(\mu)) &\le \frac{1}{M_4 n^{n-1}}\left(\frac{M_3}{\mu_{\min}}+(2m+n)\right)^{n-1}\frac{1}{\mu}\\ &= M_5\frac{1}{\mu}.\end{aligned}$$

Hence, we have

$$\lim_{\mu\to\infty}\lambda_{\max}(\hat{H}_k(\mu))=0.$$

This completes the proof. □

Now, we give an upper bound for $\|d_k(\mu)\|$.

Lemma 3.4 *Suppose that Assumption 3.1 holds. Then,*

$$\|d_k(\mu)\|\le U_d,$$

where

$$U_d=\frac{L_f M_5}{\mu_{\min}}.$$

Proof From the definition of $d_k(\mu)$, (3.4), and Lemma 3.3, we have that

$$\begin{aligned}\|d_k(\mu)\| &= \|\hat{H}_k(\mu)\nabla f(x_k)\|\\ &\le \|\hat{H}_k(\mu)\|\|\nabla f(x_k)\|\\ &= \lambda_{\max}(\hat{H}_k(\mu))\|\nabla f(x_k)\|\\ &\le \lambda_{\max}(\hat{H}_k(\mu))L_f\\ &\le \frac{L_f M_5}{\mu}\\ &\le \frac{L_f M_5}{\mu_{\min}}=U_d.\end{aligned}$$

This completes the proof. □

Lemma 3.4 implies that

$$x_k+\nu d_k(\mu)\in\Omega+\mathrm{B}(0,U_d),\quad \forall\nu\in[0,1],\quad \forall\mu\in[\mu_{\min},\infty),\quad \forall k\ge 0.$$

Moreover, since $\Omega+\mathrm{B}(0,U_d)$ is compact and f is twice continuously differentiable, $\nabla f(x_k)$ is Lipschitz continuous on $\Omega+\mathrm{B}(0,U_d)$. That is, there exists a positive constant L_g such that

$$\|\nabla^2 f(x_k)\|\le L_g,\quad \forall x_k\in\Omega+\mathrm{B}(0,U_d). \tag{3.8}$$

Next, we investigate the values of μ that satisfy the termination condition $r_k(d_k(\mu),\mu)\ge\eta_1$ in the inner iterations of Step 2-2 in Algorithm 2.1.

Lemma 3.5 *Suppose that Assumption 3.1 holds. Then, we have*

$$f(x_k) - f(x_k + d_k(\mu)) - \eta_1(f(x_k) - q_k(d_k(\mu), \mu)) \geq \frac{1}{2}((2-\eta_1)\lambda_{\min}(\hat{H}_k(\mu)^{-1}) - L_g)\|d_k(\mu)\|^2.$$

Proof We have from Taylor's theorem that

$$\begin{aligned} f(x_k + d_k(\mu)) &= f(x_k) + \int_0^1 \nabla f(x_k + \tau d_k(\mu))^T d_k(\mu) d\tau \\ &= f(x_k) + \nabla f(x_k)^T d_k(\mu) + \int_0^1 (\nabla f(x_k + \tau d_k(\mu)) - \nabla f(x_k))^T d_k(\mu) d\tau. \end{aligned}$$

From the Lipschitz continuity of $\nabla f(x_k)$ in (3.8), we get

$$\begin{aligned} f(x_k) \quad &- \quad f(x_k + d_k(\mu)) - \eta_1(f(x_k) - q_k(d_k(\mu), \mu)) \\ &= -\nabla f(x_k)^T d_k(\mu) - \int_0^1 (\nabla f(x_k + \tau d_k(\mu)) - \nabla f(x_k))^T d_k(\mu) d\tau - \frac{\eta_1}{2} d_k(\mu)^T (\hat{H}_k(\mu)^{-1}) d_k(\mu) \\ &= \frac{(2-\eta_1)}{2} d_k(\mu)^T (\hat{H}_k(\mu)^{-1}) d_k(\mu) - \int_0^1 (\nabla f(x_k + \tau d_k(\mu)) - \nabla f(x_k))^T d_k(\mu) d\tau \\ &\geq \frac{(2-\eta_1)}{2} \lambda_{\min}(\hat{H}_k(\mu)^{-1}) \|d_k(\mu)\|^2 - \int_0^1 L_g \tau \|d_k(\mu)\|^2 d\tau \\ &= \frac{1}{2}((2-\eta_1)\lambda_{\min}(\hat{H}_k(\mu)^{-1}) - L_g)\|d_k(\mu)\|^2. \end{aligned}$$

This completes the proof. $\square$

From Lemma 3.5, if μ satisfies

$$\lambda_{\min}(\hat{H}_k^{-1}(\mu)) \geq \frac{L_g}{2-\eta_1}, \tag{3.9}$$

then we have

$$r_k(d_k(\mu), \mu) \geq \eta_1, \tag{3.10}$$

that is, the inner loops of Algorithm 2.1 must terminate.

Next, we give an upper bound for the parameter μ_k.

Lemma 3.6 *Suppose that Assumption 3.1 holds. Then, for any $k \geq 0$,*

$$\mu_k^* \leq U_\mu,$$

where

$$U_\mu = \gamma_2 M_5 \frac{L_g}{2-\eta_1}.$$

Proof If $\bar{\mu}_{l_k}$ satisfies (3.9), then $r_k(d_k(\bar{\mu}_{l_k}), \bar{\mu}_{l_k}) \geq \eta_1$ from Lemma 3.5. Therefore, the inner loops must terminate, and we set $\mu_k^* = \bar{\mu}_{l_k}$.

Now, we give the termination condition on μ for the inner loop. We have from Lemma 3.3 that

$$\begin{aligned}\lambda_{\min}(\hat{H}_k^{-1}(\mu)) &= \frac{1}{\lambda_{\max}(\hat{H}_k(\mu))} \\ &\geq \frac{\mu}{M_5}. \end{aligned} \tag{3.11}$$

It then follows from (3.9) that the termination condition of the inner loop holds when

$$\mu \geq M_5 \frac{L_g}{2-\eta_1}. \tag{3.12}$$

Note that, if the inner loop terminates at l_k, then (3.12) does not hold with $\mu = \mu_{l_k-1}$, that is,

$$\bar{\mu}_{l_k-1} < M_5 \frac{L_g}{2-\eta_1}.$$

Since $\mu_k^* = \bar{\mu}_{l_k} = \gamma_2 \bar{\mu}_{l_k-1}$, we have

$$\mu_k^* = \gamma_2 \bar{\mu}_{l_k-1} < \gamma_2 M_5 \frac{L_g}{2-\eta_1} = U_\mu. \tag{3.13}$$

This completes the proof. □

Next, we give a lower bound for the reduction in the model function q_k.

Lemma 3.7 *Suppose that Assumption 3.1 holds. Then, we have*

$$f(x_k) - q_k(d_k(\mu), \mu) \quad \geq \quad M_6 \|\nabla f(x_k)\|^2,$$

where

$$M_6 = \frac{1}{2(M_3 + (2m+n)\mu_{\min})}.$$

Proof It follows from the definition of the model function $q_k(d_k(\mu), \mu)$ and Lemmas 3.1, 3.6 that

$$
\begin{aligned}
f(x_k) - q_k(d_k(\mu), \mu) &= -\frac{1}{2} d_k(\mu)^T (\hat{H}_k^{-1}(\mu)) d_k(\mu) - \nabla f(x_k)^T d_k(\mu) \\
&= -\frac{1}{2} \nabla f(x_k)^T \hat{H}_k(\mu) \nabla f(x_k) + \nabla f(x_k)^T \hat{H}_k(\mu) \nabla f(x_k) \\
&= \frac{1}{2} \nabla f(x_k)^T \hat{H}_k(\mu) \nabla f(x_k) \\
&\geq \frac{1}{2} \lambda_{\min}(\hat{H}_k(\mu)) \|\nabla f(x_k)\|^2 \\
&= \frac{\|\nabla f(x_k)\|^2}{2\lambda_{\max}(\hat{H}_k^{-1}(\mu))} \\
&= \frac{\|\nabla f(x_k)\|^2}{2\lambda_{\max}(\hat{B}_k(\mu))} \\
&\geq \frac{\|\nabla f(x_k)\|^2}{2\mathrm{tr}(\hat{B}_k(\mu))} \\
&\geq \frac{\|\nabla f(x_k)\|^2}{2(M_3 + (2m+n)\mu)} \\
&= \frac{1}{2(M_3 + (2m+n)U_\mu)} \|\nabla f(x_k)\|^2 \\
&= M_6 \|\nabla f(x_k)\|^2.
\end{aligned}
$$

This completes the proof. □

From this lemma, we can give a lower bound for the reduction in the objective function value when x_k is not a stationary point.

Lemma 3.8 *Suppose that Assumption 3.1 holds. If there exists a positive constant ϵ_g such that $\|\nabla f(x_k)\| \geq \epsilon_g$, then we have $f(x_k) - f(x_{k+1}) \geq \rho \epsilon_g^2$, where $\rho = \eta_1 M_6$.*

Proof It follows from Lemmas 3.6 and 3.7 that

$$
\begin{aligned}
f(x_k) - f(x_{k+1}) &\geq \eta_1 (f(x_k) - q_k(d_k(\mu_k^*), \mu_k^*)) \\
&\geq \eta_1 M_6 \|\nabla f(x_k)\|^2 \\
&\geq \rho \epsilon_g^2.
\end{aligned}
$$

This completes the proof. □

We are now in a position to prove the main theorem of this section.

Theorem 3.1 *Suppose that Assumption 3.1 holds. Then,* $\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0$ *or there exists $K \geq 0$ such that $\|\nabla f(x_K)\| = 0$.*

Proof Suppose the contrary, i.e., there exists a positive constant ϵ_g such that $\|\nabla f(x_k)\| \geq \epsilon_g$ for all $k \geq 0$. It follows from Lemma 3.8 that

$$\begin{aligned} f(x_0) - f(x_k) &= \sum_{j=0}^{k-1}(f(x_j) - f(x_{j+1})) \\ &\geq \sum_{j=0}^{k-1} \rho\epsilon_g^2 \\ &= \rho\epsilon_g^2 k. \end{aligned}$$

Taking $k \to \infty$, the right-hand side of the final inequality goes to infinity, and hence

$$\lim_{k\to\infty} f(x_k) = -\infty.$$

This contradicts the existence of $f_{\min}$ in Assumption 3.1 (iv). This completes the proof. □

4 Implementation issues

In this section, we discuss some issues regarding the practical implementation of the regularized L-BFGS method.

Scaling parameter γ_k in the initial matrix.

The regularized L-BFGS method uses the following initial matrix in each iteration:

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

The parameter γ_k represents the scale of $\nabla^2 f(x)$. Thus, we exploit some properties of the scaling parameter γ_k used in [1, 3, 11, 16, 17], that is, we set

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

It is known that the L-BFGS method with this scaling in the initial matrix has an efficient performance [3, 11]. Note that $\gamma_k > 0$ to ensure the positive-definiteness of $\hat{H}_k^{(0)}(\mu)$. If $s_{k-1}^T y_{k-1} < \alpha\|s_{k-1}\|^2$, then we set $\gamma_k = \alpha\frac{\|s_{k-1}\|^2}{\|y_{k-1}\|^2}$, where α is a small, positive constant.

Nonmonotone decreasing technique.

In Algorithm 2.1, we control the regularized parameter μ to satisfy the descent condition $f(x_{k+1}) < f(x_k)$. However, μ sometimes becomes quite large when treating ill-posed problems. In this situation, we require a large number of function evaluations. Therefore, we use the concept of a nonmonotone line search technique [8, 18] to overcome this difficulty. We replace the ratio function $r_k(d_k(\mu), \mu)$ with the following new ratio function $\bar{r}_k(d_k(\mu), \mu)$:

$$\bar{r}_k(d_k(\mu), \mu) = \frac{\max_{0\leq j\leq m(k)} f(x_{k-j}) - f(x_k + d_k(\mu))}{f(x_k) - q_k(d_k(\mu), \mu)},$$

where

$$m(0) = 0, \quad 0 \leq m(k) \leq \min\{m(k-1)+1, M\},$$

and M is a nonnegative integer constant. This modification retains the global convergence of the regularized L-BFGS method.

In the numerical experiments reported in the next section, when $k < M$, we use the original ratio function $r_k(d_k(\mu), \mu)$, and if $k \geq M$, we use the new ratio function $\bar{r}_k(d_k(\mu), \mu)$.

Use of gradient information from rejected steps.

We focus on the rejected steps. These occur in Step 2-2 of the regularized L-BFGS algorithm. Namely, if l_k satisfies $r_k(d_k(\mu_{l_k}), \mu_{l_k}) \leq \eta_1$ in Step 2-2, then l_k is rejected, and we may exploit information about the gradient at this rejected step. These gradients contain important information about the behavior of the objective function. Therefore, we store (s_{l_k}, y_{l_k}) at these rejected step to update the approximate Hessian. We do not have to increase the number of pairs m. Instead, when we store the new pair (s_{l_k}, y_{l_k}) at the rejected step, we discard the oldest vector pair. It is known that the L-BFGS method with this modification has good performance [2].

5 Numerical results

In this section, we report some numerical results for the proposed algorithm. We consider the following.

1. The influence of parameters contained in Algorithm 2.1.
2. Comparison of the proposed algorithm with the original L-BFGS method [13].

For each experiment, benchmark problems were chosen from CUTEst [7]. All algorithms were coded in FORTRAN90 and run on a machine with an Intel Core i5 1.7 GHz CPU and 4 GB RAM. We used the original L-BFGS algorithm coded by Nocedal [13] with the default settings. We adopted an initial point x_0 given by CUTEst, and set the termination criteria as

$$\frac{\|f(x_k)\|}{\max(1, \|x_k\|)} < \epsilon \quad \text{and} \quad n_f > 10000,$$

where n_f is the number of function evaluations. These criteria are similar to those in [13]. We consider trials in which $n_f > 10000$ to have failed.

We compared the algorithms using the following distribution function, which was proposed in [6]. Let $\mathcal{S}$ be a set of solvers, and let $\mathcal{P}_{\mathcal{S}}$ be a set of problems that can be solved by all methods in $\mathcal{S}$. We denote a measure for the number of evaluations required by solver s for problem p as $t_{p,s}$, and the best $t_{p,s}$ for each p is written as t_p^*, i.e., $t_p^* := \min\{t_{p,s} | s \in \mathcal{S}\}$. Then, the distribution function $F_s^{\mathcal{S}}(\tau)$ for a method s is defined by

$$F_s^{\mathcal{S}}(\tau) = \frac{|\{p \in \mathcal{P}_{\mathcal{S}} | t_{p,s} \leq \tau t_p^*\}|}{|\mathcal{P}_{\mathcal{S}}|}, \quad \tau \geq 1.$$

The algorithm for which $F_s^{\mathcal{S}}(\tau)$ is closest to 1 is considered to be superior to the other algorithms in $\mathcal{S}$.

5.1 Influence of various parameters in Algorithm 2.1

The proposed algorithm has several parameters. Therefore, we investigated their influence, and identified favorable values.

First of all, we focused on γ_1 and γ_2, which are used to control the regularized parameter μ. The other parameters were set as follows:

$$\eta_1 = 0.01,\ \eta_2 = 0.9,\ \mu_{\min} = 1.0 \times 10^{-3},\ m = 5.$$

The nonmonotone parameter M was set to 8. We compare the parameter sets $P_1, \cdots, P_9$ in Table 5.1. The table also shows the number of successes and the success rate for all 313 test problems.

Table 5.1: Number of successes and success rate at each (γ_1, γ_2).

	γ_1	γ_2	Number of successes	Success rate (%)
P_1	0.1	2.0	266	85.0
P_2	0.1	5.0	267	85.3
P_3	0.1	10.0	268	85.6
P_4	0.2	2.0	268	85.6
P_5	0.2	5.0	267	85.3
P_6	0.2	10.0	266	85.0
P_7	0.5	2.0	267	85.3
P_8	0.5	5.0	268	85.6
P_9	0.5	10.0	265	84.7

Figure 5.1 shows the distribution function of these parameter sets in terms of the CPU time.

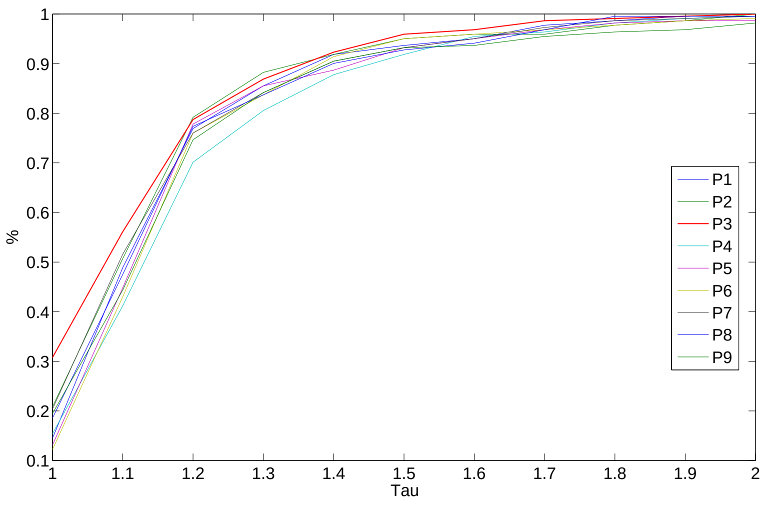

Figure 5.1: Comparison of (γ_1, γ_2).

We can see that the parameter set P_3 gives the best results. Therefore, we set $\gamma_1 = 0.1$ and $\gamma_2 = 10.0$ in the subsequent experiments.

Next, we focused on the number of memories m. In the original L-BFGS method, we usually set m in the range $3 \leq m \leq 7$ [11]. Thus, we compare $m = 3,\ 5,\ 7$. The other parameters were set as follows:

$$\gamma_1 = 0.1,\ \gamma_2 = 10.0,\ \eta_1 = 0.01,\ \eta_2 = 0.9,\ \mu_{\min} = 1.0 \times 10^{-3},\ M = 8.$$

Table 5.2 shows the number of successes and the success rate for all 313 test problems.

Table 5.2: Number of successes and success rate at each m.

m	Number of successes	Success rate (%)
3	267	85.3
5	268	85.6
7	266	85.0

Figure 5.2 shows the distribution function of the number of memories in terms of the CPU time.

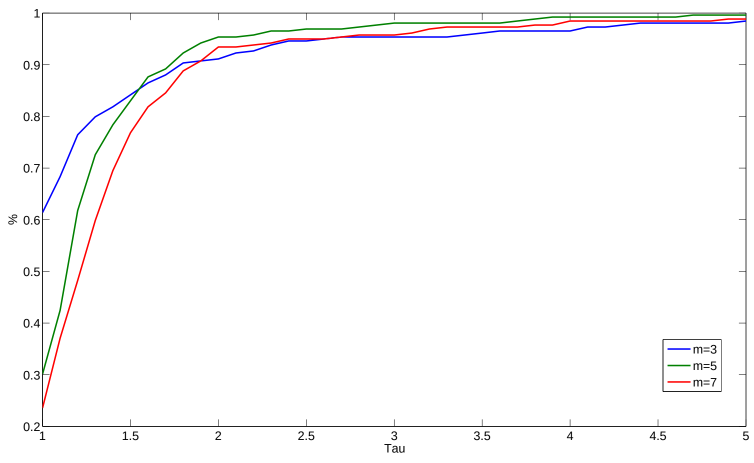

Figure 5.2: Comparison of $m = 3, 5, 7$.

We can see that $m = 5$ gives the best results. Therefore, we set $m = 5$ in the subsequent experiments.

Next, we focused on the nonmonotone parameter M. We compared $M = 4,\ 6,\ 8,\ 10,\ 12$ and $M = 0$, which corresponds to the usual monotone decreasing case. The other parameters were set as follows:

$$\gamma_1 = 0.1,\ \gamma_2 = 10.0,\ \eta_1 = 0.01,\ \eta_2 = 0.9,\ \mu_{\min} = 1.0 \times 10^{-3},\ m = 5.$$

Table 5.3 shows the number of successes and the success rate for all 313 test problems.

Table 5.3: Number of successes and success rate at each M.

M	Number of successes	Success rate (%)
Monotone (M=0)	225	71.9
4	263	84.0
6	265	84.7
8	265	84.7
10	268	85.6
12	268	85.6

Figure 5.3 shows the distribution function of the nonmonotone parameter in terms of the CPU time.

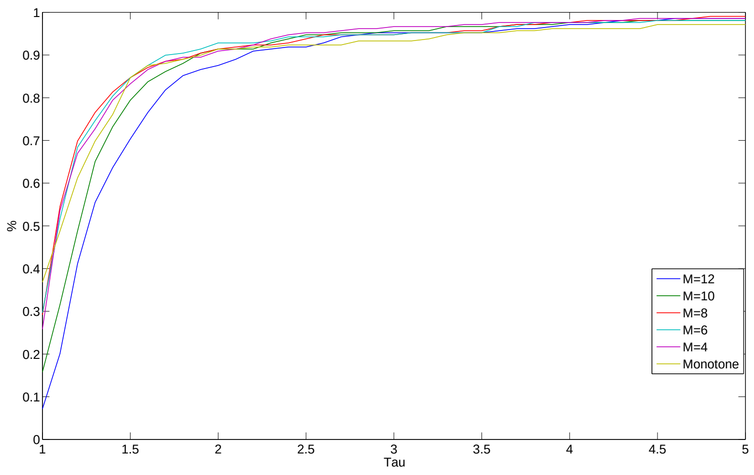

Figure 5.3: Comparison of $M = 4,\ 6,\ 8,\ 10,\ 12$ and monotone decreasing $(M = 0)$.

We can see that $M = 8$ produces the best results. Therefore, we set $M = 8$ in subsequent experiments.

Finally, we compared the algorithm using the rejected information (Reject) with the basic one (Basic) introduced in Section 4. The other parameters were set as follows:

$$\gamma_1 = 0.1,\ \gamma_2 = 10.0,\ \eta_1 = 0.01,\ \eta_2 = 0.9,\ \mu_{\min} = 1.0 \times 10^{-3},\ m = 5,\ M = 8.$$

Table 5.4 shows the number of successes and the success rate for all 313 test problems.

Table 5.4: Number of successes and success rate of Reject and Basic.

Algorithm	Number of successes	Success rate (%)
Reject	268	85.6
Basic	270	86.3

Figure 5.4 shows the distribution function of these two algorithms in terms of the CPU time.

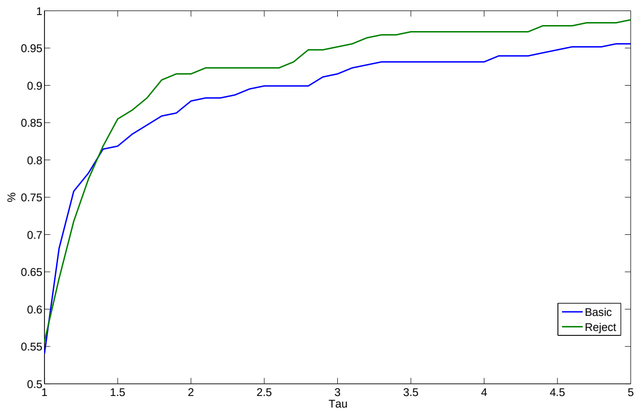

Figure 5.4: Comparison of Reject and Basic.

We see that the algorithm using the rejected information outperforms the basic algorithm. Therefore, we used the algorithm that utilizes rejected information to compare the regularized L-BFGS method and the conventional L-BFGS method.

5.2 Comparison of the regularized L-BFGS method and the conventional L-BFGS method

We compare the regularized L-BFGS method (RL-BFGS) and the L-BFGS method in terms of the number of function evaluations and CPU time.

Table 5.5 shows the number of successes and the success rate for all 313 test problems. Figures 5.5 and 5.6 show the results for $\mathcal{P}_{\mathcal{S}}$, and Figures 5.7 and 5.8 show the results for large-scale problems. Here, we define a large-scale problem as one in which the dimension of the decision variables is greater than 1000. There are 147 large-scale test problems. ${\mathcal{P}_{\mathcal{S}}}^{\text{large}}$ denotes the set of large-scale problems extracted from $\mathcal{P}_{\mathcal{S}}$. Table 5.6 shows the number of successes and the success rate for these 147 large-scale test problems.

The success rates in Tables 5.5 and 5.6 show that RL-BFGS solves 85% in each case. However, the L-BFGS method only solves 71.9% of all test problems and 68.7% of the large-scale test problems. Therefore, we argue that the RL-BFGS method is superior to the L-BFGS method.

Figure 5.5 shows that RL-BFGS requires fewer function evaluations than L-BFGS. Furthermore, Figure 5.7 illustrates that the results for large-scale test problems show a slight improvement over those for all test problems. Figure 5.6 shows that the RL-BFGS method is slightly faster than the L-BFGS method. However, from Figure 5.8, we can see that the results for large-scale test problems do not show such an improvement. As some large-scale problems are comparatively easy to solve, we can infer that the line search technique in the L-BFGS method has good performance for these easy problems.

Table 5.5: Number of successes and the success rate of RL-BFGS and L-BFGS for all 313 problems.

Algorithm	Number of successes	Success rate (%)
RL-BFGS	266	85.0
L-BFGS	225	71.9

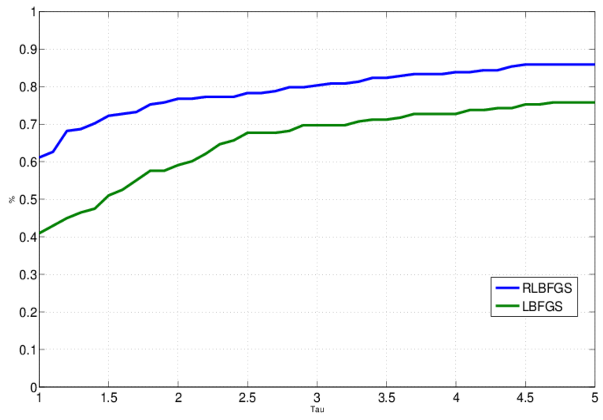

Figure 5.5: Comparison of RL-BFGS and L-BFGS in terms of n_f for $\mathcal{P}_{\mathcal{S}}$.

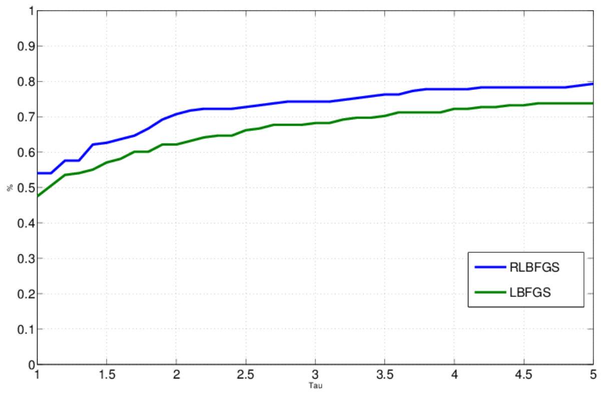

Figure 5.6: Comparison of RL-BFGS and L-BFGS in terms of CPU time for $\mathcal{P}_{\mathcal{S}}$.

Table 5.6: Number of successes and success rate of RL-BFGS and L-BFGS for 147 large-scale problems.

Algorithm	Number of successes	Success rate (%)
RL-BFGS	125	85.0
L-BFGS	101	68.7

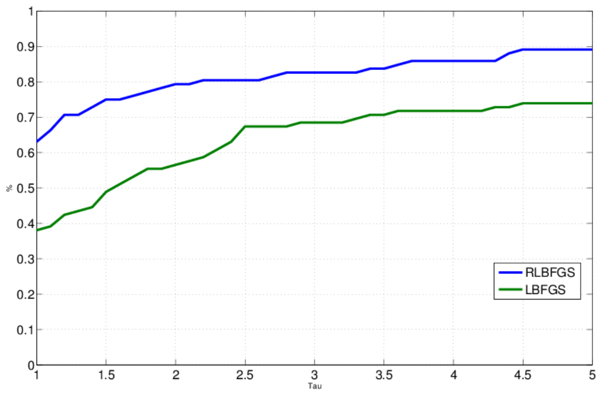

Figure 5.7: Comparison of RL-BFGS and L-BFGS in terms of n_f for ${\mathcal{P}_{\mathcal{S}}}^{\text{large}}$.

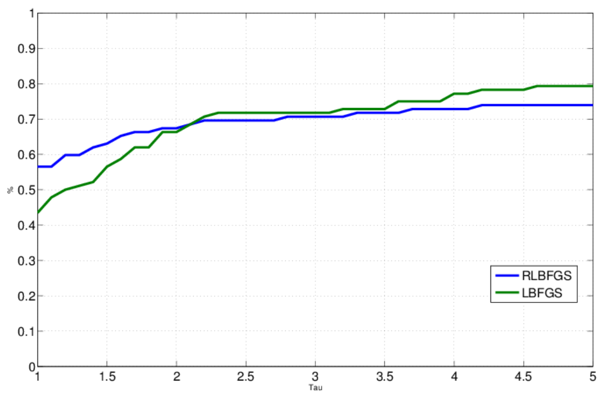

Figure 5.8: Comparison of RL-BFGS and L-BFGS in terms of CPU time for ${\mathcal{P}_\mathcal{S}}^{\text{large}}$.

6 Conclusion

In this paper, we proposed a regularized L-BFGS method for unconstrained minimization problems, and demonstrated its global convergence under some appropriate assumptions. From the numerical results, we saw that the regularized L-BFGS method is able to solve more test problems than the original L-BFGS. This result indicates that the proposed method is robust.

In future work, we will consider solving the following boxed constrained optimization problems:

$$\begin{aligned} &\text{minimize} \;\; f(x) \\ &\text{subject to} \;\; l \le x \le u, \end{aligned}$$

where l and u are vectors such that $l_i \in [-\infty, \infty)$, $u_i \in (-\infty, \infty]$ and $l_i < u_i$ for all i. It is known that boxed constrained optimization problems play an important role in the development of algorithms for general constrained problems.

References

[1] J. Barazilai and J. M. Borwein, *Two-Point Step Size Gradient Methods*, IMA Journal of Numerical Analysis 8 (1988), pp. 141–148.

[2] J. V. Burke and A. Wiegmann, *Notes on limited memory BFGS updating in a trust-region framework*, Technical report, Department of Mathematics, University of Washington, 1996.

[3] J. V. Burke, A. Wiegmann and L. Xu, *Limited memory BFGS updating in a trust-region framework*, Technical report, Department of Mathematics, University of Washington, 2008.

[4] R. H. Byrd, J. Nocedal and R. B. Schnabel, *Representations of quasi-Newton matrices and their use in limited memory methods*, Mathematical Programming 63 (1994), pp. 129–156.

[5] J. E. Dennis Jr. and J. J. Moré, *Quasi-Newton methods, motivation and theory*, SIAM review 19 (1977), pp. 46–89.

[6] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming 91 (2002), pp. 201–213.

[7] N. I. M. Gould, D. Orban and P. L. Toint, *CUTEr and SifDec, a constrained and unconstrained testing environment, revisited*, ACM Transactions on Mathematical Software 29 (2003), pp. 373–394.

[8] L. Grippo, F. Lampariello and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM Journal on Numerical Analysis 23 (1986), pp. 707–716.

[9] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Mathematical Programming 45 (1989), pp. 503–528.

[10] J. J. Moré and C. S. Danny, *Computing a trust region step*, SIAM Journal on Scientific and Statistical Computing 4 (1983), pp. 553–572.

[11] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Mathematics of Computation 35 (1980), pp. 773–782.

[12] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research, New York, 1999.

[13] J. Nocedal, *Software for Large-scale Unconstrained Optimization: L-BFGS distribution*, Available at:http://www.ece.northwestern.edu/ nocedal/lbfgs.html.

[14] J. D. Pearson, *Variable metric methods of minimisation*, The Computer Journal 12 (1969), pp. 171–178.

[15] M. J. D. Powell, *Some global convergence properties of a variable metric algorithm for minimization without exact line search*, in: R. W. Cottle and C. E. Lemke eds., *Nonlinear Programming*, SIAM-AMS Proceedings IX, SIAM Publications, 1976.

[16] M. Raydan, *The Barzilai and Borwein gradient method for large scale unconstrained minimization problem*, SIAM Journal Optimization 7 (1997), pp. 26–33.

[17] D. F. Shanno, and P. A. Kang-Hoh, *Matrix conditioning and nonlinear optimization*, Mathematical Programming 14 (1978), pp. 149–160.

[18] W. Sun, *Nonmonotone trust region method for solving optimization problems*, Applied Mathematics and Computation 156 (2004), pp. 159–174.

[19] K. Ueda, and N. Yamashita, *Convergence Properties of the Regularized Newton Method for the Unconstrained Nonconvex Optimization*, Applied Mathematics and Optimization, 62 (2010), pp. 27–46.

[20] K. Ueda, and N. Yamashita, *A regularized Newton method without line search for unconstrained optimization*, Technical Report, Department of Applied Mathematics and Physics, Kyoto University, 2009.

[21] K. Ueda, *Studies on Regularized Newton-type methods for unconstrained minimization problems and their global complexity bounds*, Doctoral thesis, Department of Applied Mathematics and Physics, Kyoto University, 2012.