# Regularization of Limited Memory Quasi-Newton Methods for Large-Scale Nonconvex Minimization

Christian Kanzow* Daniel Steck†

October 21, 2019

**Abstract.** This paper deals with the unconstrained optimization of smooth objective functions. It presents a class of regularized quasi-Newton methods whose globalization turns out to be more efficient than standard line search or trust-region strategies. The focus is therefore on the solution of large-scale problems using limited memory quasi-Newton techniques. Global convergence of the regularization methods is shown under mild assumptions. The details of the regularized limited memory quasi-Newton updates are discussed including their compact representations. Numerical results using all large-scale test problems from the CUTEst collection indicate that the regularization method outperforms the standard line search limited memory BFGS method.

**Keywords.** Limited memory methods, quasi-Newton methods, L-BFGS, regularized Newton methods, global convergence, large-scale optimization.

**AMS subject classifications.** 49M, 65K, 90C.

## 1 Introduction

Let $f : \mathbb{R}^n \to \mathbb{R}$, $n \in \mathbb{N}$, be a twice continuously differentiable function, and consider the nonlinear minimization problem

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}}\ f(\mathbf{x}). \tag{1}$$

Methods of Newton or quasi-Newton type are commonly acknowledged to be some of the most efficient algorithms for the solution of such problems. Given a current iterate $\mathbf{x}_k$, these methods compute the iteration step $\mathbf{d}_k$ by solving a *(quasi-)Newton equation* of the form

$$\mathbf{B}_k\mathbf{d}_k = -\nabla f(\mathbf{x}_k), \tag{2}$$

where $\mathbf{B}_k \in \mathbb{R}^{n\times n}$ is either the Hessian $\nabla^2 f(\mathbf{x}_k)$ or an approximation thereof. When $n$ is large, the matrix $\mathbf{B}_k$ is usually not stored explicitly. Instead, one uses so-called *limited memory* quasi-Newton methods, which require the storage of a few vector pairs

$$\mathbf{s}_k := \mathbf{x}_{k+1} - \mathbf{x}_k, \qquad \mathbf{y}_k := \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k),$$

and use this information to construct an implicit approximation to the Hessian matrix. This approximation is never formed explicitly; instead, the pairs $(\mathbf{s}_k, \mathbf{y}_k)$ are used to directly

*University of Würzburg, Institute of Mathematics, Campus Hubland Nord, Emil-Fischer-Str. 30, 97074 Würzburg, Germany; kanzow@mathematik.uni-wuerzburg.de

†Independent Researcher, London E2, United Kingdom; mail@danielsteck.net

evaluate matrix–vector products of the form $\mathbf{B}_k\mathbf{x}$ or $\mathbf{B}_k^{-1}\mathbf{y}$ as necessary. Arguably the most successful quasi-Newton schemes are the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [8] and its limited memory counterpart L-BFGS [5, 17, 19]. Other examples include symmetric rank-one (SR1), Powell-symmetric-Broyden (PSB), Davidon–Fletcher–Powell (DFP), the so called Broyden class, and many more; see [8, 16, 25].

In today's optimization landscape, L-BFGS is the de facto standard for smooth large-scale optimization. The method is usually combined with a line search technique to ensure global convergence [17]. There have also been efforts dedicated to making quasi-Newton methods compatible with the trust-region framework; see [2, 4, 10] for L-BFGS and [1] for L-SR1. This is facilitated by the fact that most quasi-Newton schemes admit a so-called *compact representation* of the form

$$\mathbf{B}_k = \mathbf{B}_{0,k} + \mathbf{A}_k\mathbf{Q}_k^{-1}\mathbf{A}_k^\mathsf{T}, \tag{3}$$

where $\mathbf{B}_{0,k} \in \mathbb{R}^{n\times n}$, $\mathbf{A}_k \in \mathbb{R}^{n\times s}$, $\mathbf{Q}_k \in \mathbb{R}^{s\times s}$ and $s \ll n$. (We put $\mathbf{Q}_k^{-1}$ instead of $\mathbf{Q}_k$ in the above equation because this will be more convenient later on.) The initial matrix $\mathbf{B}_{0,k}$ is usually a multiple of the identity or some other diagonal matrix. Decompositions of the above form have been given by many authors [3, 5, 7], and they are immensely useful in optimization methods since they usually allow the computation of matrix operations involving $\mathbf{B}_k$ in the lower dimension $s$. In particular, they facilitate the efficient computation of quasi-Newton directions and the solution of trust-region subproblems; see the references above.

In this paper, we will pursue a different globalization technique which can be seen as a (less well-known) sibling of line search and trust-region methods, the so-called *regularized Newton methods* [13, 15, 23, 24, 26]. These are generally characterized by regularized quasi-Newton equations of the form

$$(\mathbf{B}_k + \mu_k\mathbf{I})\mathbf{d}_k = -\nabla f(\mathbf{x}_k), \tag{4}$$

where $\mu_k \geq 0$ is called the regularization parameter. The attractive feature of these methods is that they combine some of the respective benefits of line search and trust region methods, and moreover they are highly compatible with compact representations of quasi-Newton matrices. We will therefore present an algorithmic framework designed to efficiently combine limited memory and regularization techniques, with the following benefits:

- The step computation is almost as cheap as for line search L-BFGS algorithms. More specifically, the cost of each successful iteration (in the BFGS case) is $4mn$ plus the solution of a $2m \times 2m$ symmetric linear system. In particular, no inner loop is necessary for the computation of eigenvalue decompositions or trust-region solutions.

- At the same time, the step quality is close to that of trust-region type limited memory algorithms because the regularization parameter (4) mimics the Lagrange multiplier arising in trust-region subproblems. The method can therefore be considered as a kind of "implicit" trust-region algorithm.

- As a result of the above, the proportion of accepted steps is extremely high, leading to a very low number of function and gradient evaluations (on a level with trust-region type methods) while at the same time preserving the "cheap" steps of line search methods.

The use of regularization techniques has another important benefit over line search methods. In the line search setting, many authors advocate trying the "full" step size $t_k = 1$ first, the motivation being that L-BFGS and similar methods are fundamentally algorithms of Newton type and the full step size may lead to fast convergence. However, the step size

also serves the purpose of adapting the algorithm to the nonlinearity of the problem, and re-initializing the line search procedure with $t_k = 1$ at each step makes it hard to carry this information over from one step to the next. In contrast, the regularization approach that we advocate here provides a more seamless transition between the full (quasi-)Newton step and a truncated version thereof (similar to trust region methods), which suggests that algorithms of this type may be able to handle nonlinear or nonconvex problems more effectively.

The idea of combining limited memory and regularization techniques is not entirely new. Multiple authors [13, 21, 22] have advocated modifying the secant equation in quasi-Newton methods to instead approximate the sum $\nabla^2 f(\mathbf{x}_k) + \mu_k \mathbf{I}$. However, none of these methods fully exploit the quasi-Newton approximation of the Hessian and the compact representation (3). The method we present takes full advantage of these tools and appears to perform better than comparable methods from the literature (see Section 5).

In addition to the quasi-Newton approach, the present paper also contains a general convergence result for regularized Newton methods which, to the authors' knowledge, does not exist in this generality in the literature. This may be of interest to researchers in the field and provide a basis for future research on related methods.

This paper is organized as follows. Section 2 contains a detailed description of a general class of regularized quasi-Newton methods. Global convergence results for this class of methods are presented in Section 3 under fairly mild assumptions. The realization of this method using several limited memory quasi-Newton matrices are discussed in Section 4 and based on suitable compact representation of these matrices; in particular, the compact representation of the PSB-formula given there is new. The numerical experiments in Section 5 indicate that the new technique is superior to the conventional limited memory BFGS method. We close with some final remarks in Section 6.

**Notation**

Matrices and vectors will be denoted by boldface letters $\mathbf{M}$ and $\mathbf{v}$, respectively. Given a matrix $\mathbf{M} \in \mathbb{R}^{s\times s}$, we write $\mathbf{L}(\mathbf{M})$, $\mathbf{D}(\mathbf{M})$, and $\mathbf{U}(\mathbf{M})$ for the *strictly lower, diagonal,* and *strictly upper* parts of $\mathbf{M}$, respectively. In particular, it always holds that

$$\mathbf{M} = \mathbf{L}(\mathbf{M}) + \mathbf{D}(\mathbf{M}) + \mathbf{U}(\mathbf{M}).$$

The gradient of the smooth function $f$ evaluated at an iterate $\mathbf{x}_k$ will often be denoted by $\mathbf{g}_k$.

## 2 Regularized Quasi-Newton Methods

As discussed in the introduction, the fundamental principle underlying the methods in this paper is that of regularized Newton and quasi-Newton methods, which are generally characterized by regularized quasi-Newton equations of the form

$$(\mathbf{B}_k + \mu_k \mathbf{I})\mathbf{d}_k = -\nabla f(\mathbf{x}_k), \tag{5}$$

where $\mathbf{B}_k$ is either the Hessian $\nabla^2 f(\mathbf{x}_k)$ or an approximation thereof, and $\mu_k \geq 0$ is the *regularization parameter*. Clearly, if $\mu_k = 0$, then (5) reduces to the standard quasi-Newton equation $\mathbf{B}_k \mathbf{d}_k = -\nabla f(\mathbf{x}_k)$. On the other hand, if $\mu_k \gg 0$ is large, then the matrix $\mathbf{B}_k + \mu_k \mathbf{I}$ will be invertible, and the step $\mathbf{d}_k$ produced by (5) will essentially be the negative gradient direction (up to normalization; see Lemma 3.2).

## 2.1 Mathematical Motivation

The virtues of the regularization approach can be understood by recognizing that this essentially amounts to minimizing the regularized quadratic model

$$\hat{q}_k(\mathbf{d}) := f(\mathbf{x}_k) + \mathbf{g}_k^\mathsf{T}\mathbf{d} + \frac{1}{2}\mathbf{d}^\mathsf{T}\mathbf{B}_k\mathbf{d} + \frac{\mu_k}{2}\|\mathbf{d}\|^2, \tag{6}$$

which differs from the conventional Newton model by Tikhonov regularization. Thus, a positive value of $\mu_k$ may dampen the impact of negative eigenvalues of $\mathbf{B}_k$ on the search direction, prevent excessively long steps in negative curvature directions, and possibly guarantee that the model (6) admits a unique minimizer (i.e., that the matrix $\mathbf{B}_k + \mu_k\mathbf{I}$ is positive definite). The anticipated setting is that $\mu_k$ will initially be kept sufficiently large to guarantee global convergence, eventually decreasing rapidly enough so as to not impede fast local convergence.

A more rigorous interpretation is given by trust-region methods. Indeed, if $\mathbf{d}_k := -(\mathbf{B}_k + \mu_k\mathbf{I})^{-1}\mathbf{g}_k$ for some $\mu_k \geq 0$, and if $\Delta := \|\mathbf{d}_k\|$, then $\mathbf{d}_k$ is a stationary point of the *trust-region subproblem*

$$\underset{\|\mathbf{d}\|\leq\Delta}{\text{minimize}}\ q(\mathbf{d}) := \frac{1}{2}\mathbf{d}^\mathsf{T}\mathbf{B}_k\mathbf{d} + \mathbf{g}_k^\mathsf{T}\mathbf{d}.$$

If $\mathbf{B}_k + \mu_k\mathbf{I}$ is positive definite, then $\mathbf{d}_k$ is in fact a *solution* of this auxiliary problem. It follows that regularized Newton methods can be interpreted as "implicit" trust-region methods whereby the regularization parameter is controlled instead of the trust-region radius.

Finally, it is also interesting to analyze how the regularization technique affects the conditioning of the quadratic model (6). Assuming for the moment that $\mathbf{B}_k$ is positive definite (as it is, e.g., in BFGS-type methods), the regularization parameter improves the condition number of the underlying matrix in the sense that

$$\kappa(\mathbf{B}_k + \mu_k\mathbf{I}) = \frac{\lambda_{\max}(\mathbf{B}_k) + \mu_k}{\lambda_{\min}(\mathbf{B}_k) + \mu_k} \leq \frac{\lambda_{\max}(\mathbf{B}_k)}{\lambda_{\min}(\mathbf{B}_k)} = \kappa(\mathbf{B}_k),$$

where $\lambda_{\max}(\mathbf{B}_k), \lambda_{\min}(\mathbf{B}_k) > 0$ are the largest and smallest eigenvalues of $\mathbf{B}_k$, respectively.

## 2.2 Basic Algorithm

To control the regularization parameter $\mu_k$, let

$$q_k(\mathbf{d}) := f(\mathbf{x}_k) + \mathbf{g}_k^\mathsf{T}\mathbf{d} + \frac{1}{2}\mathbf{d}^\mathsf{T}\mathbf{B}_k\mathbf{d} \tag{7}$$

be the standard quadratic approximation of $f$ around $\mathbf{x}_k$. Note that this differs from (6) only by the absence of the Tikhonov regularization term. Given a candidate step $\mathbf{d}_k = -(\mathbf{B}_k + \mu_k\mathbf{I})^{-1}\mathbf{g}_k$, we can then define the *predicted reduction* of $f$ as

$$\text{pred}_k := f(\mathbf{x}_k) - q_k(\mathbf{d}_k) = -\mathbf{g}_k^\mathsf{T}\mathbf{d}_k - \frac{1}{2}\mathbf{d}_k^\mathsf{T}\mathbf{B}_k\mathbf{d}_k = \frac{\mu_k}{2}\|\mathbf{d}_k\|^2 - \frac{1}{2}\mathbf{g}_k^\mathsf{T}\mathbf{d}_k, \tag{8}$$

where the last equality uses the definition of $\mathbf{d}_k$. (Note that, in particular, the matrix $\mathbf{B}_k$ need not be available for the computation of $\text{pred}_k$.) This quantity will be compared to the *actual* or *achieved reduction* in step $k$,

$$\text{ared}_k := f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{d}_k). \tag{9}$$

Similar to trust-region methods [6], we use the ratio between these quantities to control the regularization parameter. To this end, we distinguish between three cases, unsuccessful **(u)**, successful **(s)**, and highly successful **(h)** steps. Special care also needs to be taken because there is no a-priori guarantee that $\text{pred}_k$ is positive (since $\mathbf{B}_k$ may be indefinite); such steps are treated in the same manner as unsuccessful ones.

**Algorithm 2.1 (Regularized quasi-Newton method).**
Choose $\mathbf{x}_0 \in \mathbb{R}^n$ and parameters $\mu_0 > 0$; $p_{\min}, c_1 \in (0,1)$; $c_2 \in (c_1, 1)$; $\sigma_1 \in (0,1)$; $\sigma_2 > 1$.

**Step 1.** If a suitable stopping criterion is satisfied, terminate.

**Step 2. (Step computation)** Choose $\mathbf{B}_k \in \mathbb{R}^{n\times n}$ and attempt to solve the regularized quasi-Newton equation

$$(\mathbf{B}_k + \mu_k \mathbf{I})\mathbf{d}_k = -\nabla f(\mathbf{x}_k). \tag{10}$$

If this equation admits no solution $\mathbf{d}_k$, or if $\text{pred}_k \leq p_{\min}\|\mathbf{g}_k\|\|\mathbf{d}_k\|$, set $\mathbf{x}_{k+1} := \mathbf{x}_k$, $\mu_{k+1} := \sigma_2\mu_k$, and go to *Step 4*. Otherwise, go to *Step 3*.

**Step 3. (Variable update)** Set $\varrho_k := \text{ared}_k/\text{pred}_k$ and perform one of the following steps:
**Step 3u** ($\varrho_k \leq c_1$)**.** Set $\mathbf{x}_{k+1} := \mathbf{x}_k$ and $\mu_{k+1} := \sigma_2\mu_k$.
**Step 3s** ($c_1 < \varrho_k \leq c_2$)**.** Set $\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{d}_k$ and $\mu_{k+1} := \mu_k$.
**Step 3h** ($c_2 < \varrho_k$)**.** Set $\mathbf{x}_{k+1} := \mathbf{x}_k + \mathbf{d}_k$ and $\mu_{k+1} := \sigma_1\mu_k$.

**Step 4.** Set $k \leftarrow k+1$ and go to Step 1.

Algorithm 2.1 is closely related to trust-region methods. The main difference between trust-region methods and our regularization framework lies in the update of the parameter $\mu_k$. The former uses an indirect way to compute $\mu_k$ (via a trust-region radius), whereas here we update the regularization parameter directly. While the indirect update follows a well-understood and well-motivated philosophy, its actual computation is sometimes time-consuming and therefore less efficient. We therefore expect a superior behavior of the direct update, in particular, for large-scale problems.

The report [22] presents a method which is almost (except for a slightly different update of the regularization parameter) identical to our Algorithm 2.1, but concentrates on the matrices $\mathbf{B}_k$ being updated by a limited memory BFGS scheme. The convergence theory in [22] assumes a bounded level set condition that is not required in our subsequent analysis which is significantly more general and only assumes the sequence $\{\mathbf{B}_k\}$ to be bounded.

# 3 General Convergence Analysis

As we shall see, Algorithm 2.1 provides a powerful framework for the application of quasi-Newton type updates. Before turning to this discussion (which is the main motivation for this paper), we shall dedicate the present section to a simple convergence analysis. Due to the non-specificity of the algorithm in its general form, it will be convenient to carry out the convergence analysis under rather general assumptions. To this end, we shall make no assumption on the particular choice of the matrices $\mathbf{B}_k$, which may or may not be approximations of the Hessian $\nabla^2 f(\mathbf{x}^k)$. The only assumption we make throughout this section is the following.

**Assumption 3.1 (Boundedness).** $\{\mathbf{B}_k\} \subseteq \mathbb{R}^{n\times n}$ is a bounded sequence.

Most practically relevant quasi-Newton schemes should have no issues satisfying the above assumption, especially when the gradient $\nabla f$ is Lipschitz continuous on an appropriate level set. Indeed, many of these techniques yield Hessian approximations which satisfy additional properties such as symmetry (which we omitted because it is unnecessary for the theory below) or positive definiteness.

The following result is an immediate consequence of Assumption 3.1.

**Lemma 3.2 (Gradient approximation).** *Let $\mu_k \to \infty$. Then $\mathbf{B}_k + \mu_k \mathbf{I}$ is invertible for sufficiently large $k \in \mathbb{N}$, and*

$$\lim_{k\to\infty} \frac{(\mathbf{B}_k + \mu_k \mathbf{I})^{-1}\mathbf{y}}{\|(\mathbf{B}_k + \mu_k \mathbf{I})^{-1}\mathbf{y}\|} = \frac{\mathbf{y}}{\|\mathbf{y}\|} \quad \text{for all } \mathbf{y} \in \mathbb{R}^n \setminus \{0\}.$$

The above result defines more precisely the intuitive relationship mentioned in Section 2; that is, if the regularization parameter is sufficiently large, then the regularized Newton equation (10) admits a unique solution, and the resulting vector will approximate the negative gradient direction as $\mu_k \to \infty$.

Another consequence of Lemma 3.2 is that the method performs infinitely many successful steps. This follows from the fact that $\mathbf{d}_k$ becomes ever smaller and approaches the (local) steepest descent direction when $\mu_k \to \infty$, thus leading to a local descent step which satisfies the sufficient decrease condition from Step 2 of the algorithm.

**Lemma 3.3 (Well-definedness).** *Assume that $\mathbf{g}_k \neq 0$ for all $k$. Then Algorithm 2.1 performs infinitely many successful or highly successful steps.*

*Proof.* Assume by contradiction that there exists $k_0 \in \mathbb{N}$ such that all steps $k \geq k_0$ are unsuccessful. In particular, this implies $\mu_k \to \infty$ as $k \to \infty$ and $\mathbf{x}_k = \mathbf{x}_{k_0}$ for all $k \geq k_0$. Since $\{\mathbf{B}_k\}$ is a bounded sequence, it follows from Lemma 3.2 that $\mathbf{B}_k + \mu_k \mathbf{I}$ is invertible for sufficiently large $k$, that $\mathbf{d}_k \to 0$, and $\mathbf{d}_k/\|\mathbf{d}_k\| \to -\mathbf{g}_{k_0}/\|\mathbf{g}_{k_0}\|$. Moreover, the regularized Newton equation (10) implies that $\mu_k\|\mathbf{d}_k\| \to \|\mathbf{g}_{k_0}\|$. It is easy to deduce from these limit relations that

$$\text{pred}_k = \frac{\mu_k}{2}\|\mathbf{d}_k\|^2 - \frac{1}{2}\mathbf{g}_k^\mathsf{T}\mathbf{d}_k > p_{\min}\|\mathbf{g}_k\|\|\mathbf{d}_k\| \quad \text{for sufficiently large } k$$

(simply divide this inequality by $\|\mathbf{d}_k\|$ and recall that $p_{\min} \in (0,1)$). Hence, the algorithm must eventually perform only Step 3u, which means that $\text{ared}_k \leq c_1 \text{pred}_k$ for all $k \geq k_0$ sufficiently large. It then follows that

$$f(\mathbf{x}_{k_0} + \mathbf{d}_k) - f(\mathbf{x}_{k_0}) = -\text{ared}_k \geq -c_1 \text{pred}_k = \frac{c_1}{2}\mathbf{g}_{k_0}^\mathsf{T}\mathbf{d}_k - \frac{c_1\mu_k}{2}\|\mathbf{d}_k\|^2 \quad \text{for } k \geq k_0. \tag{11}$$

We now divide both sides of this inequality by $t_k := \|\mathbf{d}_k\|$. Recalling that $\mathbf{d}_k/\|\mathbf{d}_k\| \to -\mathbf{g}_{k_0}/\|\mathbf{g}_{k_0}\|$, it follows that the left-hand side becomes

$$\frac{f\left(\mathbf{x}_{k_0} + t_k \frac{\mathbf{d}_k}{\|\mathbf{d}_k\|}\right) - f(\mathbf{x}_{k_0})}{t_k} \to \nabla f(\mathbf{x}_{k_0})^\mathsf{T} \frac{-\mathbf{g}_{k_0}}{\|\mathbf{g}_{k_0}\|} = -\|\mathbf{g}_{k_0}\|. \tag{12}$$

Conversely, recalling that $\mu_k\|\mathbf{d}_k\| \to \|\mathbf{g}_{k_0}\|$, the right-hand side of (11) divided by $t_k$ satisfies

$$\frac{c_1}{2}\mathbf{g}_{k_0}^\mathsf{T}\frac{\mathbf{d}_k}{\|\mathbf{d}_k\|} - \frac{c_1\mu_k}{2}\|\mathbf{d}_k\| \to \frac{c_1}{2}\mathbf{g}_{k_0}^\mathsf{T}\frac{-\mathbf{g}_{k_0}}{\|\mathbf{g}_{k_0}\|} - \frac{c_1}{2}\|\mathbf{g}_{k_0}\| = -c_1\|\mathbf{g}_{k_0}\|. \tag{13}$$

Since $c_1 \in (0,1)$, it then follows from (12), (13) that $\|\mathbf{g}_{k_0}\| = 0$, a contradiction. □

The following result builds upon the well-definedness of the algorithm and shows that it achieves asymptotic stationarity.

**Theorem 3.4 (Global convergence I).** *Let $f$ be bounded from below, and let $\{\mathbf{x}_k\}$ be generated by Algorithm 2.1. Then $\liminf_{k\to\infty} \|\mathbf{g}_k\| = 0$; in particular, given any $\varepsilon > 0$, the algorithm terminates with $\|\mathbf{g}_k\| < \varepsilon$ after finitely many iterations.*

*Proof.* Let $\mathcal{S} \subseteq \mathbb{N}$ be the set of indices where Algorithm 2.1 performs a successful or highly successful step. Note that $|\mathcal{S}| = \infty$ by Lemma 3.3. Assume that

$$\liminf_{k\to\infty} \|\mathbf{g}_k\| > 0. \tag{14}$$

Since every step $k \in \mathcal{S}$ is successful, we have by definition that

$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) \geq c_1 \text{pred}_k \geq p_{\min} c_1 \|\mathbf{g}_k\| \|\mathbf{d}_k\| \quad \text{for every } k \in \mathcal{S}.$$

By (14), there exist $k_0 \in \mathbb{N}$ and $\varepsilon > 0$ such that $\|\mathbf{g}_k\| \geq \varepsilon$ for all $k \geq k_0$. Using the fact that $f$ is bounded from below, we obtain

$$\infty > \sum_{k\in\mathbb{N}} \big[f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})\big] = \sum_{k\in\mathcal{S}} \big[f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})\big] \geq p_{\min} c_1 \varepsilon \sum_{k\in\mathcal{S},\, k\geq k_0} \|\mathbf{d}_k\| \tag{15}$$

and, in particular, $\mathbf{d}_k \to_{\mathcal{S}} 0$. Since every step $k \in \mathcal{S}$ is successful, we have $(\mathbf{B}_k + \mu_k \mathbf{I})\mathbf{d}_k = -\mathbf{g}_k$ for all $k \in \mathcal{S}$, which by (14) implies $\mu_k \to_{\mathcal{S}} +\infty$. In particular, the algorithm also performs infinitely many unsuccessful steps (i.e., $|\mathbb{N} \setminus \mathcal{S}| = \infty$), and $\mu_k \to +\infty$ since $\mu_k$ cannot decrease during unsuccessful iterations.

Now, since $\mathcal{S}$ and $\mathbb{N} \setminus \mathcal{S}$ are infinite, we may choose an infinite set $\mathcal{S}' \subseteq \mathcal{S}$ such that $k - 1 \in \mathbb{N} \setminus \mathcal{S}$ whenever $k \in \mathcal{S}'$. Since $\mathbf{x}_k$ is not updated in unsuccessful steps, it follows from (15) that

$$\infty > p_{\min} c_1 \varepsilon \sum_{k\in\mathcal{S},\, k\geq k_0} \|\mathbf{d}_k\| = p_{\min} c_1 \varepsilon \sum_{k\in\mathcal{S},\, k\geq k_0} \|\mathbf{x}_{k+1} - \mathbf{x}_k\| = p_{\min} c_1 \varepsilon \sum_{k\geq k_0} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|.$$

Hence $\{\mathbf{x}_k\}_{k\in\mathbb{N}}$ is a Cauchy sequence, and thus convergent. Let $\bar{\mathbf{x}}$ denote its limit point. In particular, we then obtain $\mathbf{x}_{k-1} \to_{\mathcal{S}'} \bar{\mathbf{x}}$; thus, using $\mu_k \to +\infty$ and arguing as in the proof of Lemma 3.3, it follows that the steps $k - 1$, $k \in \mathcal{S}'$, must be successful for sufficiently large $k \in \mathcal{S}'$. This is a contradiction. □

Note that the counterpart of Theorem 3.4 also holds for trust-region methods under the same set of assumptions. Moreover, the technique of proof used here is related to the corresponding one known for trust-region methods. Nevertheless, we stress that one has to be careful in translating the standard trust-region proof to our regularization framework since well-known properties of the solution of the trust-region subproblem may not hold in our case.

Similar to the theory of trust-region methods, we can use Theorem 3.4 to obtain a stronger convergence result under an additional assumption.

**Theorem 3.5 (Global convergence II).** *Let $f$ be bounded from below, and let $\{\mathbf{x}_k\}$ be generated by Algorithm 2.1. Suppose that $\nabla f$ is uniformly continuous on a set $X \subseteq \mathbb{R}^n$ satisfying $\{\mathbf{x}_k\} \subseteq X$. Then $\lim_{k\to\infty} \|\mathbf{g}_k\| = 0$; in particular, every accumulation point of $\{\mathbf{x}_k\}$ is a stationary point of $f$.*

*Proof.* Assume there exists $\delta > 0$ and a subsequence $\{\mathbf{x}_k\}_K$ such that

$$\|\mathbf{g}_k\| \geq 2\delta \quad \text{for all } k \in K.$$

Since $\liminf_{k\to\infty} \|\mathbf{g}_k\| = 0$ by Theorem 3.4, we can find, for each $k \in K$, an index $\ell(k) > k$ such that

$$\|\mathbf{g}_l\| \geq \delta \quad \text{for all } k \leq l < \ell(k), \qquad \text{and} \qquad \|\mathbf{g}_{\ell(k)}\| < \delta, \quad k \in K.$$

For an arbitrary $k \in K$ and a successful or highly successful iteration $l$ with $k \leq l < \ell(k)$, we obtain

$$f(\mathbf{x}_l) - f(\mathbf{x}_{l+1}) \geq c_1 \text{pred}_k \geq p_{\min} c_1 \|\mathbf{g}_l\| \|\mathbf{d}_l\| \geq p_{\min} c_1 \delta \|\mathbf{x}_{l+1} - \mathbf{x}_l\|.$$

The same inequality holds for $l$ being unsuccessful simply because $\mathbf{x}_{l+1} = \mathbf{x}_l$ in this case. This implies

$$\begin{aligned} p_{\min} c_1 \delta \|\mathbf{x}_{\ell(k)} - \mathbf{x}_k\| \leq p_{\min} c_1 \delta \sum_{l=k}^{\ell(k)-1} \|\mathbf{x}_{l+1} - \mathbf{x}_l\| &\leq \sum_{l=k}^{\ell(k)-1} \big(f(\mathbf{x}_l) - f(\mathbf{x}_{l+1})\big) \\ &= f(\mathbf{x}_k) - f(\mathbf{x}_{\ell(k)}) \end{aligned}$$

for all $k \in K$. Since $f$ is bounded from below and $\{f(\mathbf{x}_k)\}$ is monotonically decreasing, we obtain $f(\mathbf{x}_k) - f(\mathbf{x}_{\ell(k)}) \to 0$ for $k \to \infty$. This implies $\|\mathbf{x}_{\ell(k)} - \mathbf{x}_k\| \to_K 0$. The uniform continuity of $\nabla f$ on the set $X$ therefore yields

$$\|\nabla f(\mathbf{x}_{\ell(k)}) - \nabla f(\mathbf{x}_k)\| \to_K 0.$$

On the other hand, the choice of the index $\ell(k)$ implies

$$\|\nabla f(\mathbf{x}_{\ell(k)}) - \nabla f(\mathbf{x}_k)\| \geq \|\nabla f(\mathbf{x}_k)\| - \|\nabla f(\mathbf{x}_{\ell(k)})\| \geq 2\delta - \delta = \delta.$$

This contradiction completes the proof. □

We close this section by noting that regularization techniques like in Algorithm 2.1 are sometimes used in order to prove local fast convergence properties for Newton-type methods. This corresponds to the choice $\mathbf{B}_k := \nabla^2 f(\mathbf{x}_k)$ as the exact Hessian. Using a more refined update of the regularization parameter, assuming a local error bound condition and the Hessian of $f$ to be locally Lipschitz continuous, it is possible to verify local quadratic convergence for convex objective functions, cf. [15, 23, 24]. Since our focus is on large-scale problems, our subsequent analysis concentrates on $\mathbf{B}_k$ being computed by limited memory quasi-Newton matrices.

# 4 Regularized Quasi-Newton Matrices

This section provides the details of *limited memory* type implementations of quasi-Newton methods. Some of the material below can be applied with minimal modifications to full memory quasi-Newton methods, but we forgo these investigations due to our focus on large-scale optimization.

In keeping with conventional limited memory notation, we assume an algorithmic framework where the last $m$ variable steps $\mathbf{s}_i := \mathbf{x}_{i+1} - \mathbf{x}_i$ are tracked together with the corresponding gradient differences $\mathbf{y}_i := \mathbf{g}_{i+1} - \mathbf{g}_i$, where $\mathbf{g}_i = \nabla f(\mathbf{x}_i)$. For convenience of notation, we aggregate these in the matrices

$$\mathbf{S}_k := [\mathbf{s}_{k-m} \cdots \mathbf{s}_{k-1}] \in \mathbb{R}^{n\times m} \quad \text{and} \quad \mathbf{Y}_k := [\mathbf{y}_{k-m} \cdots \mathbf{y}_{k-1}] \in \mathbb{R}^{n\times m}.$$

If fewer than $m$ previous iterates are available, that is, if $k < m$, we set

$$\mathbf{S}_k := [\mathbf{s}_0 \cdots \mathbf{s}_{k-1}] \in \mathbb{R}^{n\times k} \quad \text{and} \quad \mathbf{Y}_k := [\mathbf{y}_0 \cdots \mathbf{y}_{k-1}] \in \mathbb{R}^{n\times k}.$$

These definitions may seem like a mere matter of notation, but there are actually quite pragmatic arguments why $\mathbf{S}$ and $\mathbf{Y}$ should be treated as matrices instead of collections of vectors. Many limited memory operations can be formulated as loops over the recurrent index $i = 1, \ldots, m$, and the matrix notation sometimes allows us to formulate the underlying calculations as *matrix–vector* operations (instead of a sequence of vector–vector operations). This approach should be used whenever possible in practical implementations because it leverages the power of low-level BLAS (basic linear algebra subprograms) and parallelism, providing a significant increase in computational efficiency.

**Remark 4.1 (Rejected quasi-Newton updates).** For the sake of simplicity and to avoid notational overhead, we assume that the algorithm always "accepts" the data pair $(\mathbf{s}_k, \mathbf{y}_k)$ in each successful iteration. This is not the case for some quasi-Newton schemes, especially for nonconvex objective functions (see below). Treating "rejected" quasi-Newton updates does not substantively change the subsequent analysis.

Most limited memory quasi-Newton methods implicitly generate a so-called *compact representation* of the form

$$\mathbf{B}_k = \mathbf{B}_{0,k} + \mathbf{A}_k \mathbf{Q}_k^{-1} \mathbf{A}_k^\mathsf{T}, \tag{16}$$

where $\mathbf{Q}_k \in \mathbb{R}^{s\times s}$ is a nonsingular symmetric matrix, $\mathbf{A}_k \in \mathbb{R}^{n\times s}$, and $s \ll n$ is a constant depending on the particular quasi-Newton scheme. For instance, $s = 2m$ in limited memory BFGS methods, and $s = m$ for limited memory SR1.

The above representation provides a very convenient framework for the regularization approach: given $\mu \geq 0$, the regularized Hessian approximation can be written as

$$\mathbf{B}_k + \mu\mathbf{I} = (\mathbf{B}_{0,k} + \mu\mathbf{I}) + \mathbf{A}_k \mathbf{Q}_k^{-1} \mathbf{A}_k^\mathsf{T}.$$

This facilitates the application of low-rank update formulas to compute the regularized Newton step both explicitly and extremely cheaply. To this end, let $\hat{\mathbf{B}}_k := \mathbf{B}_k + \mu\mathbf{I}$ and $\hat{\mathbf{B}}_{0,k} := \mathbf{B}_{0,k} + \mu\mathbf{I}$. Then the Sherman–Morrison–Woodbury formula implies that

$$\hat{\mathbf{B}}_k^{-1} = \hat{\mathbf{B}}_{0,k}^{-1} - \hat{\mathbf{B}}_{0,k}^{-1} \mathbf{A}_k (\mathbf{Q}_k + \mathbf{A}_k^\mathsf{T} \hat{\mathbf{B}}_{0,k}^{-1} \mathbf{A}_k)^{-1} \mathbf{A}_k^\mathsf{T} \hat{\mathbf{B}}_{0,k}^{-1} \tag{17}$$

provided that $\hat{\mathbf{B}}_{0,k}$ is nonsingular. Note that $\hat{\mathbf{B}}_{0,k}$ is usually a diagonal matrix whose inversion is trivial. Moreover, the inner matrix $\mathbf{Q}_k + \mathbf{A}_k^\mathsf{T} \hat{\mathbf{B}}_{0,k}^{-1} \mathbf{A}_k$ is of size $s \times s$, so that its inversion can be carried out cheaply in relation to the dimension $n$. By the Woodbury matrix identity, the invertibility of this inner matrix is equivalent to that of $\hat{\mathbf{B}}_k$.

In the following, we shall mainly assume that the initial matrix $\mathbf{B}_{0,k}$ is chosen as a scalar multiple of the identity, $\mathbf{B}_{0,k} := \gamma_k \mathbf{I}$. Writing $\hat{\gamma}_k := \gamma_k + \mu$, it then follows that

$$\hat{\mathbf{B}}_k^{-1} = \hat{\gamma}_k^{-1} \mathbf{I} - \hat{\gamma}_k^{-2} \mathbf{A}_k (\mathbf{Q}_k + \hat{\gamma}_k^{-1} \mathbf{A}_k^\mathsf{T} \mathbf{A}_k)^{-1} \mathbf{A}_k^\mathsf{T}. \tag{18}$$

The practical efficiency of quasi-Newton methods significantly depends on the memorization and re-use of previously computed quantities. To this end, observe that the quasi-Newton recurrence implies

$$\mathbf{s}_k = -\hat{\mathbf{B}}_k^{-1} \mathbf{g}_k = -\hat{\gamma}_k^{-1} \mathbf{g}_k + \hat{\gamma}_k^{-2} \mathbf{A}_k \mathbf{p}_k, \tag{19}$$

where

$$\mathbf{p}_k := (\mathbf{Q}_k + \hat{\gamma}_k^{-1}\mathbf{A}_k^\mathsf{T}\mathbf{A}_k)^{-1}\mathbf{A}_k^\mathsf{T}\mathbf{g}_k. \tag{20}$$

Thus, the main computational cost occurs in forming the product $\mathbf{A}_k^\mathsf{T}\mathbf{g}_k$, the solution of an $s \times s$ symmetric linear equation to obtain $\mathbf{p}_k$, and the product $\mathbf{A}_k\mathbf{p}_k$. In addition, the matrices $\mathbf{A}_k$ and $\mathbf{Q}_k$ need to be updated in each iteration, and the matrix $\mathbf{A}_k^\mathsf{T}\mathbf{A}_k$ needs to be available. As we shall see later, it is possible to reduce the cost of these computations by using the inherent dependency of the underlying formulas.

**Remark 4.2 (Modifying the secant equation).** Instead of compact representations, it is also possible to combine the regularization and quasi-Newton techniques by directly approximating the sum $\nabla^2 f(\mathbf{x}_k) + \mu\mathbf{I}$; see [22]. This idea is based on the fact that the regularized Hessian satisfies (approximately) the modified secant equation

$$(\nabla^2 f(\mathbf{x}_k) + \mu_k\mathbf{I})\mathbf{s}_k \approx \mathbf{y}_k + \mu_k\mathbf{s}_k.$$

Thus, an approximation $\hat{\mathbf{B}}_k$ to $\nabla^2 f(\mathbf{x}_k) + \mu_k\mathbf{I}$ can be constructed by taking a modified initial guess $\hat{\mathbf{B}}_{0,k} := \mathbf{B}_{0,k} + \mu_k\mathbf{I}$ and applying an arbitrary quasi-Newton scheme to the modified pair $(\mathbf{S}_k, \hat{\mathbf{Y}}_k) := (\mathbf{S}_k, \mathbf{Y}_k + \mu_k\mathbf{S}_k)$. For certain quasi-Newton schemes, it turns out that this approach actually yields the same results as the one based on compact representations (see Sections 4.2 and 4.3). In general, however, the two approaches are distinct.

### 4.1 Broyden–Fletcher–Goldfarb–Shanno (BFGS)

The BFGS update is often considered the most successful quasi-Newton scheme. Following [5], the compact representation of L-BFGS is given by

$$\mathbf{B}_k = \gamma_k\mathbf{I} - \begin{bmatrix}\mathbf{S}_k & \mathbf{Y}_k\end{bmatrix}\begin{bmatrix}\gamma_k^{-1}\mathbf{S}_k^\mathsf{T}\mathbf{S}_k & \gamma_k^{-1}\mathbf{L}_k \\ \gamma_k^{-1}\mathbf{L}_k^\mathsf{T} & -\mathbf{D}_k\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{S}_k^\mathsf{T} \\ \mathbf{Y}_k^\mathsf{T},\end{bmatrix} \tag{21}$$

where

$$\mathbf{D}_k := \mathbf{D}(\mathbf{S}_k^\mathsf{T}\mathbf{Y}_k) \quad \text{and} \quad \mathbf{L}_k := \mathbf{L}(\mathbf{S}_k^\mathsf{T}\mathbf{Y}_k) \tag{22}$$

(recall that $\mathbf{D}(\cdot)$ denotes the diagonal part and $\mathbf{L}(\cdot)$ the strict lower triangle of a given matrix). This can be written in the form (16) by defining

$$\mathbf{A_k} := \begin{bmatrix}\mathbf{S}_k & \mathbf{Y}_k\end{bmatrix} \quad \text{and} \quad \mathbf{Q}_k := \begin{bmatrix}-\gamma_k^{-1}\mathbf{S}_k^\mathsf{T}\mathbf{S}_k & -\gamma_k^{-1}\mathbf{L}_k \\ -\gamma_k^{-1}\mathbf{L}_k^\mathsf{T} & \mathbf{D}_k\end{bmatrix}. \tag{23}$$

Note that $\mathbf{Q}_k \in \mathbb{R}^{2m\times 2m}$.

The BFGS formula has a significant advantage in that the well-definedness of the updates can be controlled. More specifically, assuming that $\mathbf{s}_k^\mathsf{T}\mathbf{y}_k > 0$ for all $k$, it can be shown that the BFGS matrix $\mathbf{B}_k$ is positive definite, so that the regularized BFGS matrix $\hat{\mathbf{B}}_k = \mathbf{B}_k + \mu\mathbf{I}$ is also positive definite and therefore nonsingular. By the Woodbury matrix identity, this implies that the inner matrix $\mathbf{Q}_k + \hat{\gamma}_k^{-1}\mathbf{A}_k^\mathsf{T}\mathbf{A}_k$ in (18) is invertible, and thus the regularized Newton step is well-defined for all $\mu \geq 0$.

In practice, the well-definedness is controlled by means of a so-called *cautious updating* mechanism [14]. The previous limited memory data is only updated with the next pair $(\mathbf{s}_k, \mathbf{y}_k)$ if

$$\mathbf{y}_k^\mathsf{T}\mathbf{s}_k \geq \varepsilon\|\mathbf{s}_k\|^2, \tag{24}$$

where $\varepsilon > 0$ is some predefined constant. This guarantees that the L-BFGS matrices $\mathbf{B}_k$ are uniformly positive definite. If $\nabla f$ is Lipschitz continuous on the set of iterates (or an appropriate level set), then (24) also guarantees that $\{\mathbf{B}_k\}$ is bounded.

**Updating L-BFGS information**

We now describe how the L-BFGS information can be updated in an efficient manner. To avoid repetition, we only describe the case where the previous information is already "full", i.e., where at least $m$ previous data pairs $(\mathbf{s}_i, \mathbf{y}_i)$ are available. The modifications necessary to treat the initial steps essentially amount to re-indexing and will not be detailed here.

Much of the computational effort of regularized L-BFGS can be mitigated by memorizing certain intermediate results. Motivated by a related trust-region approach in [2], we track, in addition to the matrices $\mathbf{S}_k$ and $\mathbf{Y}_k$, the quantities

$$\mathbf{A}_k^\mathsf{T}\mathbf{A}_k \in \mathbb{R}^{2m\times 2m} \quad \text{and} \quad \mathbf{A}_k^\mathsf{T}\mathbf{g}_k \in \mathbb{R}^{2m}.$$

Both of these quantities are necessary for the computation of the regularized quasi-Newton step (19), (20), but they also occur in other places of the iteration and updating process, so that memorizing them can save redundant computational effort. Recall that $\mathbf{A}_k = [\mathbf{S}_k\,,\,\mathbf{Y}_k]$, so that in particular

$$\mathbf{A}_k^\mathsf{T}\mathbf{A}_k = \begin{pmatrix} \mathbf{S}_k^\mathsf{T}\mathbf{S}_k & \mathbf{S}_k^\mathsf{T}\mathbf{Y}_k \\ \mathbf{Y}_k^\mathsf{T}\mathbf{S}_k & \mathbf{Y}_k^\mathsf{T}\mathbf{Y}_k \end{pmatrix} \quad \text{and} \quad \mathbf{A}_k^\mathsf{T}\mathbf{g}_k = \begin{pmatrix} \mathbf{S}_k^\mathsf{T}\mathbf{g}_k \\ \mathbf{Y}_k^\mathsf{T}\mathbf{g}_k \end{pmatrix}.$$

Hence, the matrix $\mathbf{A}_k^\mathsf{T}\mathbf{A}_k$ contains the blocks $\mathbf{S}_k^\mathsf{T}\mathbf{S}_k$, $\mathbf{L}_k$, and $\mathbf{D}_k$ from (23) as submatrices.

When passing from $k$ to $k+1$, these matrices and vectors can be updated as follows. If the data pair $(\mathbf{s}_k, \mathbf{y}_k)$ is rejected, then $\mathbf{A}_k$ remains unchanged, and we may update $\mathbf{A}_k^\mathsf{T}\mathbf{g}_k$ by direct computation. If the data pair is accepted, then the updating process requires more care since both $\mathbf{A}_k^\mathsf{T}\mathbf{A}_k$ and $\mathbf{A}_k^\mathsf{T}\mathbf{g}_k$ need to be incremented. In this case, the new matrices $\mathbf{S}_{k+1}$ and $\mathbf{Y}_{k+1}$ consist of the last $m-1$ columns of the old matrices $\mathbf{S}_k$ and $\mathbf{Y}_k$, respectively, to which the new vectors $\mathbf{s}_k$ and $\mathbf{y}_k$ are appended in the last column. We then begin by computing the vectors

$$\mathbf{v} := \mathbf{A}_k^\mathsf{T}\mathbf{s}_k = -\hat{\gamma}_k^{-1}\mathbf{A}_k^\mathsf{T}\mathbf{g}_k + \hat{\gamma}_k^{-2}(\mathbf{A}_k^\mathsf{T}\mathbf{A}_k)\mathbf{p}_k, \qquad \mathbf{w} := \mathbf{A}_{k+1}^\mathsf{T}\mathbf{g}_{k+1}, \tag{25}$$

where $\mathbf{p}_k$ is given by (20); as well as the scalar quantities

$$\alpha := \mathbf{s}_k^\mathsf{T}\mathbf{s}_k, \quad \beta := \mathbf{s}_k^\mathsf{T}\mathbf{y}_k, \quad \gamma := \mathbf{y}_k^\mathsf{T}\mathbf{y}_k.$$

This information is then used to update $\mathbf{A}_k^\mathsf{T}\mathbf{A}_k$ blockwise using the formulas

$$\mathbf{S}_{k+1}^\mathsf{T}\mathbf{S}_{k+1} = \begin{bmatrix} (\mathbf{S}_k^\mathsf{T}\mathbf{S}_k)_{2:m,2:m} & \mathbf{v}_{2:m} \\ * & \alpha \end{bmatrix}, \tag{26a}$$

$$\mathbf{S}_{k+1}^\mathsf{T}\mathbf{Y}_{k+1} = \begin{bmatrix} (\mathbf{S}_k^\mathsf{T}\mathbf{Y}_k)_{2:m,2:m} & \mathbf{w}_{1:m-1} - (\mathbf{A}_k^\mathsf{T}\mathbf{g}_k)_{2:m} \\ \mathbf{v}_{m+2:2m}^\mathsf{T} & \beta \end{bmatrix}, \tag{26b}$$

$$\mathbf{Y}_{k+1}^\mathsf{T}\mathbf{Y}_{k+1} = \begin{bmatrix} (\mathbf{Y}_k^\mathsf{T}\mathbf{Y}_k)_{2:m,2:m} & \mathbf{w}_{m+1:2m-1} - (\mathbf{A}_k^\mathsf{T}\mathbf{g}_k)_{m+2:2m} \\ * & \gamma \end{bmatrix}, \tag{26c}$$

where "$*$" is given by symmetry, and expressions of the form $(\mathbf{S}_k^\mathsf{T}\mathbf{S}_k)_{2:m,2:m}$ or $\mathbf{v}_{2:m}$ denote submatrices and -vectors built from the subscripted index ranges. Finally, we have $\mathbf{Y}_{k+1}^\mathsf{T}\mathbf{S}_{k+1} = (\mathbf{S}_{k+1}^\mathsf{T}\mathbf{Y}_{k+1})^\mathsf{T}$, and the new vector $\mathbf{A}_{k+1}^\mathsf{T}\mathbf{g}_{k+1}$ is given by $\mathbf{w}$.

**Computational complexity**

Let us now comment on the complexity involved in the computation of the regularized quasi-Newton step. Assuming that the product $\mathbf{A}_k^\mathsf{T}\mathbf{g}_k$ has been formed, the main cost is the solution of a $2m \times 2m$ symmetric linear system to form $\mathbf{p}_k$, and the multiplication of $\mathbf{p}_k$ with the $n \times 2m$ matrix $\mathbf{A}_k$. Hence, the complexity of the regularized quasi-Newton equation is $2mn + O(m^3)$ multiplications.

When a step is successful, the existing data needs to be updated according to the formulas developed above. This incurs at most $2mn$ multiplications (or less if the data pair is rejected). Hence, the overall computational effort is at most $2mn$ multiplications for an unsuccessful step, and $4mn$ for a successful step.

The computational cost of the $2m \times 2m$ linear equation (20) for the computation of $\mathbf{p}_k$ is of order $O(m^3)$. Thus, if $m \ll n$, this cost is negligible in comparison to $mn$. The slight computational overhead induced by this linear equation can be mitigated further by using a technique from [4], which reduces the $2m \times 2m$ inversion to two $m \times m$ Cholesky factorizations. In any case, we have found in our experiments that the time spent on this part of the computation is negligible.

## 4.2 Symmetric rank-one (SR1)

For SR1, the compact representation takes on the form

$$\mathbf{B}_k = \mathbf{B}_{0,k} + (\mathbf{Y}_k - \mathbf{B}_{0,k}\mathbf{S}_k)(\mathbf{D}_k + \mathbf{L}_k + \mathbf{L}_k^\mathsf{T} - \mathbf{S}_k^\mathsf{T}\mathbf{B}_{0,k}\mathbf{S}_k)^{-1}(\mathbf{Y}_k - \mathbf{B}_{0,k}\mathbf{S}_k)^\mathsf{T}, \tag{27}$$

where $\mathbf{D}_k$ and $\mathbf{L}_k$ are again given by (22). This can be written in the form (16) by defining

$$\mathbf{A}_k := \mathbf{Y}_k - \mathbf{B}_{0,k}\mathbf{S}_k \quad \text{and} \quad \mathbf{Q}_k := \mathbf{D}_k + \mathbf{L}_k + \mathbf{L}_k^\mathsf{T} - \mathbf{S}_k^\mathsf{T}\mathbf{B}_{0,k}\mathbf{S}_k. \tag{28}$$

Note that $\mathbf{Q}_k \in \mathbb{R}^{m\times m}$ in this case.

If $\mathbf{B}_{0,k} = \gamma_k\mathbf{I}$, then (27) can be simplified to

$$\mathbf{B}_k = \gamma_k\mathbf{I} + (\mathbf{Y}_k - \gamma_k\mathbf{S}_k)(\mathbf{D}_k + \mathbf{L}_k + \mathbf{L}_k^\mathsf{T} - \gamma_k\mathbf{S}_k^\mathsf{T}\mathbf{S}_k)(\mathbf{Y}_k - \gamma_k\mathbf{S}_k)^\mathsf{T}. \tag{29}$$

The well-definedness of the SR1 update is hard to guarantee in practice because the underlying rank one formula involves a denominator of the form $(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^\mathsf{T}\mathbf{s}_k$, which can vanish. Thus, when applying formula (17) to the SR1 setting, it is important to clarify how this situation is handled. Note that it is not possible to predict which new data $(\mathbf{s}_{k+1}, \mathbf{y}_{k+1})$ might lead to ill-conditioning because this crucially depends on the previous information $(\mathbf{S}_k, \mathbf{Y}_k)$. In fact, even the discarding of old data at some point during the iteration might have an influence and change the well-definedness of the SR1 update.

Fortunately, there is a simple and effective way of skipping ill-conditioned updates "on the fly", i.e., during the computation of the quasi-Newton step. This effectively amounts to skipping an intermediate step $(\mathbf{s}_i, \mathbf{y}_i)$ when necessary and proceeding the SR1 update with $(\mathbf{s}_{i+1}, \mathbf{y}_{i+1})$ instead. It was observed in [5] that ill-definedness of one of these updates amounts to the singularity of a principal minor of $\mathbf{Q}_k$, or equivalently, to a vanishing pivot element during a triangularization of $\mathbf{Q}_k$. When this occurs, it is proposed in [5] to skip the update by essentially ignoring the current row and column of $\mathbf{Q}_k$, and the current column of $\mathbf{A}_k$ (which contains the corresponding vectors $\mathbf{s}_i$ and $\mathbf{y}_i$).

The above procedure can be adapted to the *regularized* SR1 setting by observing that the SR1 update "commutes" with the regularization in a certain sense. More specifically, if

$\mathbf{B}_k = \mathcal{U}(\mathbf{B}_{0,k}, \mathbf{S}, \mathbf{Y})$ denotes the SR1 update, then

$$\mathcal{U}(\mathbf{B}_{0,k} + \mu\mathbf{I}, \mathbf{S}, \mathbf{Y} + \mu\mathbf{S}) = \mathcal{U}(\mathbf{B}_{0,k}, \mathbf{S}, \mathbf{Y}) + \mu\mathbf{I}$$

for all $\mu \geq 0$, provided that the left side exists. Moreover, an easy calculation shows that the matrix $\mathbf{Q}_k + \mathbf{A}_k^\mathsf{T}\hat{\mathbf{B}}_{0,k}^{-1}\mathbf{A}_k$ from (17), which needs to be inverted for the computation of the regularized Newton step, coincides (up to scaling) with the analogue of $\mathbf{Q}_k$ which would arise for the SR1 update corresponding to $\hat{\mathbf{B}}_{0,k}$ and $\mathbf{Y}_k + \mu\mathbf{S}_k$.

**Updating L-SR1 information**

The quantities involved in the L-SR1 computations can be updated in a similar fashion to the L-BFGS case; see Section 4.1. We again maintain the quantities

$$\mathbf{S}_k^\mathsf{T}\mathbf{S}_k,\ \mathbf{S}_k^\mathsf{T}\mathbf{Y}_k,\ \mathbf{Y}_k^\mathsf{T}\mathbf{Y}_k \in \mathbb{R}^{m\times m} \quad \text{and} \quad \mathbf{S}_k^\mathsf{T}\mathbf{g}_k,\ \mathbf{Y}_k^\mathsf{T}\mathbf{g}_k \in \mathbb{R}^m. \tag{30}$$

These can be formed and updated as before. It is easy to see that these can be used to directly form the matrices $\mathbf{A}_k$ and $\mathbf{Q}_k$, the product $\mathbf{A}_k^\mathsf{T}\mathbf{g}_k$, and the matrix $\mathbf{A}_k^\mathsf{T}\mathbf{A}_k$.

**Computational complexity**

The computational cost of the regularized L-SR1 method is as follows. In each successful iteration, the quantities (30) are updated, and the matrix $\mathbf{A}_k = \mathbf{Y}_k - \mathbf{B}_{0,k}\mathbf{S}_k$ is formed. Using the techniques from Section 4.1, these operations require $3mn$ multiplications.

Moreover, the quasi-Newton step needs to be calculated in each step, which entails the solution of an $m \times m$ symmetric linear system to obtain $\mathbf{p}_k$, and the multiplication of $\mathbf{p}_k$ with the $n \times m$ matrix $\mathbf{A}_k$, requiring another $mn$ multiplications.

In total, the cost of a successful step is therefore $4mn$ multiplications, and the cost of an unsuccessful step is $mn$ multiplications (down from $2mn$ in the BFGS case).

## 4.3 Powell-symmetric-Broyden (PSB)

As a third example, we include the classical PSB formula from [20]. This approach is interesting because the PSB update is always well-defined and has certain well-known minimality properties. The PSB update is given by

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)\mathbf{s}_k^\mathsf{T} + \mathbf{s}_k(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^\mathsf{T}}{\mathbf{s}_k^\mathsf{T}\mathbf{s}_k} - \frac{(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)^\mathsf{T}\mathbf{s}_k}{(\mathbf{s}_k^\mathsf{T}\mathbf{s}_k)^2}\mathbf{s}_k\mathbf{s}_k^\mathsf{T}. \tag{31}$$

The compact representation of PSB is given in the following theorem.

**Theorem 4.3 (Compact representation of PSB).** *The PSB formula admits the compact representation*

$$\mathbf{B}_k = \mathbf{B}_{0,k} + \begin{bmatrix}\mathbf{S}_k & \mathbf{W}_k\end{bmatrix}\begin{bmatrix}0 & \mathbf{U}_k \\ \mathbf{U}_k^\mathsf{T} & \mathbf{L}_k + \mathbf{D}_k + \mathbf{L}_k^\mathsf{T}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{S}_k & \mathbf{W}_k\end{bmatrix}^\mathsf{T}, \tag{32}$$

*where* $\mathbf{W}_k := \mathbf{Y}_k - \mathbf{B}_{0,k}\mathbf{S}_k$, $\mathbf{U}_k$ *is the (non-strictly) upper triangular part of* $\mathbf{S}_k^\mathsf{T}\mathbf{S}_k$, $\mathbf{L}_k$ *is the strictly lower triangular part of* $\mathbf{S}_k^\mathsf{T}\mathbf{W}_k$, *and* $\mathbf{D}_k$ *is the diagonal part of* $\mathbf{S}_k^\mathsf{T}\mathbf{W}_k$.

*Proof.* To simplify some technical details, we restrict the proof to the case where $k = m$ (i.e., the algorithm has performed exactly $m$ steps, and the matrices $\mathbf{S}_k$ and $\mathbf{Y}_k$ are "full"). Observe first that (31) can be rewritten as

$$\mathbf{B}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{s}_k\mathbf{s}_k^\mathsf{T}}{\mathbf{s}_k^\mathsf{T}\mathbf{s}_k}\right)\mathbf{B}_k\left(\mathbf{I} - \frac{\mathbf{s}_k\mathbf{s}_k^\mathsf{T}}{\mathbf{s}_k^\mathsf{T}\mathbf{s}_k}\right) + \begin{bmatrix}\mathbf{s}_k & \mathbf{y}_k\end{bmatrix}\begin{bmatrix}0 & \mathbf{s}_k^\mathsf{T}\mathbf{s}_k \\ \mathbf{s}_k^\mathsf{T}\mathbf{s}_k & \mathbf{s}_k^\mathsf{T}\mathbf{y}_k\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{s}_k & \mathbf{y}_k\end{bmatrix}^\mathsf{T}.$$

Therefore, we can write $\mathbf{B}_k = \mathbf{M}_k + \mathbf{N}_k$, where $\mathbf{M}_k, \mathbf{N}_k$ are recursively defined through the formulas

$$\begin{aligned}\mathbf{M}_0 = \mathbf{B}_{0,k}, \qquad & \mathbf{M}_{i+1} = \mathbf{V}_i\mathbf{M}_i\mathbf{V}_i, \\ \mathbf{N}_0 = 0, \qquad & \mathbf{N}_{i+1} = \mathbf{V}_i\mathbf{N}_i\mathbf{V}_i + \begin{bmatrix}\mathbf{s}_i & \mathbf{y}_i\end{bmatrix}\begin{bmatrix}0 & \mathbf{s}_i^\mathsf{T}\mathbf{s}_i \\ \mathbf{s}_i^\mathsf{T}\mathbf{s}_i & \mathbf{s}_i^\mathsf{T}\mathbf{y}_i\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{s}_i & \mathbf{y}_i\end{bmatrix}^\mathsf{T},\end{aligned}$$

where $\mathbf{V}_i := \mathbf{I} - (\mathbf{s}_i^\mathsf{T}\mathbf{s}_i)^{-1}\mathbf{s}_i\mathbf{s}_i^\mathsf{T}$ for all $i$. Observe now that $\mathbf{V}_0 \cdot \ldots \cdot \mathbf{V}_{k-1} = \mathbf{I} - \mathbf{S}_k\mathbf{U}_k^{-1}\mathbf{S}_k^\mathsf{T}$ by [5, Lem. 2.1], so that

$$\mathbf{M}_k = \big(\mathbf{I} - \mathbf{S}_k\mathbf{U}_k^{-\mathsf{T}}\mathbf{S}_k^\mathsf{T}\big)\mathbf{B}_{0,k}\big(\mathbf{I} - \mathbf{S}_k\mathbf{U}_k^{-1}\mathbf{S}_k^\mathsf{T}\big).$$

We proceed by using (finite) induction to show that

$$\mathbf{N}_i = \mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}}\mathbf{Y}_i^\mathsf{T} + \mathbf{Y}_i\mathbf{U}_i^{-1}\mathbf{S}_i^\mathsf{T} - \mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}}\big(\tilde{\mathbf{L}}_i + \tilde{\mathbf{D}}_i + \tilde{\mathbf{L}}_i^\mathsf{T}\big)\mathbf{U}_i^{-1}\mathbf{S}_i^\mathsf{T} \quad \text{for all } i = 1, \ldots, k, \quad (33)$$

where $\tilde{\mathbf{L}}_i := \mathbf{L}(\mathbf{S}_i^\mathsf{T}\mathbf{Y}_i)$ and $\tilde{\mathbf{D}}_i := \mathbf{D}(\mathbf{S}_i^\mathsf{T}\mathbf{Y}_i)$. Before we verify this formula, we show that it yields the desired compact representation of the PSB formula. Indeed, using (33) and the definitions of the matrices $\tilde{\mathbf{L}}_k, \tilde{\mathbf{D}}_k, \mathbf{L}_k, \mathbf{D}_k$, respectively, we obtain

$$\begin{aligned}\mathbf{B}_k &= \mathbf{M}_k + \mathbf{N}_k \\ &= \mathbf{B}_{0,k} - \mathbf{S}_k\mathbf{U}_k^{-\mathsf{T}}\mathbf{S}_k^\mathsf{T}\mathbf{B}_{0,k} - \mathbf{B}_{0,k}\mathbf{S}_k\mathbf{U}_k^{-1}\mathbf{S}_k^\mathsf{T} + \mathbf{S}_k\mathbf{U}_k^{-\mathsf{T}}\mathbf{Y}_k^\mathsf{T} + \mathbf{Y}_k\mathbf{U}_k^{-1}\mathbf{S}_k^\mathsf{T} \\ &\quad -\mathbf{S}_k\mathbf{U}_k^{-\mathsf{T}}\big(\mathbf{L}_k + \mathbf{D}_k + \mathbf{L}_k^\mathsf{T}\big)\mathbf{U}_k^{-1}\mathbf{S}_k^\mathsf{T}.\end{aligned}$$

On the other hand, exploiting the fact that

$$\begin{bmatrix}0 & \mathbf{U}_k \\ \mathbf{U}_k^\mathsf{T} & \mathbf{L}_k + \mathbf{D}_k + \mathbf{L}_k^\mathsf{T}\end{bmatrix}^{-1} = \begin{bmatrix}-\mathbf{U}_k^{-\mathsf{T}}\big(\mathbf{L}_k + \mathbf{D}_k + \mathbf{L}_k^\mathsf{T}\big)\mathbf{U}_k^{-1} & \mathbf{U}_k^{-\mathsf{T}} \\ \mathbf{U}_k^{-1} & 0\end{bmatrix},$$

using $\mathbf{W}_k = \mathbf{Y}_k - \mathbf{B}_{0,k}\mathbf{S}_k$, and expanding (32), it is easy to see that we obtain the same expression.

Hence it remains to verify (33) by induction. For $i = 1$, we have

$$\mathbf{S}_1 = \begin{bmatrix}s_0\end{bmatrix}, \quad \mathbf{Y}_1 = \begin{bmatrix}y_0\end{bmatrix}, \quad \mathbf{U}_1^{-1} = \frac{1}{\mathbf{s}_0^\mathsf{T}\mathbf{s}_0}, \quad \tilde{\mathbf{L}}_1 = \begin{bmatrix}0\end{bmatrix}, \quad \tilde{\mathbf{D}}_1 = \mathbf{s}_0^\mathsf{T}\mathbf{y_0}.$$

Together with the observation that

$$\begin{bmatrix}0 & \mathbf{s}_i^\mathsf{T}\mathbf{s}_i \\ \mathbf{s}_i^\mathsf{T}\mathbf{s}_i & \mathbf{s}_i^\mathsf{T}\mathbf{y}_i\end{bmatrix}^{-1} = \begin{bmatrix}-\frac{\mathbf{s}_i^\mathsf{T}\mathbf{y}_i}{(\mathbf{s}_i^\mathsf{T}\mathbf{s}_i)^2} & \frac{1}{\mathbf{s}_i^\mathsf{T}\mathbf{s}_i} \\ \frac{1}{\mathbf{s}_i^\mathsf{T}\mathbf{s}_i} & 0\end{bmatrix}, \quad (34)$$

an elementary calculation shows that (33) holds for $i = 1$. Suppose the statement is true for some $i$. Using the induction hypothesis together with (34), a simple calculation shows that

$$\begin{aligned}\mathbf{N}_{i+1} &= \mathbf{V}_i\mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}}\mathbf{Y}_i^{\mathsf{T}}\mathbf{V}_i + \mathbf{V}_i\mathbf{Y}_i\mathbf{U}_i^{-1}\mathbf{S}_i^{\mathsf{T}}\mathbf{V}_i - \mathbf{V}_i\mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}}\big(\tilde{\mathbf{L}}_i + \tilde{\mathbf{D}}_i + \tilde{\mathbf{L}}_i^{\mathsf{T}}\big)\mathbf{U}_i^{-1}\mathbf{S}_i^{\mathsf{T}}\mathbf{V}_i \\ &\quad -\frac{\mathbf{s}_i^{\mathsf{T}}\mathbf{y}_i}{(\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i)^2}\mathbf{s}_i\mathbf{s}_i^{\mathsf{T}} + \frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\mathbf{s}_i\mathbf{y}_i^{\mathsf{T}} + \frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\mathbf{y}_i\mathbf{s}_i^{\mathsf{T}}.\end{aligned}$$

On the other hand, let us calculate the expression (33) for $i + 1$. Based on the partitions

$$\begin{aligned}&\mathbf{S}_{i+1} = \begin{bmatrix}\mathbf{S}_i & \mathbf{s}_i\end{bmatrix} \\ &\mathbf{Y}_{i+1} = \begin{bmatrix}\mathbf{Y}_i & \mathbf{y}_i\end{bmatrix} \\ &\mathbf{U}_{i+1} = \begin{bmatrix}\mathbf{U}_i & \mathbf{S}_i^{\mathsf{T}}\mathbf{s}_i \\ 0 & \mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i\end{bmatrix} \quad\Longrightarrow\quad \mathbf{U}_{i+1}^{-1} = \begin{bmatrix}\mathbf{U}_i^{-1} & -\frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\mathbf{U}_i^{-1}\mathbf{S}_i^{\mathsf{T}}\mathbf{s}_i \\ 0 & \frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\end{bmatrix}, \\ &\tilde{\mathbf{L}}_{i+1} = \begin{bmatrix}\tilde{\mathbf{L}}_i & 0 \\ \mathbf{s}_i^{\mathsf{T}}\mathbf{Y}_i & 0\end{bmatrix}, \\ &\tilde{\mathbf{D}}_{i+1} = \begin{bmatrix}\tilde{\mathbf{D}}_i & 0 \\ 0 & \mathbf{s}_i^{\mathsf{T}}\mathbf{y}_i\end{bmatrix},\end{aligned}$$

we obtain

$$\begin{aligned}\mathbf{S}_{i+1}\mathbf{U}_{i+1}^{-\mathsf{T}} &= \begin{bmatrix}\mathbf{V}_i\mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}} & \frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\mathbf{s}_i\end{bmatrix}, \\ \mathbf{S}_{i+1}\mathbf{U}_{i+1}^{-\mathsf{T}}\mathbf{Y}_{i+1}^{\mathsf{T}} &= \mathbf{V}_i\mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}}\mathbf{Y}_i^{\mathsf{T}} + \frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\mathbf{s}_i\mathbf{y}_i^{\mathsf{T}}, \\ \tilde{\mathbf{L}}_{i+1} + \tilde{\mathbf{D}}_{i+1} + \tilde{\mathbf{L}}_{i+1}^{\mathsf{T}} &= \begin{bmatrix}\tilde{\mathbf{L}}_i + \tilde{\mathbf{D}}_i + \tilde{\mathbf{L}}_i^{\mathsf{T}} & \mathbf{Y}_i^{\mathsf{T}}\mathbf{s}_i \\ \mathbf{s}_i^{\mathsf{T}}\mathbf{Y}_i & \mathbf{s}_i^{\mathsf{T}}\mathbf{y}_i\end{bmatrix},\end{aligned}$$

hence

$$\begin{aligned}&\mathbf{S}_{i+1}\mathbf{U}_{i+1}^{-\mathsf{T}}\big(\tilde{\mathbf{L}}_{i+1} + \tilde{\mathbf{D}}_{i+1} + \tilde{\mathbf{L}}_{i+1}^{\mathsf{T}}\big)\mathbf{U}_{i+1}^{-1}\mathbf{S}_{i+1}^{\mathsf{T}} \\ &\quad= \mathbf{V}_i\mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}}\big(\tilde{\mathbf{L}}_i + \tilde{\mathbf{D}}_i + \tilde{\mathbf{L}}_i^{\mathsf{T}}\big)\mathbf{U}_i^{-1}\mathbf{S}_i^{\mathsf{T}}\mathbf{V}_i + \frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\mathbf{V}_i\mathbf{S}_i\mathbf{U}_i^{-\mathsf{T}}\mathbf{Y}_i^{\mathsf{T}}\mathbf{s}_i\mathbf{s}_i^{\mathsf{T}} \\ &\qquad+\frac{1}{\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i}\mathbf{s}_i\mathbf{s}_i^{\mathsf{T}}\mathbf{Y}_i\mathbf{U}_i^{-1}\mathbf{S}_i^{\mathsf{T}}\mathbf{V}_i + \frac{\mathbf{s}_i^{\mathsf{T}}\mathbf{y}_i}{(\mathbf{s}_i^{\mathsf{T}}\mathbf{s}_i)^2}\mathbf{s}_i\mathbf{s}_i^{\mathsf{T}}.\end{aligned}$$

Using these expressions and expanding (33) with $i$ replaced by $i+1$, and taking into account once again the definition of $\mathbf{V}_i$, an elementary calculation shows that the resulting matrix $\mathbf{N}_{i+1}$ coincides with the one obtained previously. This completes the induction. □

If $\mathbf{B}_{0,k} = \gamma_k\mathbf{I}$ for some $\gamma_k \in \mathbb{R}$, then (32) can be rewritten as

$$\mathbf{B}_k = \gamma_k\mathbf{I} + \mathbf{A}_k\begin{bmatrix}0 & \mathbf{U}_k \\ \mathbf{U}_k^{\mathsf{T}} & \mathbf{D}(\mathbf{S}_k^{\mathsf{T}}\mathbf{Y}_k) + \gamma_k\mathbf{D}(\mathbf{S}_k^{\mathsf{T}}\mathbf{S}_k) + \mathbf{L}(\mathbf{S}_k^{\mathsf{T}}\mathbf{Y}_k) + \mathbf{L}(\mathbf{S}_k^{\mathsf{T}}\mathbf{Y}_k)^{\mathsf{T}}\end{bmatrix}^{-1}\mathbf{A}_k^{\mathsf{T}}, \qquad (35)$$

where $\mathbf{A}_k = [\mathbf{S}_k, \mathbf{Y}_k]$ as before. This form of $\mathbf{B}_k$ has the advantage that all involved quantities can be obtained as submatrices of the product $\mathbf{A}_k^{\mathsf{T}}\mathbf{A}_k$.

**Updating and Complexity**

As before, the L-PSB quantities can be updated in a similar fashion to the L-BFGS case; see Section 4.1. We again maintain the quantities

$$\mathbf{S}_k^\mathsf{T}\mathbf{S}_k,\ \mathbf{S}_k^\mathsf{T}\mathbf{Y}_k,\ \mathbf{Y}_k^\mathsf{T}\mathbf{Y}_k \in \mathbb{R}^{m\times m} \quad \text{and} \quad \mathbf{S}_k^\mathsf{T}\mathbf{g}_k,\ \mathbf{Y}_k^\mathsf{T}\mathbf{g}_k \in \mathbb{R}^m. \tag{36}$$

These can be updated as before and used to compute the quasi-Newton direction via the inverse formula (17). The complexity of the L-PSB step equals that of L-BFGS.

# 5 Numerical Experiments

The benchmark implementation described here can be found online at `https://github.com/dmsteck/paper-regularized-qn-benchmark`.

## 5.1 Problem Set and Implementation

Algorithms were tested on 79 large-scale ($n \geq 1000$) problems from the `CUTEst` collection [12]. Examples where all tested methods failed were omitted. The implementation was done in `Python3` using the `PyCUTEst` interface [11]. We implemented the following algorithms:

**armijoLBFGS:** The conventional L-BFGS method with Armijo line search and the cautious updating scheme (24);

**wolfeLBFGS:** the Liu–Nocedal L-BFGS method [17] using Moré–Thuente line search [18] as implemented in `MINPACK`;

**regLBFGS:** Algorithm 2.1 using the L-BFGS technique as set out in Section 4.1;

**regLSR1:** Algorithm 2.1 using the L-SR1 technique as set out in Section 4.2;

**regLPSB:** Algorithm 2.1 using the L-PSB technique as set out in Section 4.3.

Comparable studies in other papers [2,22] indicate that regularized methods may benefit from a nonmonotonicity strategy. Therefore, and to obtain a larger dataset, we also implemented nonmonotone versions of all algorithms, where $M := 8$ was chosen as the nonmonotonicity offset; this was incorporated into the methods by replacing the reference value $f(\mathbf{x}_k)$ in the regularization control (9) and the line search routines by $\max_{0\leq i<M} f(\mathbf{x}_{k-i})$ for $k \geq M$. The initial steps $k = 0, \ldots, M-1$ were treated without modification.

The different realizations of Algorithm 2.1 were implemented with the hyperparameters

$$m = 5, \quad \mu_0 = 1, \quad p_{\min} = c_1 = 10^{-4}, \quad c_2 = 0.9, \quad \sigma_1 = 0.5, \quad \sigma_2 = 4. \tag{37}$$

The Armijo line search repeatedly halves the step size $t_k$ until

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + c_1 \mathbf{g}_k^\mathsf{T}\mathbf{d}_k, \tag{38}$$

where (in this context) $\mathbf{d}_k$ is the quasi-Newton step. The Moré–Thuente line search uses interpolation until (38) is satisfied and $|\nabla f(\mathbf{x}_k + t_k\mathbf{d}_k)^\mathsf{T}\mathbf{d_k}| \leq -0.5\mathbf{g}_k^\mathsf{T}\mathbf{d}_k$.

For the BFGS-type algorithms except wolfeLBFGS, the cautious updating scheme (24) is used with $\varepsilon := 10^{-8}$. The regLSR1 and regLPSB algorithms benefit from indefinite Hessian approximations and therefore were not combined with the cautious updating scheme.

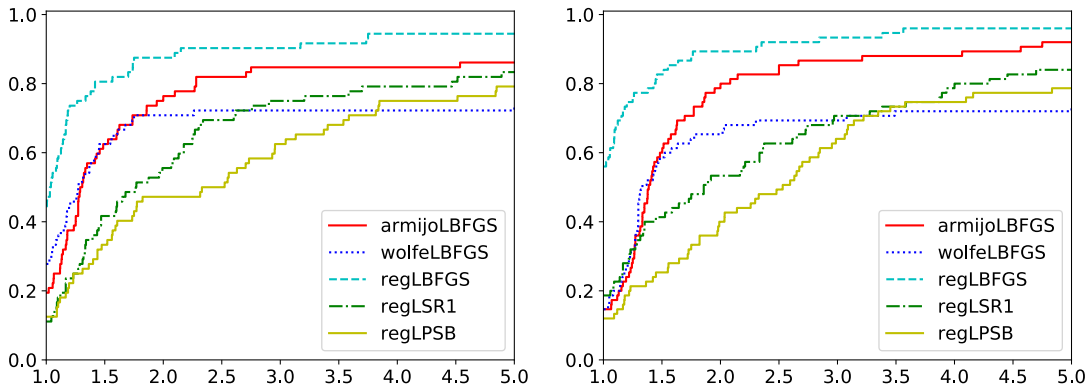


Figure 1: Performance profiles based on the number of function evaluations of the five algorithms from Section 5: monotone case (left), nonmonotone case (right).

However, for these methods, the cautious scheme was still applied to the update of the rolling initial approximation (39); see below.

The algorithms were terminated as soon as either

$$\|\mathbf{g}_k\|_\infty < 10^{-4}, \qquad k \geq 10^5, \qquad \text{or} \quad \begin{cases} \mu_k > 10^{15}, \\ t_k < 10^{-15}, \end{cases}$$

depending on whether the algorithm is of regularization or line search type.

The initial estimate $\mathbf{B}_{0,k}$ in step $k$ is defined by the standard formula

$$\mathbf{B}_{0,k} = \gamma_k \mathbf{I}, \qquad \gamma_k = \frac{\mathbf{y}_k^T \mathbf{y}_k}{\mathbf{y}_k^T \mathbf{s}_k}. \tag{39}$$

In addition, we adopted a lower threshold $\mu_{\min} := 10^{-4}$ for the regularization parameter. This improved the practical behavior of the method (particularly in the L-BFGS case) and also prevented the regularization parameter from becoming zero in limited-precision arithmetic.

It may seem that the above choices lead to a preference of high regularization parameters over low ones and could therefore impede fast asymptotic convergence. What we have found empirically is that Algorithm 2.1 (with L-BFGS) often behaves best when the regularization parameter is changed infrequently. This suggests that the parameter should be increased sharply when necessary (to avoid having to increase repeatedly), and only decreased when the step quality is very good. This is reflected in our choice of parameters.

Note also that limited memory methods rarely achieve actual superlinear convergence; the typical behavior is asymptotically linear [17], and a simple analysis of the Newton case suggests that a small but non-decaying value of $\mu_k$ will typically preserve linear convergence. This indicates that the choices made here are sound from a theoretical point of view.

The resulting data is presented in performance profiles [9], where the number of function evaluations was used as the base metric; see Figure 1. It may be interesting to also conduct an analysis of CPU times, but this would effectively require another programming language due to the lack of optimizing compilation in languages like `Python` or `MATLAB`, which incurs significant overhead on loops and repeated assignment operations. We anticipate that realistic CPU times would slightly benefit the line search L-BFGS methods due to the logistic effort associated with limited memory updating in the regularized methods (see Section 4.1).

### 5.2 Discussion of the Results

It is clear from Figure 1 that the regularization technique can substantially improve the efficiency and robustness of L-BFGS on large-scale nonlinear problems or when nonmonotone strategies are employed. An intuitive explanation of this phenomenon lies in the fact that regularization "stabilizes" the Hessian approximation in the sense that the condition number becomes smaller, which may make the method less susceptible to step jumps or "discontinuities" induced by nonmonotonicity or extreme nonlinearity.

The regularized variants of L-SR1 and L-PSB are competitive but fall short of the overall performance of the L-BFGS methods. The L-BFGS algorithm with Moré–Thuente line search [18] is competitive on the fastest problems. Similar to the results in [2], however, we found that this and similar Wolfe–Powell based line searches were significantly less efficient than their Armijo counterpart due to the excessive number of function evaluations.

An interesting observation we made during our testing is that L-SR1 and, in particular, L-PSB were actually more efficient when used with a more "optimistic" regularization scheme (i.e., lower regularization parameters). This is somewhat surprising because these methods generate indefinite Hessian approximations which should, intuitively, benefit the most from regularization; on the other hand, L-BFGS generates an approximation which is positive definite anyway, which suggests that regularization may be less necessary here. The numerical evidence we observed contradicts this intuition.

We can only give a partial explanation for this phenomenon. It is well-known that L-BFGS generalizes the classical conjugate gradient method, which suggests that L-BFGS imposes some kind of analytical relationship (a generalized "conjugacy") on successive search directions. This property may be preserved in a certain way when L-BFGS is used with a regularization parameter that changes infrequently. On the other hand, L-SR1 and L-PSB are generally acknowledged to generate more accurate approximations of the exact Hessian (especially when it is indefinite), which indicates that these methods behave more similarly to a conventional Newtonian algorithm and therefore benefit from a quicker reduction of regularization parameters.

**Remark 5.1 (Further improvements).** It is possible to implement further modifications and improvements into the regularized quasi-Newton scheme, but we have abstained from doing so in order to facilitate a fair comparison. For instance, it may be beneficial to update the quasi-Newton information in *rejected* steps since the trial function value and gradient provide meaningful information [22]. Note that this technique is covered by the framework of Algorithm 2.1 since we allow $\mathbf{B}_k$ to be chosen anew in each iteration.

**Remark 5.2 (Alternative methods).** We also implemented comparable versions of the regularized L-BFGS algorithms from [22] and [21], which differ from our regularized L-BFGS method in the way the Hessian approximation is formed; see Remark 4.2. These methods performed similarly (with a slight advantage for the method from [22]), but both were less efficient than the approach based on compact representations (Algorithm 2.1 and Section 4.1).

## 6 Final Remarks

The results and numerical evidence in this paper demonstrate conclusively that regularization is a powerful globalization technique for limited memory quasi-Newton methods. We hope that the findings presented here will facilitate more research into these techniques, for example,

on quantitative convergence results or on how to integrate regularization with BFGS in a full-memory context.

# References


[1] J. Brust, J. B. Erway, and R. F. Marcia. On solving L-SR1 trust-region subproblems. *Comput. Optim. Appl.*, 66(2):245–266, 2017.

[2] O. P. Burdakov, L. Gong, S. Zikrin, and Y.-x. Yuan. On efficiently combining limited-memory and trust-region techniques. *Math. Program. Comput.*, 9(1):101–134, 2017.

[3] O. P. Burdakov, J. M. Martínez, and E. A. Pilotta. A limited-memory multipoint symmetric secant method for bound constrained optimization. *Ann. Oper. Res.*, 117:51–70, 2002. Operations research and systems (CLAIO 2000), Part II (Mexico City).

[4] J. V. Burke, A. Wiegmann, and L. Xu. Limited memory BFGS updating in a trust–region framework. Tech. rep., University of Washington, 2008.

[5] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Programming*, 63(2, Ser. A):129–156, 1994.

[6] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-region methods*. MPS/SIAM Ser. Optim. SIAM, Philadelphia, 2000.

[7] O. DeGuchy, J. B. Erway, and R. F. Marcia. Compact representation of the full Broyden class of quasi-Newton updates. *Numer. Linear Algebra Appl.*, 25(5):e2186, 15, 2018.

[8] J. E. Dennis, Jr. and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19(1):46–89, 1977.

[9] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2, Ser. A):201–213, 2002.

[10] J. B. Erway, V. Jain, and R. F. Marcia. Shifted L-BFGS systems. *Optim. Methods Softw.*, 29(5):992–1004, 2014.

[11] J. Fowkes and L. Roberts. PyCUTEst: Python interface to the CUTEst optimization test environment. `https://jfowkes.github.io/pycutest`. Accessed May 2019.

[12] N. I. M. Gould, D. Orban, and P. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.*, 60(3):545–557, 2015.

[13] D.-H. Li and M. Fukushima. A modified BFGS method and its global convergence in nonconvex minimization. *J. Comput. Appl. Math.*, 129(1-2):15–35, 2001.

[14] D.-H. Li and M. Fukushima. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM J. Optim.*, 11(4):1054–1064, 2001.

[15] D.-H. Li, M. Fukushima, L. Qi, and N. Yamashita. Regularized Newton methods for convex minimization problems with singular solutions. *Comput. Optim. Appl.*, 28(2):131–147, 2004.

[16] C. Liu and S. A. Vander Wiel. Statistical quasi-Newton: a new look at least change. *SIAM J. Optim.*, 18(4):1266–1285, 2007.

[17] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.

[18] J. J. Moré and D. J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software*, 20(3):286–307, 1994.

[19] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comp.*, 35(151):773–782, 1980.

[20] M. J. D. Powell. A new algorithm for unconstrained optimization. In *Nonlinear Programming (Proc. Sympos., Univ. of Wisconsin, Madison, Wis., 1970)*, pages 31–65. Academic Press, New York, 1970.

[21] N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pages 436–443, 2007.

[22] S. Sugimoto and N. Yamashita. A regularized limited-memory BFGS method for unconstrained minimization problems. Technical report 2014–001, Department of Applied Mathematics and Physics, Kyoto University, August 2014.

[23] K. Ueda and N. Yamashita. Convergence properties of the regularized Newton method for the unconstrained nonconvex optimization. *Appl. Math. Optim.*, 62(1):27–46, 2010.

[24] K. Ueda and N. Yamashita. A regularized Newton method without line search for unconstrained optimization. *Comput. Optim. Appl.*, 59(1-2):321–351, 2014.

[25] Z. Wei, G. Li, and L. Qi. New quasi-Newton methods for unconstrained optimization problems. *Appl. Math. Comput.*, 175(2):1156–1188, 2006.

[26] H. Zhang and Q. Ni. A new regularized quasi-Newton method for unconstrained optimization. *Optim. Lett.*, 12(7):1639–1658, 2018.