Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Limited-Memory M5 Quasi-Newton Method

Milan De

Dr. B. R. Ambedkar National Institute of Technology Jalandhar

Manoj Kumar

manojk@nitj.ac.in

Dr BR Ambedkar National Institute of Technology https://orcid.org/0000-0002-8942-1754

Research Article

Keywords: M5 method, L-BFGS, quasi-Newton methods, limited-memory optimization, nonlinear optimization, global and R-linear convergence, unconstrained optimization

Posted Date: November 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11775564/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

A Limited-Memory M5 Quasi-Newton Method

Milan De[1] and Manoj Kumar[1*]

[1]Department of Mathematics and Computing, Dr B R Ambedkar National Institute of Technology Jalandhar, Jalandhar, Punjab, India.

*Corresponding author(s). E-mail(s): manojk@nitj.ac.in;
Contributing authors: miland.mc.25@nitj.ac.in;

Abstract

We propose M5–LBFGS, a limited-memory quasi-Newton method for large-scale unconstrained optimization that combines the three-stage, fifth-order M5 method for nonlinear equations with the limited-memory BFGS (L-BFGS) method to approximate the inverse of the Hessian. A common inverse Hessian matrix with a damping parameter is used across all three stages. The damping parameter is selected by backtracking to satisfy the sufficient descent and Armijo conditions. Under the stated assumptions, we prove that the backtracking procedure terminates finitely and establish global convergence. We further establish R-linear convergence for the proposed M5–LBFGS method, trading the M5 method's original fifth-order local convergence for a convergence guarantee compatible with the limited-memory quasi-Newton framework. Numerical experiments on large-scale CUTEst problems show that M5–LBFGS solves several problems that L-BFGS implementations fail to solve and substantially reduces the number of iterations on many test problems. However, these reductions do not consistently translate into lower central processing unit (CPU) time or fewer gradient evaluations due to M5's multiple stages. The results, therefore, indicate that M5–LBFGS can be attractive when iteration reduction and robustness are important, while its computational advantage depends on the problem and the cost of gradient evaluations.

Keywords: M5 method, L-BFGS, quasi-Newton methods, limited-memory optimization, nonlinear optimization, global and R-linear convergence, unconstrained optimization

1 Introduction

Consider the unconstrained optimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable with large number of variables n.

Quasi-Newton methods are widely used to solve such problems. Quasi-Newton methods approximate the inverse of the Hessian matrix using first-order gradient differences without computing the Hessian matrix itself [1]. BFGS is the best-known of these methods, for which global convergence with inexact line searches was proved by Powell [2]. Its use for large-scale optimization is limited by the inherent structure of BFGS, which stores a dense $n \times n$ matrix. Nocedal [3] therefore proposed the limited-memory BFGS method (L-BFGS), which stores only the last m pairs of steps and gradient differences, so that each iteration costs $O(mn)$ operations; compact representations of these matrices were given by Byrd, Nocedal, and Schnabel [4]. Liu and Nocedal [5] proved that L-BFGS converges for uniformly convex problems, and it is of rate R-linear. L-BFGS is now the state-of-the-art method for large smooth problems, and many variants with convergence guarantees have been proposed, for example, regularized L-BFGS methods [6].

Many studies have proposed multistep methods for finding roots of higher-order nonlinear equations [7]. Due to the success of these methods, researchers have proposed methods in the field of optimization [8]. Arroyo, Cordero, and Torregrosa [9] have proposed the M5 method, a fifth-order method with semi-local convergence. It is a three-stage method for solving nonlinear equations.

M5 method can be used in optimization since a minimizer of f solves the nonlinear equation $\nabla f(x) = 0$. Ford and Moghrabi [10] used information from several previous iterates in quasi-Newton updates. Chakraborty and Panda [11] combined a two-stage scheme with the BFGS update and observed fewer iterations than BFGS. These methods, however, use a dense matrix and were evaluated only on small problems. More recently, Moghrabi and Hassan [12] proposed a limited-memory multi-step BFGS method. Cordero et al. [13] developed higher-order quasi-Newton methods and also investigated their limited-memory variants. Our approach differs in the particular way the three-stage M5 scheme is incorporated into the L-BFGS framework. We use the same inverse-Hessian approximation across all three M5 stages, together with a damping parameter selected to satisfy both sufficient descent and Armijo conditions.

This motivated us to investigate the following research questions: Does combining the three-stage M5 scheme with L-BFGS preserve the method's higher-order convergence? What would be the time and space complexities of the modified higher-order method?

These research questions will have two major challenges. The M5 method relies on intermediate updates and does not guarantee movement in the descent direction. We also cannot use the usual line search to ensure the update is in the descent direction.

Another important point is that the M5 method exhibits semilocal convergence; thus, we need to establish global convergence.

The main contributions of this paper are as follows:

1. We propose limited memory M5 method, hereafter referred as M5–LBFGS. The same L-BFGS matrix is used across the three M5 stages, with a damping parameter. An update step is accepted if it satisfies a sufficient descent condition and the Armijo condition [14].
2. Under Assumption 3.1, we prove that the backtracking procedure terminates after a uniformly bounded number of trials, that each iteration decreases f by an amount proportional to $\|\nabla f(x_k)\|^2$, and that the iterates converge to the global minimizer.
3. We establish the R-linear convergence of the proposed method, matching the rate proven by Liu and Nocedal [5] for L-BFGS.
4. We evaluate the proposed method on two sets of unconstrained CUTEst benchmark problems [15], across two dimensionality regimes with ($1000 \leq n \leq 2996$ and with $5000 \leq n \leq 10000$). Numerical results indicate that M5–LBFGS is more robust than the standard L-BFGS method, requiring about 40% fewer iterations than L-BFGS (Wolfe). However, because it evaluates multiple gradients per iteration, L-BFGS still outperforms the proposed method in total CPU time. Additionally, we analyze the effects of the memory size variations on overall performance.

The rest of the paper is organized as follows. Section 2 describes the original and proposed damped M5–LBFGS method and its acceptance rule. Section 3 discusses the convergence analysis. Section 4 states the algorithm and Section 5 analyses its cost. Section 6 reports the numerical results, Section 7 discusses them, and Section 8 concludes the paper. Throughout the paper, $\|\cdot\|$ denotes the Euclidean norm for vectors and the induced 2-norm for matrices.

2 Damped M5–LBFGS method

2.1 The damped M5 step

Arroyo, Cordero, and Torregrosa [9] proposed a higher-order M5 scheme to solve a nonlinear system $F(x) = 0$, $F : \mathbb{R}^n \to \mathbb{R}^n$, with three stages per iteration. The update rule at iteration k is given below. The scheme uses the same inverse Jacobian $J(x_k)^{-1}$ across the three steps in a given iteration k.

$$y_k = x_k - J(x_k)^{-1}F(x_k), \tag{2}$$

$$z_k = y_k - 5\,J(x_k)^{-1}F(y_k), \tag{3}$$

$$x_{k+1} = z_k - \tfrac{1}{5}\,J(x_k)^{-1}\big(-16\,F(y_k) + F(z_k)\big). \tag{4}$$

The first stage is a Newton step, while the second and third stages use the function values at the intermediate points y_k and z_k. The second and third steps are designed as corrections to the predictor. Under the assumptions established in the original analysis, the M5 method achieves fifth-order convergence [9, Theorem 1].

The necessary condition for optimality of problem (1) requires finding the solution of the equation $\nabla f(x) = 0$. The M5 scheme cannot be applied directly to large-scale unconstrained optimization, as it uses a complete Jacobian Matrix to solve a system of linear equations. We write

$$g_k := \nabla f(x_k). \tag{5}$$

We replace the inverse Hessian by the L-BFGS approximation H_k, described in Section 2.2. The same matrix H_k is used across the three stages. Once we compute g_k, the M5 scheme requires two additional gradient evaluations, at y_k and z_k, without explicitly computing the Hessian.

When the initial point is chosen far from the optimal solution, the full three-stage step update may fail to reduce the objective function value. To guarantee steady progress, we introduce a damping parameter $\alpha \in (0, 1]$ into each three-stage. For a given α, we define

$$y_k(\alpha) = x_k - \alpha H_k g_k, \tag{6}$$

$$z_k(\alpha) = y_k(\alpha) - 5\alpha H_k \nabla f\big(y_k(\alpha)\big), \tag{7}$$

$$\hat{x}_{k+1}(\alpha) = z_k(\alpha) - \tfrac{\alpha}{5} H_k\big(-16\,\nabla f\big(y_k(\alpha)\big) + \nabla f\big(z_k(\alpha)\big)\big), \tag{8}$$

and the search direction

$$d_k(\alpha) := \hat{x}_{k+1}(\alpha) - x_k. \tag{9}$$

For $\alpha = 1$, these equations recover the three-stage M5 construction applied to $F = \nabla f$, with H_k replacing the exact inverse Hessian $\nabla^2 f(x_k)^{-1}$. The accepted iterate is

$$x_{k+1} = x_k + d_k(\alpha_k), \tag{10}$$

where the damping parameter α_k is chosen by the backtracking rule of Section 2.3.

This damping strategy differs from a conventional line search. As α varies, the intermediate points $y_k(\alpha)$ and $z_k(\alpha)$ change accordingly, which subsequently modifies the gradients evaluated at these points. Because α alters the underlying geometry of the stages rather than just scaling the final step, our approach bypasses the traditional line minimization problem ($\min_\alpha f(x_k+\alpha d_k)$). We have formulated the acceptance rule and the convergence analysis presented in Section 3 in terms of the path-dependent search $d_k(\alpha)$.

2.2 The L-BFGS matrix

Once we accept the step size, the following correction pairs are formed:

$$s_k := x_{k+1} - x_k, \qquad r_k := g_{k+1} - g_k. \tag{11}$$

The pair is stored only if

$$s_k^T r_k \;\geq\; \varepsilon_{\text{curv}} \|s_k\|^2, \tag{12}$$

where $\varepsilon_{\text{curv}} > 0$ is a small fixed number, and subsequently update the list of m most recently stored pairs. The equation (12) also ensures the positive definiteness of $s_k^T r_k$, which may not otherwise hold.

Let $(s^{(1)}, r^{(1)}), \ldots, (s^{(p)}, r^{(p)})$, $0 \le p \le m$, be the pairs stored at iteration k, previous m observations from current pair. The matrix H_k is obtained by applying p inverse BFGS updates to a multiple of the identity [3, 5]:

$$H^{(0)} = \gamma_k I,$$
$$H^{(i)} = \big(I - \rho_i s^{(i)} r^{(i)T}\big) H^{(i-1)} \big(I - \rho_i r^{(i)} s^{(i)T}\big) + \rho_i s^{(i)} s^{(i)T}, \qquad \rho_i = \frac{1}{s^{(i)T} r^{(i)}}, \quad (13)$$

for $i = 1, \ldots, p$, and $H_k := H^{(p)}$. For the initial scaling we use the standard choice [5, strategy M3], [1, Sect. 7.2]

$$\gamma_k = \frac{s^{(p)T} r^{(p)}}{r^{(p)T} r^{(p)}} \quad (p \ge 1), \qquad \gamma_k = \gamma_{\text{init}} > 0 \quad (p = 0). \quad (14)$$

The value γ_{init} is needed only while no pair has been stored, in particular at $k = 0$. If $\|g_0\| = 0$, the algorithm terminates at x_0; under Assumption 3.1, this is the unique global minimizer. Otherwise, we take $\gamma_{\text{init}} = 1/\|g_0\|$, so that the step size in the first stage of the first iteration is $-g_0/\|g_0\|$.

A product $H_k v$ is computed by the two-loop recursion [3], [1, Algorithm 7.4] in $O(mn)$ operations, using only the last m stored pairs instead of storing the full inverse-Hessian approximation H_k. The stored pairs do not change within the iteration as the memory is being updated only after a step has been accepted. Therefore, the same matrix H_k is used in all three stages of (6)–(8) and for every step size α. The product $H_k g_k$ is independent of α and computed only once per iteration, whereas $H_k \nabla f(y_k(\alpha))$ and $H_k \nabla f(z_k(\alpha))$ depends on the choice of α and must therefore be recomputed for each trial value of α.

2.3 The acceptance rule

The damping parameter is chosen by backtracking. We fix constants

$$c_1 \in (0, 1), \qquad c_d \in (0, 3), \qquad \beta \in (0, 1). \quad (15)$$

At a given iteration with $g_k \neq 0$, the trial values $\alpha = 1, \beta, \beta^2, \ldots$ are tested in this order, and all three stages (6)–(9) are computed for each trial value. The first trial value that satisfies

$$-g_k^T d_k(\alpha) \;\ge\; c_d\, \alpha\, g_k^T H_k g_k, \quad (16)$$
$$f\big(x_k + d_k(\alpha)\big) \;\le\; f(x_k) + c_1\, g_k^T d_k(\alpha) \quad (17)$$

is accepted and denoted by α_k.

Condition (17) is the Armijo sufficient-decrease condition [14], applied to the trial point $x_k + d_k(\alpha)$ generated by the three-stage M5 procedure. Condition (16) provides a sufficient-descent requirement: the decrease predicted by $g_k^T d_k(\alpha)$ must be at least a fixed fraction of $\alpha\, g_k^T H_k g_k$, which represents the decrease predicted by the first stage alone. This condition requires no additional computation of $H_k g_k$, since this quantity

is already available from the first stage. It also involves no unknown constant and remains unchanged when f is multiplied by a positive number. Condition (16) plays an important role in the convergence analysis. Without it, a trial value such as $\alpha = 1$ could be accepted even when $g_k^T d_k(\alpha)$ is negative but arbitrarily close to zero, in which case (17) would guarantee almost no decrease in f. With (16), every accepted step size decreases f by an amount proportional to $\|g_k\|^2$, elaborated in the Section 3.

We use backtracking rather than a Wolfe line search because the trial steps $d_k(\alpha)$ vary along a curve as α changes (Section 2.1). Since the Wolfe curvature condition is formulated along a fixed direction, the retention test (12) is used here to ensure the required positive curvature.

Parameter restrictions.

The restrictions in (15) serve different purposes in the acceptance rule and convergence analysis of the proposed method.

- The conditions $c_1 > 0$ and $c_d > 0$ together ensures positive decrease in f. In particular, (16), together with the positive definiteness of H_k, ensures that the accepted step is a descent step, while (17) guarantees a decrease in the objective function. Section 3 shows the necessity of both conditions to ensure the sufficient decrease.
- The condition $\beta \in (0, 1)$ ensures that the trial values remain positive and approach zero. Consequently, the backtracking procedure can reach sufficiently small step sizes for which acceptance is guaranteed.
- The restriction $c_1 < 1$ is required to establish that (17) holds for sufficiently small α. In particular, the proof in Section 3 relies on $1 - c_1 > 0$. The stronger restriction $c_1 < \frac{1}{2}$, commonly used in quasi-Newton methods, is relevant only to establishing eventual acceptance of the unit step near the solution.
- The upper bound $c_d < 3$ follows from the behavior of the M5 step for small values of the damping parameter. As shown in Section 3,

$$d_k(\alpha) = -3\alpha H_k g_k + O(\alpha^2 \|g_k\|).$$

  The coefficient -3 arises from the combined contributions -1, -5 and $+3$ of the three stages. Hence, for $g_k \neq 0$ and sufficiently small α,

$$\frac{-g_k^T d_k(\alpha)}{\alpha\, g_k^T H_k g_k} = 3 + O(\alpha).$$

  Thus, when $c_d < 3$, condition (16) is guaranteed to hold for all sufficiently small positive α.

Under the assumptions stated in Section 3, these restrictions are sufficient to guarantee that the backtracking procedure terminates in a finite number of trials for every parameter choice satisfying (15).

3 Convergence analysis

In this section, we present the convergence analysis of our proposed method. The iterates are given by $x_{k+1} = x_k + d_k(\alpha_k)$, where $d_k(\alpha)$ is the damped M5 step defined by (6)–(9). The matrix H_k is the L-BFGS matrix defined by (13)–(14), with memory size $m \geq 1$, curvature threshold $\varepsilon_{\text{curv}} > 0$, and initial scaling $\gamma_{\text{init}} > 0$. At each iteration, α_k is the first trial value in the sequence $1, \beta, \beta^2, \ldots$ that satisfies (16) and (17).

For the convergence analysis, we consider Algorithm 2 of Section 4 with $\varepsilon_{\text{tol}} = 0$ and $J_{\max} = \infty$, so that the method terminates only when $g_k = 0$. Following the analysis of L-BFGS by Liu and Nocedal [5, Sect. 7], we first establish bounds on the L-BFGS matrices, then prove a sufficient-decrease estimate of the form

$$f(x_k) - f(x_{k+1}) \geq \Gamma \|g_k\|^2, \qquad \Gamma > 0,$$

and finally establish global convergence and R-linear convergence.

For symmetric matrices, $A \preceq B$ means that $B - A$ is positive semidefinite. We use the following elementary facts. If A is symmetric positive definite, then $\lambda_{\min}(A) \|v\|^2 \leq v^T A v \leq \lambda_{\max}(A) \|v\|^2$ for all v, $\|A\| = \lambda_{\max}(A)$, and the eigenvalues of A^{-1} are the reciprocals of those of A. For $u, v \in \mathbb{R}^n$, $\|uv^T\| = \|u\| \|v\|$ and $uu^T \preceq \|u\|^2 I$.

Assumption 3.1 (Objective function) *Let $\mathcal{L}_0 := \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$.*

(i) The function f is twice continuously differentiable on $\mathbb{R}^n$.
(ii) The level set $\mathcal{L}_0$ is convex.
(iii) There exist constants $0 < m_f \leq L_f < \infty$ such that

$$m_f \|v\|^2 \;\leq\; v^T \nabla^2 f(x)\, v \;\leq\; L_f \|v\|^2 \qquad \text{for all } x \in \mathcal{L}_0,\ v \in \mathbb{R}^n. \tag{18}$$

3.1 Consequences of Assumption 3.1

The results of this subsection are standard consequences of strong convexity (see, e.g., [1, Ch. 2] and [16]). We include short proofs because Assumption 3.1 imposes the bounds (18) only on the set $\mathcal{L}_0$ and not on all of $\mathbb{R}^n$, and to keep the paper self-contained.

Lemma 3.2 (Averaged Hessian) *Under Assumption 3.1, for all $x, y \in \mathcal{L}_0$ there is a symmetric matrix $G = G(x, y)$ such that*

$$\nabla f(y) - \nabla f(x) = G(y - x), \qquad m_f I \preceq G \preceq L_f I. \tag{19}$$

Proof Since $\mathcal{L}_0$ is convex, $\phi(t) := x + t(y - x) \in \mathcal{L}_0$ for $t \in [0, 1]$. The map $t \mapsto \nabla f(\phi(t))$ is continuously differentiable with derivative $\nabla^2 f(\phi(t))(y - x)$. By the fundamental theorem of calculus, applied componentwise,

$$\nabla f(y) - \nabla f(x) = \int_0^1 \nabla^2 f(\phi(t))(y - x)\, dt = G(y - x), \qquad G := \int_0^1 \nabla^2 f(\phi(t))\, dt,$$

where the integral is taken entrywise. G is symmetric because every $\nabla^2 f(\phi(t))$ is symmetric. For $v \in \mathbb{R}^n$ we have $v^T G v = \int_0^1 v^T \nabla^2 f(\phi(t)) v \, dt$, and by (18) the integrand lies in $[m_f \|v\|^2, L_f \|v\|^2]$ for every t. Integrating gives $m_f \|v\|^2 \le v^T G v \le L_f \|v\|^2$. □

Lemma 3.3 *Suppose Assumption 3.1 holds. Then for all $x, y \in \mathcal{L}_0$:*

(a) $\|\nabla f(y) - \nabla f(x)\| \le L_f \|y - x\|$;
(b) $f(y) \le f(x) + \nabla f(x)^T (y - x) + \frac{L_f}{2} \|y - x\|^2$;
(c) $f(y) \ge f(x) + \nabla f(x)^T (y - x) + \frac{m_f}{2} \|y - x\|^2$;
(d) $\|\nabla f(y) - \nabla f(x)\| \ge m_f \|y - x\|$;
(e) the set $\mathcal{L}_0$ is compact, and f has exactly one stationary point x^* in $\mathcal{L}_0$. Moreover, x^* is the unique global minimizer of f over $\mathbb{R}^n$, and $f^* := f(x^*) = \min_{x \in \mathbb{R}^n} f(x)$;
(f) for every $x \in \mathcal{L}_0$,

$$\|x - x^*\| \le R_0 := \Big[\frac{2(f(x_0) - f^*)}{m_f}\Big]^{1/2}, \qquad \|\nabla f(x)\| \le \bar{g} := L_f R_0.$$

Proof Let $h := y - x$ and let G be the matrix of Lemma 3.2, so that $\nabla f(y) - \nabla f(x) = Gh$. Since $m_f > 0$, G is positive definite with eigenvalues in $[m_f, L_f]$.

(a) We have $\|Gh\| \le \|G\| \, \|h\| = \lambda_{\max}(G) \, \|h\| \le L_f \|h\|$.

(b), (c) Put $\psi(t) := f(x + th)$, $t \in [0, 1]$; the segment lies in $\mathcal{L}_0$ because $\mathcal{L}_0$ is convex. Then ψ is twice continuously differentiable with $\psi'(0) = \nabla f(x)^T h$ and $\psi''(t) = h^T \nabla^2 f(x + th) h \in [m_f \|h\|^2, L_f \|h\|^2]$ by (18). Taylor's formula with integral remainder gives

$$f(y) = \psi(1) = \psi(0) + \psi'(0) + \int_0^1 (1 - t) \, \psi''(t) \, dt.$$

Since $\int_0^1 (1 - t) \, dt = \frac{1}{2}$, the integral lies between $\frac{m_f}{2} \|h\|^2$ and $\frac{L_f}{2} \|h\|^2$, which proves (b) and (c).

(d) By the Cauchy–Schwarz inequality, $\|Gh\| \, \|h\| \ge h^T G h \ge m_f \|h\|^2$. If $h \ne 0$, divide by $\|h\|$; if $h = 0$, (d) is trivial.

(e) The set $\mathcal{L}_0$ is nonempty because $x_0 \in \mathcal{L}_0$. Moreover, $\mathcal{L}_0$ is closed, since f is continuous. We now show that $\mathcal{L}_0$ is bounded. Let $y \in \mathcal{L}_0$. Inequality (c), with $x = x_0$ and $g_0 = \nabla f(x_0)$, gives

$$f(y) \ge f(x_0) + g_0^T (y - x_0) + \frac{m_f}{2} \|y - x_0\|^2 .$$

Using the Cauchy–Schwarz inequality and the fact that $f(y) \le f(x_0)$, we obtain

$$\frac{m_f}{2} \|y - x_0\|^2 \le \|g_0\| \, \|y - x_0\| .$$

Hence

$$\|y - x_0\| \le \frac{2 \|g_0\|}{m_f},$$

where the conclusion is immediate if $y = x_0$. Thus $\mathcal{L}_0$ is bounded and therefore compact. Since f is continuous, it attains its minimum over $\mathcal{L}_0$ at some point $x^* \in \mathcal{L}_0$.

We now show that x^* is a global minimizer of f over $\mathbb{R}^n$. For every $x \in \mathcal{L}_0$, we have $f(x^*) \le f(x)$ by the choice of x^*. On the other hand, if $x \notin \mathcal{L}_0$, then $f(x) > f(x_0) \ge f(x^*)$. Thus, $f(x^*) \le f(x)$ for all $x \in \mathbb{R}^n$. Since f is differentiable, it follows that $\nabla f(x^*) = 0$.

Suppose now that $\bar{x} \in \mathcal{L}_0$ also satisfies $\nabla f(\bar{x}) = 0$. Applying inequality (d) to x^* and $\bar{x}$, we obtain

$$m_f \left\|\bar{x} - x^*\right\| \le \left\|\nabla f(\bar{x}) - \nabla f(x^*)\right\| = 0.$$

Since $m_f > 0$, we conclude that $\bar{x} = x^*$. Hence x^* is the unique stationary point of f in $\mathcal{L}_0$. Finally, let $\hat{x}$ be any global minimizer of f on $\mathbb{R}^n$. Then $f(\hat{x}) = f^* \le f(x_0)$, so $\hat{x} \in \mathcal{L}_0$, and $\nabla f(\hat{x}) = 0$. Therefore $\hat{x} = x^*$, and x^* is the unique global minimizer, with $f^* = f(x^*) = \min_{x \in \mathbb{R}^n} f(x)$.

(f) Let $x \in \mathcal{L}_0$. Applying (c) with the base point $x^* \in \mathcal{L}_0$ and using $\nabla f(x^*) = 0$,

$$\frac{m_f}{2} \left\|x - x^*\right\|^2 \le f(x) - f^* \le f(x_0) - f^*,$$

so $\|x - x^*\| \le R_0$. Then (a) gives $\|\nabla f(x)\| = \|\nabla f(x) - \nabla f(x^*)\| \le L_f \|x - x^*\| \le L_f R_0 = \bar{g}$. □

Since $g_0 \neq 0$, Lemma 3.3(c) gives $f(x_0) > f^*$, so $R_0 > 0$ and $\bar{g} > 0$.

The next lemma is the only place where the behaviour of f outside $\mathcal{L}_0$ enters. It replaces any assumption on the intermediate points of the M5 step by a consequence of the continuity of $\nabla^2 f$ and the compactness of $\mathcal{L}_0$.

Lemma 3.4 (Local regularity near $\mathcal{L}_0$) *Suppose Assumption 3.1 holds. Fix any $\rho > 0$ (for instance $\rho = 1$) and put*

$$\mathcal{K}_\rho := \{x \in \mathbb{R}^n : \ \operatorname{dist}(x, \mathcal{L}_0) \le \rho\}, \qquad L' := \max\Big\{L_f, \ \max_{x \in \mathcal{K}_\rho} \left\|\nabla^2 f(x)\right\|\Big\}.$$

Then $\mathcal{K}_\rho$ is compact, $L' < \infty$, and for all $x \in \mathcal{L}_0$ and all $w \in \mathbb{R}^n$ with $\|w - x\| \le \rho$:

(a) $\|\nabla f(w) - \nabla f(x)\| \le L' \|w - x\|$;
(b) $f(w) \le f(x) + \nabla f(x)^T (w - x) + \frac{L'}{2} \|w - x\|^2$.

Moreover, $L' \ge L_f$.

Proof By Lemma 3.3(e), $\mathcal{L}_0$ is compact, so $\mathcal{K}_\rho = \mathcal{L}_0 + \overline{B}(0, \rho)$ is the sum of two compact sets and hence compact. Since $\nabla^2 f$ is continuous, $x \mapsto \left\|\nabla^2 f(x)\right\|$ is continuous and attains a finite maximum on $\mathcal{K}_\rho$; thus $L' < \infty$, and $L' \ge L_f$ by definition.

Let $x \in \mathcal{L}_0$ and $\|w - x\| \le \rho$, and put $\phi(t) := x + t(w - x)$ for $t \in [0, 1]$. Then $\|\phi(t) - x\| \le \rho$, so $\phi(t) \in \mathcal{K}_\rho$ and $\left\|\nabla^2 f(\phi(t))\right\| \le L'$ for all $t \in [0, 1]$. As in the proof of Lemma 3.2,

$$\nabla f(w) - \nabla f(x) = \int_0^1 \nabla^2 f(\phi(t))(w - x)\, dt,$$

and therefore $\|\nabla f(w) - \nabla f(x)\| \le L' \|w - x\|$, which is (a). For (b), put $\psi(t) := f(\phi(t))$. Then $\psi''(t) = (w - x)^T \nabla^2 f(\phi(t))(w - x) \le L' \|w - x\|^2$, and Taylor's formula with integral remainder, as in the proof of Lemma 3.3(b),(c), gives (b). □

Remark 3.5 The parameter $\rho > 0$ can be chosen arbitrarily. The constants depend on this choice: since K_ρ increases with ρ, the constants L', and hence C, C_s and C_ρ, are nondecreasing functions of ρ. However, the last term in (26), namely $\rho/(C_\rho \bar{g})$, need not be monotone in ρ. If (18) holds on all of $\mathbb{R}^n$, then one can take $L' = L_f$, independently of ρ. Consequently, C, C_s and C_ρ are also independent of ρ, and $\rho/(C_\rho \bar{g}) \to \infty$ as $\rho \to \infty$. Thus, the last term in (26) eventually ceases to be restrictive.

Lemma 3.6 (Polyak–Łojasiewicz inequality [16]) *Under Assumption 3.1, for every $x \in \mathcal{L}_0$,*

$$\|\nabla f(x)\|^2 \;\geq\; 2m_f\big(f(x) - f^*\big). \tag{20}$$

Proof By Lemma 3.3(e), the minimizer x^* belongs to $\mathcal{L}_0$. Fix any $x \in \mathcal{L}_0$, and set $t := \|x^* - x\|$ and $\eta := \|\nabla f(x)\|$. Applying Lemma 3.3(c) with $y = x^*$, and using $f(x^*) = f^*$, we obtain

$$f^* \;\geq\; f(x) + \nabla f(x)^T(x^* - x) + \frac{m_f}{2}t^2.$$

Using the Cauchy–Schwarz inequality,

$$\nabla f(x)^T(x^* - x) \geq -\|\nabla f(x)\|\,\big\|x^* - x\big\| = -\eta t.$$

It follows that

$$f^* \;\geq\; f(x) - \eta t + \frac{m_f}{2}t^2.$$

Completing the square gives

$$-\eta t + \frac{m_f}{2}t^2 = \frac{m_f}{2}\left(t - \frac{\eta}{m_f}\right)^2 - \frac{\eta^2}{2m_f} \geq -\frac{\eta^2}{2m_f}.$$

Hence,

$$f^* \geq f(x) - \frac{\eta^2}{2m_f},$$

which, after rearrangement, yields

$$\|\nabla f(x)\|^2 \geq 2m_f\big(f(x) - f^*\big).$$

Since $x \in \mathcal{L}_0$ was arbitrary, the inequality holds throughout $\mathcal{L}_0$. This completes the proof. □

3.2 Bounds on the L-BFGS matrices

We first show that every iterate stays in $\mathcal{L}_0$ and the Hessian H_k is positive definite. This is straightforward using equation (12) and the acceptance conditions (16)–(17).

Lemma 3.7 *Let iteration k be reached. Then for every $j \leq k$:*

(a) $x_j \in \mathcal{L}_0$*;*
(b) H_j *is symmetric positive definite;*

and for every $j < k$:

(c) $s_j \neq 0$ *and* $f(x_{j+1}) < f(x_j)$*.*

Moreover, every pair (s, r) stored at iteration k satisfies $s \neq 0$ and $s^T r \geq \varepsilon_{\mathrm{curv}}\|s\|^2 > 0$.

Proof We prove (a) and (b) by induction on j, and obtain (c) along the way. For $j = 0$, $x_0 \in \mathcal{L}_0$ and no pair is stored, so $H_0 = \gamma_{\mathrm{init}} I$ is symmetric positive definite.

Suppose that (a) and (b) hold for some $j < k$. Since $g_j \neq 0$ and H_j is positive definite, $g_j^T H_j g_j > 0$. By (16), $g_j^T d_j(\alpha_j) \leq -c_d \alpha_j g_j^T H_j g_j < 0$, and in particular $s_j = d_j(\alpha_j) \neq 0$. By (17),

$$f(x_{j+1}) \leq f(x_j) + c_1 g_j^T d_j(\alpha_j) < f(x_j) \leq f(x_0),$$

so $x_{j+1} \in \mathcal{L}_0$ and $f(x_{j+1}) < f(x_j)$. This proves (c) at j and (a) at $j+1$.

We now have to prove (b) at $j+1$. Every pair (s, r) stored at iteration $j+1$ was formed at some iteration $i \leq j$, so $s = s_i \neq 0$ by the preceding paragraph applied at i, and the pair passed the retention test (12), that is, $s^T r \geq \varepsilon_{\text{curv}} \|s\|^2 > 0$. In particular $r \neq 0$, and $\rho = (s^T r)^{-1}$ is well defined. If $p \geq 1$, the scaling $\gamma_{j+1} = s^{(p)T} r^{(p)} / r^{(p)T} r^{(p)}$ is positive; if $p = 0$, $\gamma_{j+1} = \gamma_{\text{init}} > 0$. Hence $H^{(0)} = \gamma_{j+1} I$ is symmetric positive definite.

Suppose that $H^{(i-1)}$ is symmetric positive definite, and write $(s, r) = (s^{(i)}, r^{(i)})$ and $\rho_i = (s^T r)^{-1}$. By (13), $H^{(i)}$ is symmetric. Moreover, for any $v \in \mathbb{R}^n$,

$$v^T H^{(i)} v = w^T H^{(i-1)} w + \rho_i (s^T v)^2, \qquad w = (I - \rho_i r s^T) v.$$

Both terms on the right-hand side are nonnegative. If their sum were zero, positive definiteness of $H^{(i-1)}$ would imply $w = 0$, while the second term would give $s^T v = 0$. The latter equality yields $w = v$, which implies $v = 0$. Thus $H^{(i)}$ is positive definite. Applying this argument to each stored pair shows that $H_{j+1} = H^{(p)}$ is symmetric positive definite, which is (b) at $j+1$. The last assertion of the lemma was shown in the same argument. □

Lemma 3.8 (Curvature pairs) *Under Assumption 3.1, let iteration $k+1$ be reached. Then the pair (11) satisfies*

$$s_k^T r_k \;\geq\; m_f \|s_k\|^2, \qquad \|r_k\| \;\leq\; L_f \|s_k\|, \tag{21}$$

and, since $s_k \neq 0$,

$$m_f \;\leq\; \frac{r_k^T r_k}{s_k^T r_k} \;\leq\; L_f. \tag{22}$$

Proof By Lemma 3.7, x_k and x_{k+1} belong to $\mathcal{L}_0$ and $s_k \neq 0$. Applying Lemma 3.2, we have $r_k = G_k s_k$, where G_k is symmetric and satisfies $m_f I \preceq G_k \preceq L_f I$. We can write

$$s_k^T r_k = s_k^T G_k s_k \geq m_f \|s_k\|^2,$$

and

$$\|r_k\| = \|G_k s_k\| \leq \|G_k\| \, \|s_k\| \leq L_f \|s_k\|.$$

These establish the inequalities in (21).

The square root $G_k^{1/2}$ exists and is symmetric positive definite because G_k is symmetric positive definite. Set $w := G_k^{1/2} s_k$, so that $w \neq 0$. Using $r_k = G_k s_k$, we obtain

$$s_k^T r_k = w^T w, \qquad r_k^T r_k = w^T G_k w.$$

It follows that

$$\frac{r_k^T r_k}{s_k^T r_k} = \frac{w^T G_k w}{w^T w}.$$

The expression on the right is the Rayleigh quotient of G_k at the nonzero vector w. Since all eigenvalues of G_k lie in $[m_f, L_f]$, this quotient belongs to the same interval, which proves (22). □

Remark 3.9 Under Assumption 3.1, Lemma 3.8 establishes the positive curvature of every pair formed by the method, *i.e.*, $s_k^T r_k \geq m_f \|s_k\|^2 > 0$. If $\varepsilon_{\text{curv}} \leq m_f$, all pairs pass the test and are retained among the m most recent pairs. Otherwise, certain pairs may be rejected, but subsequent pairs satisfying the condition continue to be accepted until the history of m pairs is complete.

Lemma 3.10 (Uniform bounds on H_k) *Suppose Assumption 3.1 holds, and let*

$$\underline{\gamma} := \min\Big\{\gamma_{\text{init}}, \frac{1}{L_f}\Big\}, \qquad \bar{\gamma} := \max\Big\{\gamma_{\text{init}}, \frac{1}{m_f}\Big\}, \qquad q := 1 + \frac{L_f}{m_f},$$

$$\mu := \Big(\frac{1}{\underline{\gamma}} + m\,L_f\Big)^{-1}, \qquad M := q^{2m}\,\bar{\gamma} + \frac{q^{2m}-1}{(q^2-1)\,m_f}. \tag{23}$$

Then for every reached iteration k the matrix H_k is symmetric positive definite and

$$\mu\,\|v\|^2 \;\le\; v^T H_k v \;\le\; M\,\|v\|^2 \qquad \text{for all } v \in \mathbb{R}^n. \tag{24}$$

In consequence, $\mu \le M$, $\|H_k v\| \le M\,\|v\|$ for all v, and $g_k^T H_k g_k \ge \mu\,\|g_k\|^2$.

Proof Fix a reached iteration k. By Lemma 3.7, H_k is symmetric positive definite. Let $(s^{(1)}, r^{(1)}), \dots, (s^{(p)}, r^{(p)})$, where $0 \le p \le m$, denote the pairs stored at iteration k, ordered from the oldest to the latest. Each stored pair was formed at some iteration $j < k$, so $x_j, x_{j+1} \in \mathcal{L}_0$ by Lemma 3.7, and Lemma 3.8 shows that (21)–(22) hold for every stored pair, with $s^{(i)} \neq 0$ and $s^{(i)T} r^{(i)} > 0$. If $p \ge 1$, the scaling parameter is determined by the most recently stored pair and satisfies

$$\gamma_k = \frac{s^{(p)T} r^{(p)}}{r^{(p)T} r^{(p)}} \in \left[\frac{1}{L_f}, \frac{1}{m_f}\right].$$

If $p = 0$, then $\gamma_k = \gamma_{\text{init}}$. Hence $\underline{\gamma} \le \gamma_k \le \bar{\gamma}$ in either case.

To establish the upper bound, write $(s, r) = (s^{(i)}, r^{(i)})$ and $\rho_i = (s^T r)^{-1}$. Observe from (21) that

$$\rho_i \|s\|\|r\| = \frac{\|s\|\|r\|}{s^T r} \le \frac{L_f}{m_f}, \qquad \rho_i \|s\|^2 = \frac{\|s\|^2}{s^T r} \le \frac{1}{m_f}.$$

Hence,

$$\|I - \rho_i s r^T\| \le 1 + \rho_i \|s\|\|r\| \le q,$$

and the same bound holds for $I - \rho_i r s^T$. Taking norms in (13), we obtain

$$a_i := \|H^{(i)}\| \le q^2 a_{i-1} + \frac{1}{m_f}, \qquad i = 1, \dots, p.$$

Iterating this inequality gives

$$a_p \le q^{2p} a_0 + \frac{1}{m_f} \sum_{j=0}^{p-1} q^{2j} = q^{2p}\gamma_k + \frac{q^{2p}-1}{(q^2-1)m_f}.$$

Since $q > 1$, $p \le m$, and $\gamma_k \le \bar{\gamma}$, it follows that $\|H_k\| = a_p \le M$. As H_k is symmetric positive definite, we conclude that

$$v^T H_k v \le M\|v\|^2 \qquad \text{for all } v \in \mathbb{R}^n.$$

For the lower bound, let $B^{(i)} = (H^{(i)})^{-1}$, which exists because each $H^{(i)}$ is positive definite (Lemma 3.7). The corresponding direct BFGS update is

$$B^{(i)} = B^{(i-1)} - \frac{B^{(i-1)} s s^T B^{(i-1)}}{s^T B^{(i-1)} s} + \frac{r r^T}{s^T r}, \qquad (s, r) = (s^{(i)}, r^{(i)}).$$

The subtracted matrix is positive semidefinite, since it can be written as $uu^T/(s^T B^{(i-1)} s)$ with $u = B^{(i-1)} s$. Furthermore, (22) implies

$$\frac{r r^T}{s^T r} \preceq \frac{\|r\|^2}{s^T r} I \preceq L_f I.$$

It follows that

$$B^{(i)} \preceq B^{(i-1)} + L_f I.$$

Using $B^{(0)} = \gamma_k^{-1} I \preceq \underline{\gamma}^{-1} I$ and summing over the p updates, we obtain

$$H_k^{-1} = B^{(p)} \preceq \left(\frac{1}{\underline{\gamma}} + pL_f \right) I \preceq \left(\frac{1}{\underline{\gamma}} + mL_f \right) I = \frac{1}{\mu} I.$$

Therefore, every eigenvalue of H_k^{-1} is at most $1/\mu$, so every eigenvalue of H_k is at least μ. Hence

$$v^T H_k v \geq \mu \|v\|^2 \qquad \text{for all } v \in \mathbb{R}^n.$$

Combining the upper and lower bounds establishes (24); in particular $\mu \leq M$. Finally, $\|H_k v\| \leq \|H_k\| \|v\| \leq M \|v\|$, while setting $v = g_k$ in the lower bound gives $g_k^T H_k g_k \geq \mu \|g_k\|^2$. The proof is complete. □

The constants μ and M depend only on m, γ_{init}, m_f and L_f, and not on k.

3.3 The damped M5 step

Let $\rho > 0$, L' and $\bar{g}$ be as in Lemmas 3.4 and 3.3(f), and let μ, M be given by (23). Define

$$C := ML'(3M + M^2 L'), \quad C_s := 3M + C, \quad C_\rho := \max\{6M + 5M^2 L', C_s\}, \quad \alpha_\rho := \min\left\{1, \frac{\rho}{C_\rho \bar{g}}\right\}. \tag{25}$$

Lemma 3.11 *Suppose Assumption 3.1 holds. Then, for every reached iteration k with $g_k \neq 0$ and every trial value $\alpha \in (0, \alpha_\rho]$ considered during backtracking, the points $y_k(\alpha)$, $z_k(\alpha)$ and $x_k + d_k(\alpha)$ lie in the ball $\{w : \|w - x_k\| \leq \rho\}$, and the following statements hold:*

(a) $d_k(\alpha) = -3\alpha H_k g_k + e_k(\alpha)$, *where* $\|e_k(\alpha)\| \leq C\alpha^2 \|g_k\|$;
(b) $\|d_k(\alpha)\| \leq C_s \alpha \|g_k\|$;
(c) $-g_k^T d_k(\alpha) \geq \left(3 - \frac{C\alpha}{\mu}\right) \alpha \, g_k^T H_k g_k$.

Proof For simplicity, write

$$y = y_k(\alpha), \qquad z = z_k(\alpha), \qquad H = H_k, \qquad g = g_k,$$

and define

$$a := \nabla f(y) - g, \qquad b := \nabla f(z) - g.$$

By Lemma 3.7, $x_k \in \mathcal{L}_0$, so $\|g\| \leq \bar{g}$ by Lemma 3.3(f). By Lemma 3.10,

$$\|Hv\| \leq M \|v\| \qquad \text{for every } v \in \mathbb{R}^n.$$

Recall that $\alpha \leq \alpha_\rho \leq 1$. We prove the three assertions separately.

(a). From (6), we have $y - x_k = -\alpha H g$, which implies

$$\|y - x_k\| \leq \alpha M \|g\| \leq C_\rho \alpha \bar{g} \leq \rho,$$

because $M \leq C_\rho$ and $\alpha \leq \rho/(C_\rho \bar{g})$. Hence Lemma 3.4(a) applies with $x = x_k$ and $w = y$, and gives

$$\|a\| = \|\nabla f(y) - \nabla f(x_k)\| \leq L' \|y - x_k\| \leq ML' \alpha \|g\|.$$

Next, from (7) and $\nabla f(y) = g + a$, it follows that

$$z - x_k = (y - x_k) - 5\alpha H \nabla f(y) = -\alpha H g - 5\alpha H(g + a) = -6\alpha H g - 5\alpha H a.$$

Therefore,

$$\|z - x_k\| \le 6\alpha M \|g\| + 5\alpha M \|a\| \le 6\alpha M \|g\| + 5M^2 L' \alpha^2 \|g\| \le (6M + 5M^2 L')\alpha \|g\|,$$

where the last inequality follows from $\alpha \le 1$. In particular $\|z - x_k\| \le C_\rho \alpha \bar{g} \le \rho$, and Lemma 3.4(a) gives

$$\|b\| = \|\nabla f(z) - \nabla f(x_k)\| \le L' \|z - x_k\| \le L'(6M + 5M^2 L')\alpha \|g\|.$$

We now use (8) and (9). Substituting $\nabla f(y) = g + a$ and $\nabla f(z) = g + b$ gives

$$\begin{aligned} d_k(\alpha) &= (z - x_k) - \frac{\alpha}{5} H\big(-16(g + a) + (g + b)\big) \\ &= -6\alpha H g - 5\alpha H a + 3\alpha H g + \frac{16}{5}\alpha H a - \frac{1}{5}\alpha H b \\ &= -3\alpha H g - \alpha H \left(\frac{9}{5}a + \frac{1}{5}b\right). \end{aligned}$$

Define

$$e_k(\alpha) := -\alpha H \left(\frac{9}{5}a + \frac{1}{5}b\right).$$

Then $d_k(\alpha) = -3\alpha H g + e_k(\alpha)$. Furthermore, the bounds on a and b yield

$$\begin{aligned} \|e_k(\alpha)\| &\le \alpha M \left(\frac{9}{5}\|a\| + \frac{1}{5}\|b\|\right) \\ &\le \alpha^2 M L' \left(\frac{9}{5}M + \frac{1}{5}(6M + 5M^2 L')\right)\|g\| \\ &= M L'(3M + M^2 L')\alpha^2 \|g\| \\ &= C\alpha^2 \|g\|. \end{aligned}$$

Thus, both the representation of $d_k(\alpha)$ and the required bound on $e_k(\alpha)$ hold, proving (a).

(b). Using the representation established in (a), together with $\|Hg\| \le M\|g\|$, $\|e_k(\alpha)\| \le C\alpha^2\|g\|$ and $\alpha \le 1$, we obtain

$$\begin{aligned} \|d_k(\alpha)\| &= \|-3\alpha H g + e_k(\alpha)\| \\ &\le 3\alpha \|Hg\| + \|e_k(\alpha)\| \\ &\le 3M\alpha \|g\| + C\alpha^2 \|g\| \\ &\le (3M + C)\alpha \|g\| = C_s \alpha \|g\|. \end{aligned}$$

Consequently, $\|d_k(\alpha)\| \le C_s \alpha \bar{g} \le C_\rho \alpha \bar{g} \le \rho$, so $x_k + d_k(\alpha)$ lies in the ball of radius ρ about x_k. This proves (b) and completes the statement about the three points.

(c). Taking the inner product of the representation in (a) with $-g$ gives

$$-g^T d_k(\alpha) = 3\alpha g^T H g - g^T e_k(\alpha).$$

By the Cauchy–Schwarz inequality and the bound established in (a),

$$\begin{aligned} -g^T d_k(\alpha) &\ge 3\alpha g^T H g - \|g\| \, \|e_k(\alpha)\| \\ &\ge 3\alpha g^T H g - C\alpha^2 \|g\|^2. \end{aligned}$$

The lower spectral bound in Lemma 3.10 states that $g^T H g \geq \mu \|g\|^2$. Since $\mu > 0$, this implies $\|g\|^2 \leq \frac{1}{\mu} g^T H g$. Substituting this inequality into the preceding estimate, we obtain

$$\begin{aligned} -g^T d_k(\alpha) &\geq 3\alpha g^T H g - \frac{C\alpha^2}{\mu} g^T H g \\ &= \left(3 - \frac{C\alpha}{\mu}\right) \alpha g^T H g. \end{aligned}$$

Replacing g and H by g_k and H_k, respectively, proves (c). □

Part (a) shows that, up to a remainder of order $\alpha^2 \|g_k\|$, the damped M5 displacement is three times the damped L-BFGS displacement $-\alpha H_k g_k$. This first-order relation explains the appearance of the constant 3 in the restriction $c_d < 3$ in (15).

3.4 Finite backtracking and sufficient decrease

Theorem 3.12 (Finite termination of the backtracking) *Suppose Assumption 3.1 holds, $c_1 \in (0,1)$, $c_d \in (0,3)$ and $\beta \in (0,1)$, and let μ, C, C_s, C_ρ be given by (23) and (25). Define*

$$\bar{\alpha} := \min\Big\{1,\ \frac{(3-c_d)\mu}{C},\ \frac{2(1-c_1)c_d\mu}{L'C_s^2},\ \frac{\rho}{C_\rho\,\bar{g}}\Big\} \in (0,1], \qquad \bar{j} := \min\{j \in \mathbb{N}_0 : \beta^j \leq \bar{\alpha}\}. \tag{26}$$

Then, at every reached iteration k with $g_k \neq 0$, the following statements hold:

(a) Every tested trial value $\alpha \leq \bar{\alpha}$ satisfies (16) and (17).
(b) The backtracking procedure terminates after at most $\bar{j}$ reductions; in particular, $\alpha_k \in \{1, \beta, \ldots, \beta^{\bar{j}}\}$.

Proof (a) Let $\alpha \leq \bar{\alpha}$ be a trial value tested at iteration k. Then $\alpha \leq \alpha_\rho$, so Lemma 3.11 applies. By the definition of $\bar{\alpha}$,

$$\alpha \leq \frac{(3-c_d)\mu}{C},$$

and hence $3 - C\alpha/\mu \geq c_d$. By Lemma 3.10, H_k is positive definite, so $g_k^T H_k g_k > 0$. It follows from Lemma 3.11(c) that

$$-g_k^T d_k(\alpha) \geq \left(3 - \frac{C\alpha}{\mu}\right) \alpha g_k^T H_k g_k \geq c_d\, \alpha\, g_k^T H_k g_k.$$

Thus, the sufficient-descent condition (16) is satisfied.

To establish the Armijo condition (17), note that $x_k \in \mathcal{L}_0$ by Lemma 3.7, and $\|d_k(\alpha)\| \leq \rho$ by Lemma 3.11. Applying Lemma 3.4(b) with $x = x_k$ and $w = x_k + d_k(\alpha)$, and then Lemma 3.11(b), we obtain

$$\begin{aligned} &f(x_k + d_k(\alpha)) - f(x_k) - c_1 g_k^T d_k(\alpha) \\ &\leq (1 - c_1)\, g_k^T d_k(\alpha) + \frac{L'}{2} \|d_k(\alpha)\|^2 \\ &\leq (1 - c_1)\, g_k^T d_k(\alpha) + \frac{L'}{2} C_s^2 \alpha^2 \|g_k\|^2 . \end{aligned}$$

The descent inequality just established, together with Lemma 3.10, gives

$$g_k^T d_k(\alpha) \leq -c_d \alpha g_k^T H_k g_k \leq -c_d \mu \alpha \|g_k\|^2 .$$

Substituting this bound into the preceding estimate yields

$$f(x_k + d_k(\alpha)) - f(x_k) - c_1 g_k^T d_k(\alpha)$$
$$\leq \alpha \|g_k\|^2 \left(-(1-c_1)c_d\mu + \frac{L' C_s^2}{2}\alpha \right) \leq 0,$$

where the final inequality follows from $\alpha \leq 2(1-c_1)c_d\mu/(L'C_s^2)$. Therefore, the Armijo condition (17) also holds.

(b) Since $0 < \beta < 1$, we have $\beta^j \to 0$ as $j \to \infty$. Consequently, the integer $\bar{j}$ in (26) is finite. If any of the trial values $1, \beta, \ldots, \beta^{\bar{j}-1}$ is accepted, the backtracking procedure terminates before $\bar{j}$ reductions. Otherwise, $\beta^{\bar{j}}$ is tested and, by part (a), satisfies both acceptance conditions because $\beta^{\bar{j}} \leq \bar{\alpha}$. It must therefore be accepted. Hence, the backtracking procedure terminates after at most $\bar{j}$ reductions, with $\alpha_k \in \{1, \beta, \ldots, \beta^{\bar{j}}\}$. □

Remark 3.13 Lemma 3.11 and Theorem 3.12(a) are used only for trial values $\alpha \leq \bar{\alpha}$, for which all trial points stay within distance ρ of x_k. For larger trial values the points $y_k(\alpha)$ and $z_k(\alpha)$ may lie far from $\mathcal{L}_0$, where no information on f is assumed. This does not matter: such a trial value is accepted only if it satisfies the computable tests (16) and (17), and the analysis below uses only these two inequalities for the accepted value.

Proposition 3.14 (Sufficient decrease) *Under the hypotheses of Theorem 3.12, define*

$$\Gamma := c_1 \, c_d \, \mu \, \beta \, \bar{\alpha} > 0. \tag{27}$$

Then, for every reached iteration k with $g_k \neq 0$, the following statements hold:

(a) $\alpha_k \geq \beta \, \bar{\alpha}$;
(b) $f(x_k) - f(x_{k+1}) \geq c_1\big(-g_k^T d_k(\alpha_k)\big) \geq c_1 c_d \, \alpha_k \, g_k^T H_k g_k \geq \Gamma \, \|g_k\|^2$.

Proof (a) By Theorem 3.12(b), the accepted step size can be written as $\alpha_k = \beta^j$ for some $j \in \{0, \ldots, \bar{j}\}$. If $j = 0$, then $\alpha_k = 1$, and hence $\alpha_k \geq \beta\bar{\alpha}$, since $\bar{\alpha} \leq 1$ and $\beta < 1$.

Suppose next that $j \geq 1$. The preceding trial value β^{j-1} was tested and rejected. By Theorem 3.12(a), every tested trial value not exceeding $\bar{\alpha}$ satisfies the acceptance conditions. Consequently, the rejected trial value must satisfy $\beta^{j-1} > \bar{\alpha}$. Multiplying by β gives

$$\alpha_k = \beta^j > \beta\bar{\alpha}.$$

Thus, in either case, $\alpha_k \geq \beta\bar{\alpha}$.

(b) Applying the Armijo condition (17) at the accepted step size α_k yields

$$f(x_k) - f(x_{k+1}) \geq c_1\big(-g_k^T d_k(\alpha_k)\big).$$

The sufficient-descent condition (16), also evaluated at $\alpha = \alpha_k$, then gives

$$-g_k^T d_k(\alpha_k) \geq c_d \, \alpha_k \, g_k^T H_k g_k.$$

Finally, Lemma 3.10 implies $g_k^T H_k g_k \geq \mu \|g_k\|^2$. Combining this bound with part (a), we obtain

$$\begin{aligned} f(x_k) - f(x_{k+1}) &\geq c_1 c_d \, \alpha_k \, g_k^T H_k g_k \\ &\geq c_1 c_d \mu \, \alpha_k \, \|g_k\|^2 \\ &\geq c_1 c_d \mu \beta\bar{\alpha} \, \|g_k\|^2 \\ &= \Gamma \, \|g_k\|^2 . \end{aligned}$$

This proves the sufficient-decrease estimate and completes the proof. □

3.5 Global convergence

Theorem 3.15 (Global convergence) *Suppose Assumption 3.1 holds and let* $c_1 \in (0,1)$, $c_d \in (0,3)$ *and* $\beta \in (0,1)$, *and* Γ *be defined by* (27). *If the method terminates at iteration* k, *then* $x_k = x^*$. *Otherwise, the sequence* $\{x_k\}$ *generated by the method satisfies the following properties:*

(a) Every iteration is well defined, and $f(x_{k+1}) < f(x_k)$ *for all* $k \geq 0$. *Moreover,* $x_k \in \mathcal{L}_0$ *for all* $k \geq 0$.

(b) The squared gradients are summable:

$$\sum_{k=0}^{\infty} \|g_k\|^2 \leq \frac{f(x_0) - f^*}{\Gamma},$$

and consequently $\|g_k\| \to 0$ *as* $k \to \infty$.

(c) For every $k \geq 0$,

$$\|x_k - x^*\| \leq \frac{\|g_k\|}{m_f}.$$

Hence, $x_k \to x^*$ *as* $k \to \infty$.

Proof If the method terminates at some iteration k, then $g_k = 0$. Since $x_k \in \mathcal{L}_0$ by Lemma 3.7, Lemma 3.3(e) implies that $x_k = x^*$. We may therefore assume that the method does not terminate, so that $g_k \neq 0$ at every iteration.

(a). We show by induction that every iteration k is reached. Iteration 0 is reached. Suppose that iteration k is reached. Then $x_k \in \mathcal{L}_0$ and H_k satisfies (24) by Lemmas 3.7 and 3.10. By Theorem 3.12(b), the backtracking procedure terminates after finitely many trials, so the accepted step size α_k and the point x_{k+1} are well defined, and iteration $k+1$ is reached. Proposition 3.14(b) gives

$$f(x_{k+1}) \leq f(x_k) - \Gamma \|g_k\|^2 < f(x_k) \leq f(x_0),$$

where the strict inequality follows from $\Gamma > 0$ and $g_k \neq 0$. Thus $x_{k+1} \in \mathcal{L}_0$, and the objective values strictly decrease.

(b). By Proposition 3.14(b), for every $k \geq 0$,

$$\Gamma \|g_k\|^2 \leq f(x_k) - f(x_{k+1}).$$

Summing these inequalities from $k = 0$ to N and using the telescoping sum, we obtain

$$\begin{aligned} \Gamma \sum_{k=0}^{N} \|g_k\|^2 &\leq \sum_{k=0}^{N} \big(f(x_k) - f(x_{k+1})\big) \\ &= f(x_0) - f(x_{N+1}) \\ &\leq f(x_0) - f^*, \end{aligned}$$

where the last inequality follows from $f^* = \min_{x \in \mathbb{R}^n} f(x)$, as established in Lemma 3.3(e). Since $\Gamma > 0$, it follows that

$$\sum_{k=0}^{N} \|g_k\|^2 \leq \frac{f(x_0) - f^*}{\Gamma} \qquad \text{for every } N \geq 0.$$

The partial sums on the left are nondecreasing and bounded above. Therefore, the series $\sum_{k=0}^{\infty} \|g_k\|^2$ converges and satisfies the asserted bound. In particular, since its terms are nonnegative, we conclude that $\|g_k\|^2 \to 0$, and hence $\|g_k\| \to 0$.

(c). Since $x_k, x^* \in \mathcal{L}_0$ and $\nabla f(x^*) = 0$, Lemma 3.3(d) yields

$$\begin{aligned} \|g_k\| &= \|\nabla f(x_k) - \nabla f(x^*)\| \\ &\geq m_f \|x_k - x^*\|. \end{aligned}$$

Therefore,

$$\|x_k - x^*\| \leq \frac{\|g_k\|}{m_f}.$$

Since $m_f > 0$ and $\|g_k\| \to 0$ by part (b), we obtain $x_k \to x^*$. This completes the proof. □

3.6 R-linear convergence

Following [17, Ch. 9], a sequence $\{x_k\}$ converges to x^* *R-linearly* if $\limsup_{k\to\infty} \|x_k - x^*\|^{1/k} < 1$. This holds, in particular, if there are constants $K \geq 0$ and $\nu \in [0, 1)$ such that $\|x_k - x^*\| \leq K\nu^k$ for all k, since then $\|x_k - x^*\|^{1/k} \leq K^{1/k}\nu \to \nu$ when $K > 0$. The next theorem is the analogue of [5, Theorem 7.1] for M5–LBFGS.

Theorem 3.16 (R-linear convergence) *Suppose Assumption 3.1 holds, $c_1 \in (0, 1)$, $c_d \in (0, 3)$ and $\beta \in (0, 1)$. If the method stops at iteration k_0, set $x_k := x_{k_0}$ for all $k > k_0$. Let Γ be given by (27) and define*

$$\theta := 1 - 2m_f\Gamma. \tag{28}$$

Then $\theta \in (0, 1)$, and for every $k \geq 0$,

$$f(x_k) - f^* \leq \theta^k \big(f(x_0) - f^*\big), \tag{29}$$

$$\|x_k - x^*\| \leq \Big[\frac{2(f(x_0) - f^*)}{m_f}\Big]^{1/2} (\sqrt{\theta})^k, \tag{30}$$

$$\|g_k\| \leq L_f \Big[\frac{2(f(x_0) - f^*)}{m_f}\Big]^{1/2} (\sqrt{\theta})^k. \tag{31}$$

In particular, $\{x_k\}$ converges to x^* R-linearly.

Proof Put $\delta_k := f(x_k) - f^*$. By Theorem 3.15, $x_k \in \mathcal{L}_0$ and $\delta_k \geq 0$ for all k. If the method stops at iteration k_0, then $x_{k_0} = x^*$, and hence $x_k = x^*$ and $\delta_k = 0$ for all $k \geq k_0$. Thus, it suffices to establish the estimates before termination.

We first show that $\theta \in (0, 1)$. Since $m_f > 0$ and $\Gamma > 0$, we have $\theta < 1$. To establish the positivity of θ, we derive an upper bound for Γ. By (26),

$$\bar{\alpha} \leq \frac{2(1 - c_1)c_d\mu}{L'C_s^2}.$$

Moreover, by (25) and $\mu \leq M$ (Lemma 3.10),

$$C_s \geq 3M \geq 3\mu, \qquad \text{so} \qquad C_s^2 \geq 9\mu^2.$$

Using $\Gamma = c_1 c_d \mu\beta\bar{\alpha}$, we obtain

$$\Gamma \leq \frac{2c_1(1 - c_1)c_d^2\beta\mu^2}{L'C_s^2}$$

$$\leq \frac{2c_1(1-c_1)c_d^2\beta}{9L'}$$
$$< \frac{2\cdot\frac{1}{4}\cdot 9}{9L'} = \frac{1}{2L'},$$

where we used $c_1(1-c_1) \leq \frac{1}{4}$, $c_d < 3$, and $\beta < 1$. Since $L' \geq L_f$, it follows that

$$2m_f\Gamma < \frac{m_f}{L'} \leq \frac{m_f}{L_f} \leq 1,$$

which implies $\theta > 0$. Therefore, $\theta \in (0,1)$.

We next prove (29). For any iteration before termination, Proposition 3.14(b) and Lemma 3.6 (applicable since $x_k \in \mathcal{L}_0$) yield

$$\begin{aligned}\delta_k - \delta_{k+1} &= f(x_k) - f(x_{k+1}) \\ &\geq \Gamma \|g_k\|^2 \\ &\geq 2m_f\Gamma\,\delta_k.\end{aligned}$$

It follows that

$$\delta_{k+1} \leq (1-2m_f\Gamma)\delta_k = \theta\delta_k.$$

Applying this inequality successively gives

$$\delta_k \leq \theta^k\delta_0 = \theta^k\big(f(x_0) - f^*\big),$$

which proves (29). The estimate remains valid after termination because $\delta_k = 0$ for all $k \geq k_0$.

To prove (30), apply Lemma 3.3(c) with $x = x^*$ and $y = x_k$. Since $\nabla f(x^*) = 0$ and $f(x^*) = f^*$, we obtain

$$\frac{m_f}{2}\|x_k - x^*\|^2 \leq f(x_k) - f^* = \delta_k \leq \theta^k\delta_0.$$

Taking square roots gives

$$\|x_k - x^*\| \leq \Big[\frac{2(f(x_0)-f^*)}{m_f}\Big]^{1/2}(\sqrt{\theta})^k,$$

as required.

Finally, Lemma 3.3(a), together with $\nabla f(x^*) = 0$, implies

$$\|g_k\| = \|\nabla f(x_k) - \nabla f(x^*)\| \leq L_f\|x_k - x^*\|.$$

Combining this inequality with (30) establishes (31).

Since $\theta \in (0,1)$, we have $\sqrt{\theta} \in (0,1)$. Estimate (30) therefore gives constants $K \geq 0$ and $\nu = \sqrt{\theta} \in (0,1)$ such that

$$\|x_k - x^*\| \leq K\nu^k \qquad \text{for all } k \geq 0.$$

Hence $\{x_k\}$ converges to x^* R-linearly. □

Corollary 3.17 (Iteration bound) *Under the hypotheses of Theorem 3.16, suppose that $g_0 \neq 0$ and let $\varepsilon > 0$. Define*

$$K_\varepsilon := \max\left\{0,\ \left\lceil \frac{\ln\left(\dfrac{2L_f^2(f(x_0)-f^*)}{m_f\varepsilon^2}\right)}{\ln(1/\theta)} \right\rceil\right\}.$$

Then

$$\|g_k\| \leq \varepsilon \qquad \text{for every integer } k \geq K_\varepsilon.$$

Consequently, for any fixed problem, the number of iterations sufficient to attain $\|g_k\| \leq \varepsilon$ is $O(\log(1/\varepsilon))$ as $\varepsilon \to 0$.

Proof Since $g_0 \neq 0$, we have $x_0 \neq x^*$ and hence $f(x_0) > f^*$ by Lemma 3.3(c). The gradient estimate (31) gives

$$\|g_k\| \leq L_f \left[\frac{2(f(x_0) - f^*)}{m_f}\right]^{1/2} (\sqrt{\theta})^k.$$

Therefore, $\|g_k\| \leq \varepsilon$ is guaranteed whenever

$$\theta^k \leq \frac{m_f \varepsilon^2}{2L_f^2(f(x_0) - f^*)}.$$

Because $0 < \theta < 1$, taking logarithms and rearranging yields

$$k \geq \frac{\ln\left(\dfrac{2L_f^2(f(x_0) - f^*)}{m_f \varepsilon^2}\right)}{\ln(1/\theta)}.$$

Since k must be a nonnegative integer, the definition of K_ε gives the stated bound.

Finally, for a fixed problem, the quantities L_f, m_f, $f(x_0) - f^*$ and θ are independent of ε. The numerator in the iteration bound grows as $2\ln(1/\varepsilon) + O(1)$ as $\varepsilon \to 0$, whereas $\ln(1/\theta) > 0$ is constant. Hence $K_\varepsilon = O(\log(1/\varepsilon))$. □

Remark 3.18 Theorem 3.16 establishes an R-linear convergence result analogous to that of Liu and Nocedal [5, Theorem 7.1], under the same assumptions on the objective function f. Their analysis uses bounds on the trace and determinant of H_k^{-1} to control the angle between the L-BFGS search direction and the negative gradient $-g_k$. In contrast, the M5 step involves three gradient evaluations, with the same inverse-Hessian approximation H_k applied to the gradients at the intermediate points $y_k(\alpha)$ and $z_k(\alpha)$ and at the current iterate x_k. We therefore establish the convergence rate using the two-sided spectral bounds of Lemma 3.10, together with the sufficient-decrease estimate in Proposition 3.14 and the Polyak–Łojasiewicz inequality in Lemma 3.6.

The factor θ in Theorem 3.16 provides an explicit uniform contraction bound for the objective function error and guarantees R-linear convergence of the iterates. The constants $\bar{\alpha}$, Γ and θ depend on the Hessian bounds m_f and L_f on $\mathcal{L}_0$, the initial objective gap $f(x_0) - f^*$, the local Hessian bound L' and the chosen neighbourhood radius ρ, as well as the algorithmic parameters m, γ_{init}, c_1, c_d and β. These constants are independent of the iteration index k.

4 The M5–LBFGS algorithm

We now state the method of Section 2 as an algorithm. Algorithm 1 computes one damped M5 trial step, and Algorithm 2, which calls it, states the complete method. The analysis of Section 3 corresponds to $\varepsilon_{\text{tol}} = 0$ and $J_{\max} = \infty$, for which the method stops only when $g_k = 0$. In the experiments we use $\varepsilon_{\text{tol}} = 10^{-5}$, so that the iteration stops when $\|g_k\| < 10^{-5} \max\{1, \|x_k\|\}$, the stopping test of Liu and Nocedal [5, eq. (2.8)], and at most $J_{\max} = 60$ trial values are tested per iteration.

Algorithm 1 Damped M5 trial step at iteration k

Require: x_k, $u_0 = H_k g_k$, the stored pairs, a trial value $\alpha \in (0, 1]$

$y \leftarrow x_k - \alpha u_0$; $\quad u_1 \leftarrow H_k \nabla f(y)$ ▷ two-loop recursion

$z \leftarrow y - 5\alpha u_1$; $\quad u_2 \leftarrow H_k \nabla f(z)$ ▷ two-loop recursion

return $d_k(\alpha) \leftarrow z - \frac{\alpha}{5}(-16u_1 + u_2) - x_k$

Algorithm 2 M5–LBFGS

Require: x_0; $m \geq 1$; c_1, c_d, β as in (15); $\varepsilon_{\text{curv}}, \gamma_{\text{init}} > 0$; $\varepsilon_{\text{tol}} \geq 0$; $J_{\max}$

$g_0 \leftarrow \nabla f(x_0)$; no pairs stored

for $k = 0, 1, 2, \ldots$ **do**

  if $g_k = 0$ **or** $\|g_k\| < \varepsilon_{\text{tol}} \max\{1, \|x_k\|\}$ **then return** x_k

  $u_0 \leftarrow H_k g_k$; $\alpha \leftarrow 1$ ▷ two-loop recursion, H_k as in (13)–(14)

  for $j = 1, \ldots, J_{\max}$ **do**

    compute $d_k(\alpha)$ by Algorithm 1

    if $-g_k^T d_k(\alpha) \geq c_d\, \alpha\, g_k^T u_0$ **and** $f(x_k + d_k(\alpha)) \leq f(x_k) + c_1 g_k^T d_k(\alpha)$ **then** $\alpha_k \leftarrow \alpha$; **go to** line 11

    $\alpha \leftarrow \beta\alpha$

  end for

  return failure ▷ no trial value accepted

  $x_{k+1} \leftarrow x_k + d_k(\alpha_k)$; $g_{k+1} \leftarrow \nabla f(x_{k+1})$; $s_k \leftarrow x_{k+1} - x_k$; $r_k \leftarrow g_{k+1} - g_k$

  if $s_k^T r_k \geq \varepsilon_{\text{curv}} \|s_k\|^2$ **then** store (s_k, r_k) and keep only the m newest pairs

end for

A trial that produces a value that is not finite is rejected, and if all $J_{\max}$ trial values are rejected, the method stops and reports a failure; no other fallback is used. The analysis of Section 3 assumes $J_{\max} = \infty$. The number $\bar{j}$ of reductions in Theorem 3.12 is a worst-case bound: since $\bar{\alpha}$ contains high powers of L_f/m_f, it can be much larger than 60, so with $J_{\max} = 60$ the practical method may fail. With $\beta = \frac{1}{2}$, however, the last trial value is $2^{-59} \approx 1.7 \times 10^{-18}$, far below the unit roundoff, so further reductions are of little practical use. In our experiments an accepted step needed at most 52 reductions, and the backtracking failed only on PENALTY2 and SCOSINE, which no method solved (Section 6). The value $f(x_k + d_k(\alpha))$ is computed only if (16) holds, and at the accepted trial value it is reused as $f(x_{k+1})$. Hence an iteration with j trial values costs $2j + 1$ gradient evaluations, at most j function evaluations and $2j + 1$ products with H_k. When the unit value $\alpha = 1$ is accepted, this is three gradient evaluations and three products with H_k, compared with one of each for an L-BFGS iteration whose unit step is accepted.

5 Cost of the method

Let n be the number of variables and m the memory size. A product $H_k v$ computed by the two-loop recursion costs $O(mn)$ operations, and the stored pairs need $O(mn)$ storage [3], [1, Sect. 7.2].

Let j_k be the number of trial values tested at iteration k, including the accepted one. As noted after Algorithm 2, the iteration needs $2j_k + 1$ products with H_k, $2j_k + 1$ gradient evaluations and at most j_k function evaluations; all other operations cost $O(n)$ per trial. By Theorem 3.12(b), $j_k \le \bar{j} + 1$ for every k. Hence one iteration costs

$$O\big((\bar{j}+1)\,mn\big) \text{ operations} \quad \text{and} \quad O(\bar{j}+1) \text{ evaluations of } f \text{ and } \nabla f,$$

where the constant $\bar{j}$ of (26) depends on the constants of Assumption 3.1 but not on k. Together with Corollary 3.17, an iterate with $\|g_k\| \le \varepsilon$ is therefore reached after $O(mn \log(1/\varepsilon))$ operations and $O(\log(1/\varepsilon))$ evaluations of f and ∇f. These are the same orders as for L-BFGS. The constants, however, differ: when the unit value $\alpha_k = 1$ is accepted, one iteration of M5–LBFGS costs about as much as three iterations of L-BFGS shown in Section 6.

6 Numerical experiments

In this section we compare M5–LBFGS with L-BFGS on unconstrained problems from the CUTEst collection [15]. We report the number of iterations, the number of gradient evaluations and the CPU time, first for problems with 1000 to 2996 variables and then for problems with 5000 to 10000 variables. Moreover, we study the effect of the memory size for large-scale problems.

6.1 Test problems

We use the CUTEst collection of unconstrained optimization problems, which are twice continuously differentiable with user-defined dimensions. The problems can be accessed through PyCUTEst [18]. The initial point considered is the standard starting point of each problem. Four problems are excluded: for FLETCBV2 and MOREBV, the starting point already satisfies the stopping criterion; and FLETCBV3 and INDEF are unbounded below [19]. We form two test sets as follows:

- *Test set I* ($1000 \le n \le 2996$): This group consists of 98 problems defined at their smallest available dimension within this specific range.
- *Test set II* ($5000 \le n \le 10000$): This group consists of 66 problems defined at their largest available dimension in this range.

6.2 Methods, settings and computing environment

We compare three methods.

- **L-BFGS (Wolfe)**: the method of Liu and Nocedal [5]. The step length satisfies the strong Wolfe conditions with $c_1 = 10^{-4}$ and $c_2 = 0.9$; it is computed by the line

search of Moré and Thuente [20] as implemented in SciPy [21], starting from the unit step. Every correction pair is stored, since the Wolfe conditions give $s_k^T r_k > 0$.

- **L-BFGS (Armijo)**: This method uses the standard L-BFGS search direction $-H_k g_k$, with the step length selected by Armijo backtracking. We set $\beta = \frac{1}{2}$ and allow at most 60 trial step lengths. The correction pairs are retained using the same criterion (12) as in M5–LBFGS. Armijo line searches have also been studied in structured variants of L-BFGS [22, 23].
- **M5–LBFGS**: Algorithm 2 with $c_1 = c_d = 10^{-4}$, $\beta = \frac{1}{2}$ and $J_{\max} = 60$.

L-BFGS (Wolfe) serves as the standard reference method. L-BFGS (Armijo) is included as a control for comparison with M5–LBFGS. Both methods use the same backtracking parameters and correction-pair retention criterion, but L-BFGS (Armijo) uses the standard direction $-H_k g_k$ and requires only the Armijo condition, whereas M5–LBFGS uses the three-stage M5 procedure and requires both the sufficient-descent and Armijo conditions. Thus, the comparison evaluates their combined effect.

All methods use the memory size $m = 5$, a commonly used choice in recent studies of limited-memory methods on CUTEst [6, 24, 25]; other values are studied in Section 6.5. All methods use the initial matrix (14) with $\gamma_{\text{init}} = 1/\|g_0\|$. The two Armijo methods use $\varepsilon_{\text{curv}} = 10^{-10}$ in (12). A run is considered successful if it satisfies the stopping criterion

$$\|g_k\| < 10^{-5} \max\{1, \|x_k\|\},$$

which is the stopping test of Liu and Nocedal [5, eq. (2.8)]. A run is considered unsuccessful if it does not satisfy the stopping criterion within 50,000 iterations, if the Wolfe line search fails, or if all 60 trial values in the backtracking procedure are rejected. Iteration limits of a comparable order are used in other CUTEst studies; for example, Kanzow and Steck [25] allow up to 10^5 iterations.

The number of gradient evaluations includes the gradients evaluated at the intermediate M5 points and at rejected trial values. CPU time is measured as process time. CPU-time measurements may exhibit run-to-run variability; therefore, repeated timing runs are used to obtain a more robust estimate of the computational cost. For each problem–method pair whose initial run converges, two additional runs are performed, giving three runs in total, and the median CPU time of the runs that converge is reported. If the initial run does not converge, no repeat runs are performed, and its CPU time is recorded once. All methods are implemented in Python 3.13.18 with NumPy 2.4.6 [26], pandas 2.2.3, Matplotlib 3.10.0, SciPy 1.16.3 [21], and use PyCUTEst 1.8.2 with CUTEst 2.7.1 and SIFDecode 3.1.1 (gfortran 13.3.0). The experiments were run on Google Colab, on an Intel Xeon CPU at 2.20 GHz (one thread) with 12.7 GB of memory under Ubuntu 24.04.5.

Performance profiles.

We use the performance profiles of Dolan and Moré [27]. For a problem p and a method s, let $t_{p,s}$ be the number of iterations, the number of gradient evaluations or the CPU time, with $t_{p,s} = \infty$ if the run failed. Then

$$r_{p,s} = \frac{t_{p,s}}{\min_{s'} t_{p,s'}}, \qquad \rho_s(\tau) = \frac{\big|\{p : r_{p,s} \le \tau\}\big|}{n_p}, \qquad \tau \ge 1, \tag{32}$$

where n_p is the number of problems solved by at least one method. Thus $\rho_s(1)$ is the fraction of problems on which method s is the best, and $\rho_s(\tau)$ for large τ is the fraction of problems that it solves.

6.3 Results on test set I ($1000 \le n \le 2996$)

Table 1 summarizes the results, Figs. 1–3 show the performance profiles, and Table 2 lists every run that did not converge, with the reason. The complete problem-by-problem results are provided in Appendix A.

Table 1 Test set I (98 problems, $1000 \le n \le 2996$). Top: number of problems solved and statistics over the 84 problems solved by all three methods. Bottom: pairwise better/worse counts and geometric mean ratios (M5–LBFGS / comparator; values below 1 favour M5–LBFGS).

Method	Solved	Iterations mean / median	Gradient evaluations mean / median	CPU time (s) median [Q1, Q3]
L-BFGS (Wolfe)	84/98	1414.4 / 137.5	1499.0 / 146.5	0.0301 [0.00717, 0.321]
L-BFGS (Armijo)	87/98	1578.5 / 132	1579.5 / 133	0.029 [0.00656, 0.284]
M5–LBFGS	89/98	746.7 / 74.5	3617.6 / 345	0.0642 [0.0127, 0.429]
M5–LBFGS compared with:		better / worse	(geometric mean of the ratio M5–LBFGS / method)	
L-BFGS (Wolfe)		78 / 2 (0.56)	1 / 83 (2.16)	7 / 77 (1.71)
L-BFGS (Armijo)		82 / 2 (0.51)	3 / 81 (2.25)	5 / 79 (1.87)

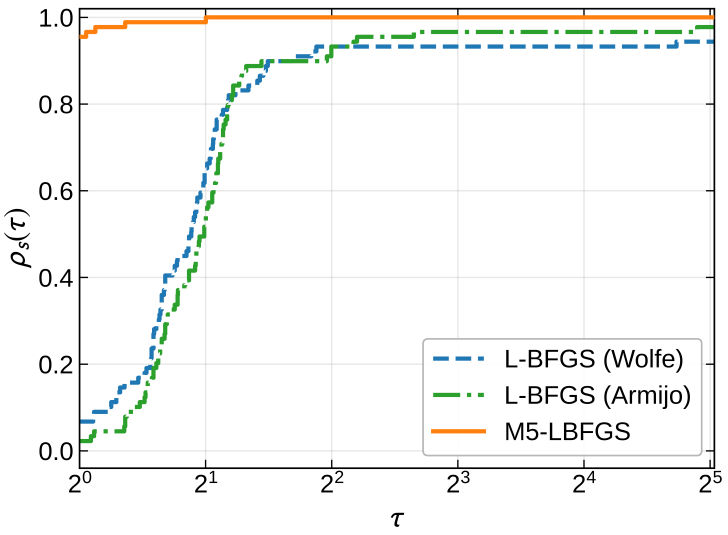


Fig. 1 Test set I: performance profiles for the number of iterations

Table 2 Test set I: all runs that did not converge (34 runs on 14 problems), with their failure reasons and iteration counts. Iteration limit reached: 50000 iterations without satisfying the stopping test; Wolfe line search failed: no step satisfying the strong Wolfe conditions was found; Backtracking failed: all 60 trial values were rejected.

Problem	n	Method	Reason	Iterations
BDQRTIC	1000	L-BFGS (Wolfe)	Wolfe line search failed	118
CURLY10	1000	L-BFGS (Wolfe)	Wolfe line search failed	8358
CURLY20	1000	L-BFGS (Wolfe)	Wolfe line search failed	8207
		L-BFGS (Armijo)	Iteration limit reached	50000
CURLY30	1000	L-BFGS (Wolfe)	Wolfe line search failed	8221
MODBEALE	2000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
NONMSQRT	1024	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
PENALTY2	1000	L-BFGS (Wolfe)	Wolfe line search failed	0
		L-BFGS (Armijo)	Backtracking failed	28
		M5–LBFGS	Backtracking failed	11
SBRYBND	1000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SCOSINE	1000	L-BFGS (Wolfe)	Wolfe line search failed	2
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Backtracking failed	32169
SCURLY10	1000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SCURLY20	1000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SCURLY30	1000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SPINLS	1082	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SSCOSINE	1000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000

Robustness. For this test set, M5–LBFGS solves 89 problems, L-BFGS (Armijo) 87, and L-BFGS (Wolfe) 84. M5–LBFGS solves every problem solved by either L-BFGS variant. Nine problems remain unsolved by all three methods. CURLY20 and MODBEALE are solved by M5–LBFGS but not by either L-BFGS variant; on MODBEALE, M5–LBFGS converges in 218 iterations, whereas both L-BFGS variants reach the iteration limit.

Iterations. Among the 84 problems solved by all three methods, M5–LBFGS requires fewer iterations than L-BFGS (Wolfe) on 78, more on 2 (COSINE and SINQUAD2, by 2 and 4 iterations, respectively), and the same number on the remaining 4. The geometric mean of the iteration ratios (M5–LBFGS / L-BFGS (Wolfe)) is 0.56. The comparison with L-BFGS (Armijo) gives a geometric mean ratio of 0.51, indicating that the reduction in iteration counts is also observed with Armijo backtracking. In

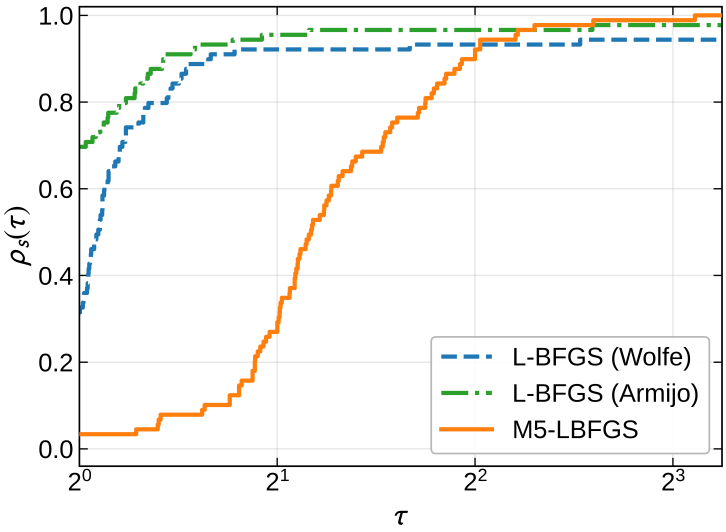


Fig. 2 Test set I: performance profiles for the number of gradient evaluations

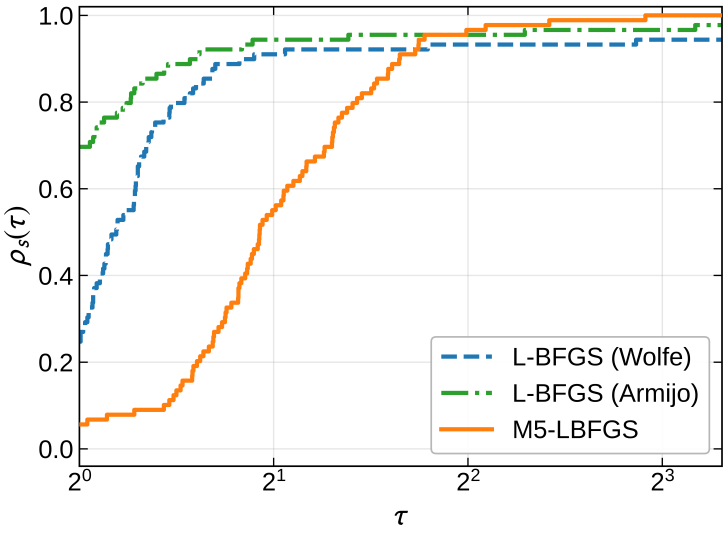


Fig. 3 Test set I: performance profiles for the CPU time

the performance profile of Fig. 1, M5–LBFGS requires the fewest iterations (including ties) on 85 of the 89 problems solved by at least one method (about 96%).

Gradient evaluations and CPU time. Each M5–LBFGS iteration requires at least three gradient evaluations. Overall, M5–LBFGS uses about 2.2 times as many gradient evaluations as L-BFGS (Wolfe) (Fig. 2). The geometric mean CPU-time ratio (M5–LBFGS / L-BFGS (Wolfe)) is 1.71, and M5–LBFGS is faster on only 7 of the 84

problems solved by all three methods. On CHAINWOO, however, M5–LBFGS requires 148 iterations and 0.11 s, compared with 3932 iterations and 0.82 s for L-BFGS (Wolfe).

Final values. On some nonconvex problems, the methods terminate at different stationary points. For example, on CHAINWOO, the final objective values are 4.6 (M5–LBFGS), 246 (L-BFGS (Wolfe)), and 455 (L-BFGS (Armijo)). The stopping criterion is relative to $\|x_k\|$; therefore, a large $\|x_k\|$ may allow termination even when $\|g_k\|$ is well above 10^{-5}. This occurs, for example, on QUARTC, DQRTIC, and FLETCHBV. The final objective values and gradient norms for all runs are available in the accompanying code repository.

6.4 Results on test set II ($5000 \leq n \leq 10000$)

The results on test set II are summarized in Table 3, Figs. 4–6 show the performance profiles, and Table 4 lists every run that did not converge. The complete problem-by-problem results are provided in Appendix A. They confirm the results on test set I.

Table 3 Test set II (66 problems, $5000 \leq n \leq 10000$). Top: number of solved problems, and statistics over the 59 problems solved by all three methods. Bottom: as in Table 1.

Method	Solved	Iterations mean / median	Gradient evaluations mean / median	CPU time (s) median [Q1, Q3]
L-BFGS (Wolfe)	60/66	810.0 / 149	883.1 / 176	0.192 [0.0288, 0.686]
L-BFGS (Armijo)	61/66	909.3 / 181	910.3 / 182	0.194 [0.0313, 0.555]
M5–LBFGS	63/66	437.6 / 102	2188.0 / 525	0.352 [0.0579, 1.2]
M5–LBFGS compared with:		better / worse	(geometric mean of the ratio M5–LBFGS / method)	
L-BFGS (Wolfe)		55 / 4 (0.62)	1 / 58 (2.40)	3 / 56 (1.97)
L-BFGS (Armijo)		57 / 2 (0.55)	0 / 59 (2.40)	1 / 58 (2.01)

Robustness. For this test set, M5–LBFGS solves 63 problems, L-BFGS (Armijo) 61, and L-BFGS (Wolfe) 60. M5–LBFGS solves every problem solved by either L-BFGS variant. Three problems (SCURLY10, SCURLY20, and SCURLY30) remain unsolved by all three methods. The Wolfe line search fails on BDQRTIC, FREUROTH, and SINQUAD; M5–LBFGS solves SINQUAD in 18 iterations. On SPARSINE, L-BFGS (Armijo) reaches the iteration limit, while M5–LBFGS requires 19,909 iterations, compared with 46,603 for L-BFGS (Wolfe).

Iterations. On the 59 problems solved by all three methods, M5–LBFGS requires fewer iterations than L-BFGS (Wolfe) on 55 problems and more on 4 (BOX, COSINE, SROSENBR, and VAREIGVL). The geometric mean iteration ratios (M5–LBFGS / comparator) are 0.62 against L-BFGS (Wolfe) and 0.55 against L-BFGS (Armijo). The reduction in iterations is slightly smaller than on test set I.

Gradient evaluations and CPU time. M5–LBFGS requires about 2.4 times as many gradient evaluations and about twice the CPU time as the L-BFGS variants. It is faster on 3 problems than L-BFGS (Wolfe) and on 1 problem than L-BFGS (Armijo).

Table 4 Test set II: all runs that did not converge (14 runs on 7 problems), with the reason and the number of iterations performed (reasons as in Table 2).

Problem	n	Method	Reason	Iterations
BDQRTIC	5000	L-BFGS (Wolfe)	Wolfe line search failed	236
FREUROTH	5000	L-BFGS (Wolfe)	Wolfe line search failed	19
SCURLY10	10000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SCURLY20	10000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SCURLY30	10000	L-BFGS (Wolfe)	Iteration limit reached	50000
		L-BFGS (Armijo)	Iteration limit reached	50000
		M5–LBFGS	Iteration limit reached	50000
SINQUAD	10000	L-BFGS (Wolfe)	Wolfe line search failed	26
		L-BFGS (Armijo)	Iteration limit reached	50000
SPARSINE	10000	L-BFGS (Armijo)	Iteration limit reached	50000

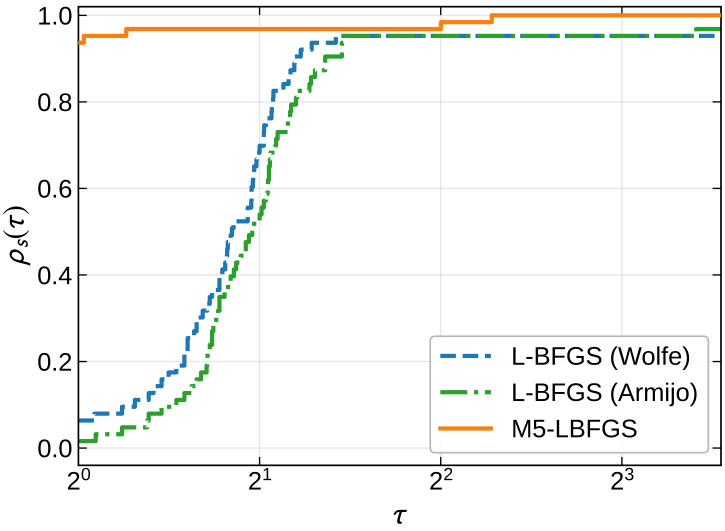


Fig. 4 Test set II: performance profiles for the number of iterations

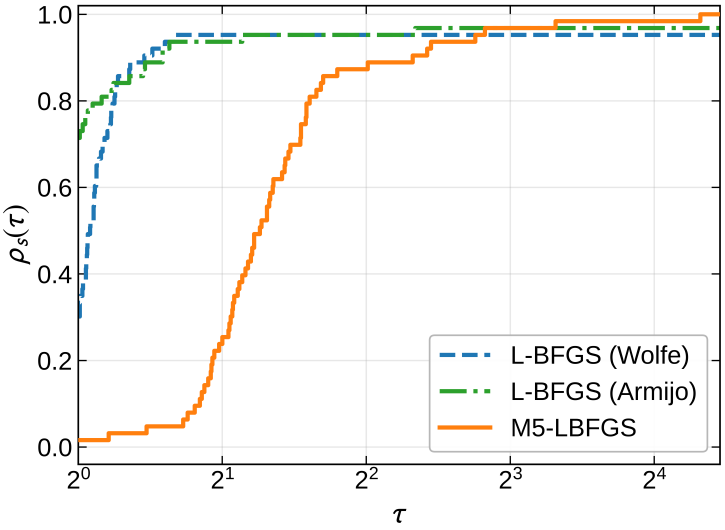


Fig. 5 Test set II: performance profiles for the number of gradient evaluations

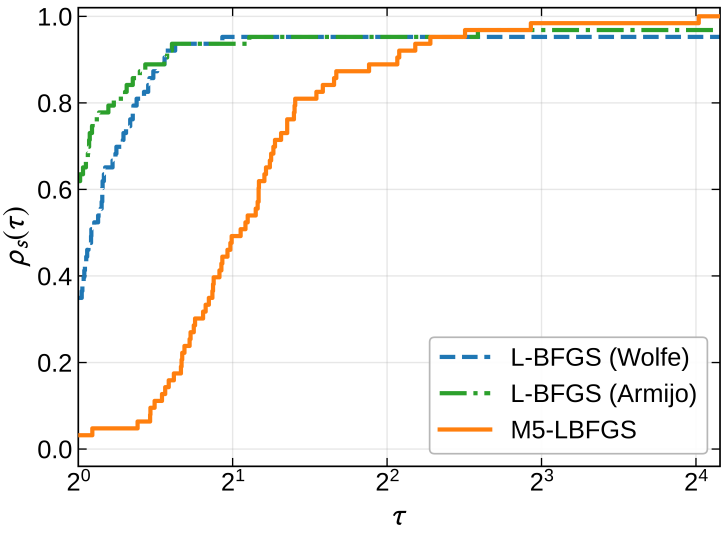


Fig. 6 Test set II: performance profiles for the CPU time

6.5 Effect of the memory size

To see the effect of the memory size, we repeated the experiment on test set I with $m = 3$, 10 and 20, in addition to the baseline value $m = 5$. Table 5, Figs. 7 and 8 show that, for each method, the number of problems solved varies by at most two across the four memory sizes. The geometric mean ratios (M5–LBFGS / L-BFGS (Wolfe)) remain nearly unchanged: between 0.54 and 0.56 for iterations and between 2.12 and 2.16 for gradient evaluations. The CPU-time ratios also vary little, ranging from 1.65 to 1.73. The median iteration counts show only modest variation with the memory size.

Table 5 Effect of the memory size m on test set I. W, A, M5: L-BFGS (Wolfe), L-BFGS (Armijo), M5–LBFGS. Median iterations and ratios (geometric means) are taken over the problems solved by all three methods for the given m.

m	Solved	Median iterations	M5–LBFGS / L-BFGS (Wolfe)		
	W / A / M5	W / A / M5	Iterations	Gradients	CPU
3	83 / 88 / 89	151 / 153 / 87	0.54	2.12	1.65
5	84 / 87 / 89	137.5 / 132 / 74.5	0.56	2.16	1.71
10	84 / 86 / 88	130 / 135 / 73	0.55	2.13	1.73
20	84 / 87 / 88	131.5 / 136 / 68	0.55	2.12	1.70

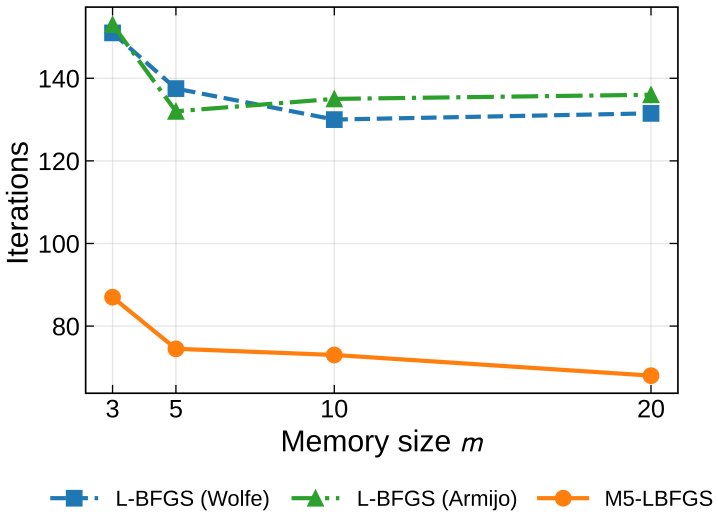


Fig. 7 Test set I: median number of iterations for the memory sizes $m = 3, 5, 10, 20$ (problems solved by all three methods)

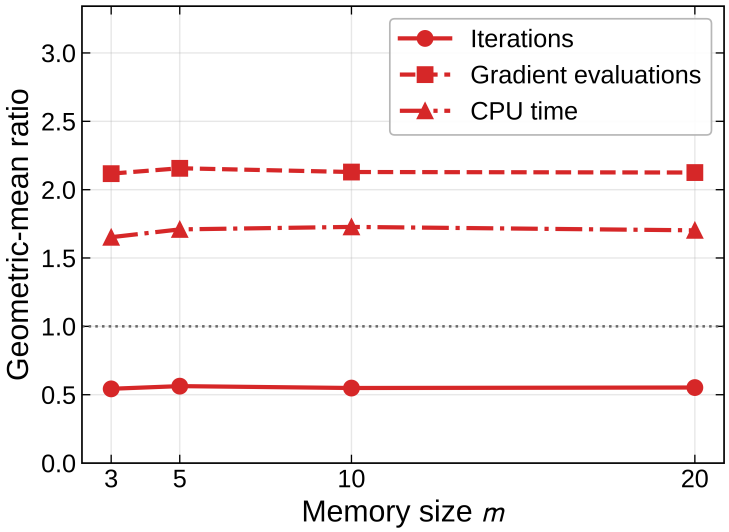


Fig. 8 Test set I: geometric means of the ratios M5–LBFGS / L-BFGS (Wolfe) of the iterations, gradient evaluations and CPU time for $m = 3, 5, 10, 20$; a value below the dotted line favours M5–LBFGS

7 Discussion

In Section 3, we establish the L-BFGS analogous guarantees for M5–LBFGS. Under the stated assumptions, the method showed global convergence and an R-linear convergence rate. The computational complexity of the proposed method is $O(mn)$. The numerical experiments show the importance of the M5 step in practice. M5–LBFGS requires about 40% fewer iterations than L-BFGS (Wolfe) on both test sets (the geometric means of the ratios are 0.56 and 0.62) and for every memory size we tried. It is more robust than L-BFGS, as it solves more number of problems. The limitation of our proposed method is that each iteration is more costly than the L-BFGS method; each iteration requires at least three gradient evaluations (Section 5) and takes approximately twice the CPU time on the CUTEst problems.

M5–LBFGS is a robust and iteration-efficient method. It achieves the required accuracy in far fewer iterations than L-BFGS, at the cost of more gradient evaluations per iteration. It is most useful when the number of iterations matters more than the number of gradient evaluations, and when robustness matters, as in problems on which L-BFGS fails or requires many iterations, such as CURLY20, MODBEALE and SINQUAD in our tests.

The cost of an iteration could be reduced by using the M5 step only in iterations in which the L-BFGS step makes little progress. Since the L-BFGS step with Armijo backtracking also satisfies a sufficient-decrease estimate of the form of Proposition 3.14, the analysis of Section 3 extends to such a combination. Other open questions are an analysis for nonconvex functions, for example along the lines of the regularized methods in [6, 25], and applications such as the training of physics-informed neural networks.

8 Conclusions

We proposed a robust M5–LBFGS method for nonlinear optimization. Under the stated assumptions, we prove that the backtracking line search terminates after a bounded number of trials and that the iterates converge to the global minimizer. Analogous to L-BFGS, we establish the R-linear convergence rate of the proposed method. Numerical experiments were performed on two sets of CUTEst problems of dimensions varying between 1000 and 10000 variables; M5–LBFGS solved more problems than L-BFGS, never failed on a problem solved by L-BFGS, and needed about 40% fewer iterations. L-BFGS was about twice as fast in CPU time because it requires fewer gradient evaluations. M5–LBFGS is best suited to problems in which robustness and the number of iterations matter more than the number of gradient evaluations.

Statements and Declarations

- **Funding:** The authors thank the Dr. B.R. Ambedkar National Institute of Technology, Jalandhar, for providing financial support through the student scholarship (to MD) and seed grant (to MK) to establish the computational resources.

- **Competing interests:** The authors have no competing interests to declare that are relevant to the content of this article.
- **Ethics approval and consent to participate:** Not applicable.
- **Consent for publication:** Not applicable.
- **Data availability:** All test problems are from the publicly available CUTEst collection [15]. The complete results of Section 6, including the final function values and gradient norms of all runs, are available in the repository given below.
- **Materials availability:** Not applicable.
- **Code availability:** The Python implementation of the three methods and a Jupyter notebook that runs all experiments of Section 6 on Google Colab and produces all tables and figures are available at https://github.com/Rittu2122/M5-LBFGS.
- **Use of AI tools:** A generative AI tool was used to provide limited assistance with the development and debugging of parts of the Python code for the numerical experiments.
- **Author contributions:** [Authors: describe the contribution of each author, for example conceptualization, methodology, software, analysis, writing.]

Appendix A Results for every problem

Tables A1 and A2 give the number of iterations, the number of gradient evaluations and the CPU time of every run with $m = 5$. W, A and M5 denote L-BFGS (Wolfe), L-BFGS (Armijo) and M5–LBFGS; a dash marks a failed run (see Tables 2 and 4 for the reasons).

Table A1: Test set I ($1000 \leq n \leq 2996$): results for every problem.

Problem	n	Iterations			Gradient evaluations			CPU time (s)		
		W	A	M5	W	A	M5	W	A	M5
ARWHEAD	1000	11	11	7	13	12	22	0.004	0.003	0.004
BDQRTIC	1000	–	200	104	–	201	505	–	0.068	0.135
BOX	1000	10	10	7	12	11	42	0.004	0.003	0.009
BOXPOWER	1000	32	53	25	47	54	136	0.006	0.007	0.014
BROWNAL	1000	3	3	2	5	4	7	0.020	0.013	0.018
BROYDN3DLS	1000	33	33	15	35	34	60	0.007	0.007	0.010
BROYDN7D	1000	355	357	208	363	358	1239	0.084	0.079	0.242
BROYDNBDLS	1000	31	31	17	33	32	68	0.007	0.006	0.011
BRYBND	1000	31	31	17	33	32	68	0.007	0.006	0.011
CHAINWOO	1000	3932	4407	148	4222	4408	731	0.824	1.018	0.113
COSINE	1000	7	19	9	17	20	34	0.002	0.003	0.006
CRAGGLVY	1000	73	85	39	79	86	186	0.018	0.017	0.031
CURLY10	1000	–	7855	4150	–	7856	18961	–	1.356	2.292
CURLY20	1000	–	–	4810	–	–	23411	–	–	3.316
CURLY30	1000	–	20484	5140	–	20485	27029	–	23.970	4.893
CYCLIC3LS	1002	1617	1351	581	1915	1352	3340	0.315	0.210	0.450
CYCLOOCFLS	2996	16538	25301	10367	17137	25302	49318	5.564	8.251	13.690
DIXMAANA1	1500	11	12	7	13	13	22	0.002	0.002	0.002
DIXMAANB	1500	11	11	7	13	12	22	0.002	0.002	0.003
DIXMAANC	1500	12	12	8	14	13	25	0.002	0.002	0.003
DIXMAAND	1500	15	14	9	17	15	28	0.003	0.002	0.004
DIXMAANE1	1500	210	218	102	219	219	443	0.056	0.048	0.085
DIXMAANF	1500	177	180	78	184	181	341	0.034	0.032	0.051
DIXMAANG	1500	181	182	91	189	183	404	0.036	0.033	0.063
DIXMAANH	1500	167	180	84	173	181	349	0.032	0.031	0.052
DIXMAANI1	1500	1965	2249	1062	2035	2250	4797	0.338	0.518	0.839
DIXMAANJ	1500	932	1015	447	960	1016	2004	0.183	0.174	0.293
DIXMAANK	1500	1074	1021	425	1107	1022	1948	0.208	0.170	0.279
DIXMAANL	1500	950	941	435	980	942	1914	0.287	0.162	0.285
DIXMAANM1	1500	1891	1879	1219	1951	1880	5622	0.314	0.274	0.725
DIXMAANN	1500	719	1267	318	741	1268	1443	0.143	0.264	0.342
DIXMAANO	1500	878	1714	273	905	1715	1202	0.166	0.296	0.183
DIXMAANP	1500	706	977	251	729	978	1120	0.138	0.170	0.168
DIXON3DQ	1000	3180	3372	1350	3300	3373	6109	0.447	0.614	0.632
DQDRTIC	1000	13	13	12	19	14	47	0.003	0.002	0.005
DQRTIC	1000	32	32	20	38	33	61	0.004	0.003	0.005
EDENSCH	2000	27	28	17	32	29	62	0.009	0.009	0.015
EIGENALS	2550	6270	9735	5123	6497	9736	23330	26.420	40.540	90.490
EIGENBLS	2550	6213	6835	3365	6360	6836	15776	34.850	41.240	86.110
EIGENCLS	2652	7701	10203	6291	7954	10204	28750	52.230	59.670	156.900
ENGVAL1	1000	17	17	11	19	18	36	0.003	0.003	0.004
EXTROSNB	1000	3728	3135	1405	4923	3136	10774	0.711	0.456	1.318
FLETBV3M	1000	21	10	20	35	11	95	0.009	0.003	0.020
FLETCHBV	1000	1050	1325	294	1091	1326	1331	0.424	0.503	0.418
FLETCHCR	1000	5020	5077	3333	5678	5078	23408	0.967	1.029	3.241
FMINSRF2	1024	136	131	65	143	132	316	0.025	0.021	0.042
FMINSURF	1024	225	226	114	235	227	527	0.043	0.039	0.079
FREUROTH	1000	28	33	15	39	34	88	0.007	0.008	0.013
GENHUMPS	1000	2763	2146	1328	3118	2147	7983	0.774	0.771	1.596
INDEFM	1000	146	176	99	193	177	582	0.045	0.043	0.107
KSSLS	1000	12	12	7	19	13	22	0.120	0.089	0.133
LIARWHD	1000	20	23	16	25	24	57	0.006	0.006	0.011
MODBEALE	2000	–	–	218	–	–	1023	–	–	0.398
MSQRTALS	1024	2378	1945	1307	2446	1946	5778	1.992	1.314	3.537

continued on the next page

Table A1 (continued)

Problem	n	Iterations			Gradient evaluations			CPU time (s)		
		W	A	M5	W	A	M5	W	A	M5
MSQRTBLS	1024	1660	1604	808	1714	1605	3639	1.194	1.304	2.208
NCB20	1010	670	546	249	747	547	1212	0.405	0.194	0.367
NCB20B	1000	2186	1649	796	2358	1650	3543	0.869	0.586	1.400
NONCVXU2	1000	1328	1571	675	1380	1572	2940	0.261	0.281	0.460
NONCVXUN	1000	1951	4509	981	2016	4510	4504	0.403	1.052	0.710
NONDIA	1000	17	18	8	20	19	25	0.004	0.003	0.003
NONDQUAR	1000	371	295	165	410	296	770	0.058	0.036	0.080
NONMSQRT	1024	–	–	–	–	–	–	–	–	–
OSCIGRAD	1000	66	66	35	72	67	144	0.014	0.010	0.019
PENALTY1	1000	64	67	42	79	68	153	0.010	0.008	0.016
PENALTY2	1000	–	–	–	–	–	–	–	–	–
POWELLSG	1000	54	73	30	61	74	123	0.014	0.014	0.019
POWER	1000	139	133	69	144	134	324	0.018	0.014	0.029
QING	1000	163	238	110	179	239	471	0.026	0.027	0.049
QUARTC	1000	32	32	20	38	33	61	0.005	0.004	0.006
SBRYBND	1000	–	–	–	–	–	–	–	–	–
SCHMVETT	1000	39	35	26	45	36	121	0.011	0.008	0.023
SCOSINE	1000	–	–	–	–	–	–	–	–	–
SCURLY10	1000	–	–	–	–	–	–	–	–	–
SCURLY20	1000	–	–	–	–	–	–	–	–	–
SCURLY30	1000	–	–	–	–	–	–	–	–	–
SENSORS	1000	27	32	23	35	33	108	7.033	7.423	16.290
SINQUAD	1000	26	23	18	38	24	91	0.007	0.004	0.012
SINQUAD2	1000	109	116	113	149	117	708	0.029	0.022	0.119
SPARSINE	1000	6001	6948	3150	6297	6949	14225	1.706	1.406	2.742
SPARSQUR	1000	27	27	16	28	28	49	0.005	0.005	0.007
SPINLS	1082	–	–	–	–	–	–	–	–	–
SPMSRTLS	1000	105	112	54	116	113	227	0.023	0.021	0.035
SROSENBR	1000	52	79	35	63	80	184	0.012	0.015	0.027
SSBRYBND	1000	14687	8996	4001	15484	8997	18804	3.417	1.832	3.340
SSCOSINE	1000	–	–	–	–	–	–	–	–	–
STRTCHDV	1000	19	19	10	20	20	31	0.007	0.006	0.009
TESTQUAD	1000	4836	4062	4426	4993	4063	19997	0.637	0.460	1.959
TOINTGSS	1000	14	26	13	20	27	70	0.003	0.004	0.009
TQUARTIC	1000	21	22	11	27	23	70	0.004	0.003	0.007
TRIDIA	1000	575	748	387	599	749	1740	0.080	0.097	0.175
TRIGON1	1000	11253	11582	5504	11600	11583	25041	120.800	115.300	238.800
TRIGON2	1000	435	582	279	471	583	1268	0.135	0.157	0.422
VAREIGVL	1000	170	171	80	174	172	373	0.038	0.035	0.066
WOODS	1000	98	92	71	128	93	432	0.019	0.013	0.051
YATP1CLS	2600	21	27	21	31	28	114	0.024	0.023	0.076
YATP1LS	2600	21	27	21	31	28	114	0.023	0.023	0.077
YATP2CLS	2600	11	16	11	20	17	68	0.018	0.015	0.047
YATP2LS	2600	11	16	11	20	17	68	0.012	0.011	0.032

Table A2: Test set II ($5000 \leq n \leq 10000$): results for every problem.

Problem	n	Iterations			Gradient evaluations			CPU time (s)		
		W	A	M5	W	A	M5	W	A	M5
ARWHEAD	5000	12	11	8	14	12	25	0.009	0.007	0.010
BDQRTIC	5000	–	342	214	–	343	2427	–	0.363	1.532
BOX	10000	10	10	12	12	11	59	0.014	0.013	0.058

continued on the next page

Table A2 (continued)

Problem	n	Iterations			Gradient evaluations			CPU time (s)		
		W	A	M5	W	A	M5	W	A	M5
BOXPOWER	10000	45	52	25	55	53	128	0.030	0.032	0.058
BROYDN3DLS	10000	30	34	17	35	35	76	0.027	0.026	0.048
BROYDN7D	10000	3117	3123	1723	3149	3124	10150	5.515	5.297	15.430
BROYDNBDLS	10000	41	74	27	50	75	150	0.068	0.102	0.251
BRYBND	10000	41	74	27	50	75	150	0.075	0.101	0.180
CHAINWOO	10000	423	307	158	467	308	763	0.595	0.386	0.866
COSINE	10000	8	85	32	17	86	169	0.018	0.109	0.139
CRAGGLVY	5000	78	85	37	88	86	164	0.064	0.066	0.106
DIXMAANA1	9000	12	12	7	14	13	22	0.008	0.006	0.008
DIXMAANB	9000	11	11	7	13	12	22	0.013	0.012	0.017
DIXMAANC	9000	12	12	8	14	13	25	0.010	0.008	0.012
DIXMAAND	9000	14	15	9	16	16	28	0.010	0.010	0.014
DIXMAANE1	9000	411	426	208	430	427	919	0.239	0.234	0.428
DIXMAANF	9000	376	302	185	389	303	820	0.425	0.223	0.495
DIXMAANG	9000	335	285	171	344	286	794	0.245	0.194	0.469
DIXMAANH	9000	291	351	142	300	352	699	0.312	0.388	0.440
DIXMAANI1	9000	1599	1953	807	1649	1954	3630	0.933	1.094	1.976
DIXMAANJ	9000	323	309	169	334	310	780	0.238	0.249	0.474
DIXMAANK	9000	279	282	144	285	283	633	0.294	0.200	0.377
DIXMAANL	9000	268	292	120	280	293	579	0.193	0.205	0.352
DIXMAANM1	9000	1808	1914	859	1875	1915	3932	1.302	1.067	1.867
DIXMAANN	9000	927	938	406	957	939	1783	0.671	0.634	1.065
DIXMAANO	9000	797	794	381	826	795	1674	0.900	0.609	1.029
DIXMAANP	9000	720	632	295	740	633	1304	0.532	0.474	1.127
DIXON3DQ	10000	2236	2034	1147	2309	2035	5190	1.268	1.073	2.413
DQDRTIC	5000	13	13	11	20	14	44	0.008	0.007	0.017
DQRTIC	5000	36	36	22	44	37	67	0.023	0.017	0.023
ENGVAL1	5000	16	16	11	18	17	36	0.009	0.008	0.015
FLETBV3M	10000	32	37	24	42	38	111	0.082	0.082	0.182
FMINSRF2	10000	322	315	166	330	316	725	0.346	0.331	0.991
FMINSURF	10000	650	642	367	667	643	1644	0.725	0.650	1.711
FREUROTH	5000	–	33	16	–	34	99	–	0.023	0.052
GENHUMPS	5000	8208	8978	4731	10719	8979	28862	10.450	8.960	22.260
INDEFM	10000	381	491	237	513	492	1312	1.104	0.992	1.573
LIARWHD	10000	24	29	14	28	30	53	0.032	0.035	0.052
NCB20	5010	400	326	191	449	327	978	0.701	0.501	1.278
NCB20B	5000	1525	1570	696	1666	1571	3141	2.512	2.644	4.043
NONCVXU2	10000	2197	2077	1081	2269	2078	4800	3.060	2.580	5.520
NONCVXUN	10000	1952	2242	1006	2004	2243	4535	2.970	3.032	5.335
NONDIA	10000	18	17	8	23	18	25	0.019	0.015	0.016
NONDQUAR	10000	149	181	85	176	182	436	0.152	0.160	0.342
OSCIGRAD	10000	62	63	31	67	64	126	0.063	0.060	0.106
POWELLSG	10000	60	59	36	69	60	165	0.038	0.031	0.071
POWER	10000	409	405	201	423	406	1018	0.192	0.237	0.605
QING	10000	462	755	336	488	756	1425	0.340	0.449	0.804
QUARTC	10000	38	38	23	46	39	70	0.021	0.020	0.028
SCHMVETT	10000	34	34	26	41	35	141	0.072	0.064	0.205
SCURLY10	10000	–	–	–	–	–	–	–	–	–
SCURLY20	10000	–	–	–	–	–	–	–	–	–
SCURLY30	10000	–	–	–	–	–	–	–	–	–
SINQUAD	10000	–	–	18	–	–	79	–	–	0.075
SINQUAD2	10000	253	218	204	349	219	1481	0.430	0.302	1.714
SPARSINE	10000	46603	–	19909	48500	–	90566	84.80	–	134.80
SPARSQUR	10000	34	34	19	39	35	58	0.046	0.041	0.063
SPMSRTLS	10000	149	161	78	161	162	375	0.193	0.206	0.400

continued on the next page

Table A2 (continued)

Problem	n	Iterations			Gradient evaluations			CPU time (s)		
		W	A	M5	W	A	M5	W	A	M5
SROSENBR	10000	49	83	50	61	84	333	0.037	0.054	0.178
SSBRYBND	5000	9257	13081	4742	9577	13082	22959	7.659	9.780	14.960
TESTQUAD	5000	4567	4329	2585	4717	4330	11690	1.294	1.274	2.520
TOINTGSS	10000	16	18	7	23	19	22	0.031	0.027	0.024
TQUARTIC	10000	22	24	14	27	25	87	0.022	0.022	0.058
TRIDIA	10000	2126	2800	1505	2193	2801	6682	1.182	1.798	3.016
VAREIGVL	5000	21	54	102	25	55	499	0.019	0.041	0.308
WOODS	10000	85	104	80	118	105	525	0.085	0.077	0.324

References

[1] Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, New York (2006). https://doi.org/10.1007/978-0-387-40065-5

[2] Powell, M.J.D.: Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In: Nonlinear Programming, SIAM-AMS Proceedings, Vol. IX, pp. 53–72 (1976)

[3] Nocedal, J.: Updating quasi-Newton matrices with limited storage. Mathematics of Computation **35**(151), 773–782 (1980) https://doi.org/10.1090/S0025-5718-1980-0572855-7

[4] Byrd, R.H., Nocedal, J., Schnabel, R.B.: Representations of quasi-Newton matrices and their use in limited memory methods. Mathematical Programming **63**, 129–156 (1994) https://doi.org/10.1007/BF01582063

[5] Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming **45**, 503–528 (1989) https://doi.org/10.1007/BF01589116

[6] Tankaria, H., Sugimoto, S., Yamashita, N.: A regularized limited memory BFGS method for large-scale unconstrained optimization and its efficient implementations. Computational Optimization and Applications **82**(1), 61–88 (2022) https://doi.org/10.1007/s10589-022-00351-5

[7] Traub, J.F.: Iterative Methods for the Solution of Equations. Prentice-Hall, Englewood Cliffs, NJ (1964)

[8] Hueso, J.L., Martínez, E., Torregrosa, J.R.: Third and fourth order iterative methods free from second derivative for nonlinear systems. Applied Mathematics and Computation **211**, 190–197 (2009) https://doi.org/10.1016/j.amc.2009.01.039

[9] Arroyo, V., Cordero, A., Torregrosa, J.R.: Approximation of artificial satellites' preliminary orbits: The efficiency challenge. Mathematical and Computer Modelling **54**, 1802–1807 (2011) https://doi.org/10.1016/j.mcm.2010.11.063

[10] Ford, J.A., Moghrabi, I.A.: Multi-step quasi-Newton methods for optimization. Journal of Computational and Applied Mathematics **50**, 305–323 (1994) https://doi.org/10.1016/0377-0427(94)90309-3

[11] Chakraborty, S.K., Panda, G.: Two-phase quasi-newton method for unconstrained optimization problem. Afrika Matematika **30**, 737–746 (2019) https://doi.org/10.1007/s13370-019-00680-5

[12] Moghrabi, I.A.R., Hassan, B.A.: An efficient limited memory multi-step quasi-newton method. Mathematics **12**(5), 768 (2024) https://doi.org/10.3390/math12050768

[13] Cordero, A., Maimó, J.G., Torregrosa, J.R., Ureña Castillo, N.: Efficiency and stability of a new hybrid unconstrained optimization algorithm with quasi-newton updates and higher-order methods. Mathematics **14**(10), 1746 (2026) https://doi.org/10.3390/math14101746

[14] Armijo, L.: Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of Mathematics **16**(1), 1–3 (1966) https://doi.org/10.2140/pjm.1966.16.1

[15] Gould, N.I.M., Orban, D., Toint, P.L.: CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. Computational Optimization and Applications **60**(3), 545–557 (2015) https://doi.org/10.1007/s10589-014-9687-3

[16] Polyak, B.T.: Gradient methods for minimizing functionals. USSR Computational Mathematics and Mathematical Physics **3**(4), 864–878 (1963) https://doi.org/10.1016/0041-5553(63)90382-3

[17] Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1970). https://doi.org/10.1016/C2013-0-11263-9

[18] Fowkes, J., Roberts, L., Bűrmen, A.: PyCUTEst: an open source Python package of optimization test problems. Journal of Open Source Software **7**(78), 4377 (2022) https://doi.org/10.21105/joss.04377

[19] Lukšan, L., Matonoha, C., Vlček, J.: Modified CUTE problems for sparse unconstrained optimization. Technical Report V-1081, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic (October 2010). https://www.cs.cas.cz/matonoha/download/V1081.pdf

[20] Moré, J.J., Thuente, D.J.: Line search algorithms with guaranteed sufficient decrease. ACM Transactions on Mathematical Software **20**(3), 286–307 (1994) https://doi.org/10.1145/192115.192132

[21] Virtanen, P., *et al.*: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods **17**, 261–272 (2020) https://doi.org/10.1038/s41592-019-0686-2

[22] Mannel, F., Aggrawal, H.O., Modersitzki, J.: A structured L-BFGS method and its application to inverse problems. Inverse Problems **40**(4), 045022 (2024) https://doi.org/10.1088/1361-6420/ad2c31

[23] Mannel, F., Aggrawal, H.O.: A structured L-BFGS method with diagonal scaling and its application to image registration. Journal of Mathematical Imaging and Vision **67**, 7 (2025) https://doi.org/10.1007/s10851-024-01215-9

[24] Brust, J., Burdakov, O., Erway, J.B., Marcia, R.F.: A dense initialization for limited-memory quasi-Newton methods. Computational Optimization and Applications **74**(1), 121–142 (2019) https://doi.org/10.1007/s10589-019-00112-x

[25] Kanzow, C., Steck, D.: Regularization of limited memory quasi-Newton methods for large-scale nonconvex minimization. Mathematical Programming Computation **15**, 417–444 (2023) https://doi.org/10.1007/s12532-023-00238-4

[26] Harris, C.R., *et al.*: Array programming with NumPy. Nature **585**, 357–362 (2020) https://doi.org/10.1038/s41586-020-2649-2

[27] Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002) https://doi.org/10.1007/s101070100263