On Recent Developments in BFGS Methods for Unconstrained Optimization

Philip E. Gill[*] Jeb H. Runnoe[*]

UCSD Center for Computational Mathematics
Technical Report CCoM-22-4
July 1, 2022, Revised October 17, 2023

Abstract

Quasi-Newton methods form the basis of many effective methods for unconstrained and constrained optimization. As the iterations proceed, a quasi-Newton method incorporates new curvature information by performing a low-rank update to a matrix that serves as an approximation to a Hessian matrix of second derivatives. In the years following the publication of the Davidon-Fletcher-Powell (DFP) method in 1963 the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update emerged as the best update formula for use in unconstrained minimization. More recently, a number of quasi-Newton methods have been proposed that are intended to improve on the efficiency and reliability of the BFGS method. Unfortunately, there is no known analytical means of determining the relative performance of these methods on a general nonlinear function, and there is a real need for extensive experimental testing to justify the theoretical basis of each approach. The goal of this report is to implement and test these methods in a uniform, systematic, and consistent way. In the first part of the report, we review several quasi-Newton methods, discuss their relative benefits, and discuss their implementation. In the second part, we investigate more recent variations, explain their motivation and theory, and investigate their performance.

Key words. Unconstrained optimization, line-search methods, quasi-Newton methods, BFGS method, self-scaled quasi-Newton methods.

AMS subject classifications. 49M37, 65K05, 90C30, 90C53

[*]Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (`pgill@ucsd.edu`, `jrunnoe@ucsd.edu`) Research supported in part by National Science Foundation grants DMS-0915220, DMS-1318480 and Research Training Grant DMS-1345013. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

1. Introduction

This report concerns the formulation and evaluation of quasi-Newton methods for finding a local solution of the unconstrained optimization problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x), \tag{UC}$$

where f is a twice-continuously differentiable scalar-valued function. All the methods to be considered are line-search methods, which generate a sequence $\{x_k\}$ with $x_{k+1} = x_k + \alpha_k p_k$, where α_k is a positive scalar and p_k is a descent direction for f at x_k. The principal feature of a quasi-Newton method is that p_k is the minimizer of the quadratic model

$$q_k(p) = f(x_k) + p^{\mathrm{T}}\nabla f(x_k) + \tfrac{1}{2}p^{\mathrm{T}}H_k p, \tag{1.1}$$

with H_k an approximation of the Hessian matrix $\nabla^2 f(x_k)$. The approximate Hessian represents (in some sense) curvature information that has been accumulated at iterates preceding x_k. The first quasi-Newton method for finding a solution of problem UC was proposed in 1959 by Davidon [5]. This method was modified and clarified by Fletcher and Powell [7], and forms the basis for the Davidon-Fletcher-Powell (DFP) method.

The move from x_k to x_{k+1} provides further information about the curvature that may be incorporated in a new Hessian approximation H_{k+1}. If $d_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, then $y_k^{\mathrm{T}} d_k$ is a first-order estimate of the curvature $d_k^{\mathrm{T}}\nabla^2 f(x_k) d_k$ and is known as the approximate curvature. The matrix H_{k+1} is chosen to satisfy the so-called quasi-Newton condition:

$$H_{k+1}d_k = y_k. \tag{1.2}$$

This condition forces the new quadratic model q_{k+1} to have curvature $y_k^{\mathrm{T}} d_k$ along d_k; i.e.,

$$d_k^{\mathrm{T}} H_{k+1} d_k = y_k^{\mathrm{T}} d_k. \tag{1.3}$$

We say that the quasi-Newton condition *installs* the approximate curvature $y_k^{\mathrm{T}} d_k$ as the *exact* curvature of the new quadratic model $q_{k+1}(p)$. After the move from x_k to x_{k+1}, a quasi-Newton method computes $H_{k+1} = H_k + U_k$, where U_k is a low-rank update matrix that is designed to retain some important properties of H_k such as symmetry or positive definiteness. An update formula $H_{k+1} = H_k + U_k$ is said to have the property of *hereditary symmetry* if symmetry of H_k implies symmetry of H_{k+1}, and *hereditary positive-definiteness* if the positive-definiteness of H_{k+1} follows from that of H_k.

If H_k is positive definite, the minimizer of the quadratic model (1.1) is the unique solution of the equations $H_k p_k = -\nabla f(x_k)$. The next iterate $x_{k+1} = x_k + \alpha_k p_k$ is chosen to give a decrease in f that is at least as good as a fixed fraction η_A $(0 < \eta_A < 1)$ of the decrease in the local affine model $f(x_k) + \nabla f(x_k)^{\mathrm{T}}(x - x_k)$. The sufficient-decrease condition may be written as

$$f(x_k + \alpha_k p_k) \le f(x_k) + \alpha_k \eta_A \nabla f(x_k)^{\mathrm{T}} p_k,$$

(see, e.g., Armijo [2], Ortega and Rheinboldt [22]). Every method considered in this report satisfies the Armijo condition in conjunction with the strong Wolfe condition

$$|\nabla f(x_k + \alpha_k p_k)^{\mathrm{T}} p_k| \le \eta_W |\nabla f(x_k)^{\mathrm{T}} p_k|, \tag{1.4}$$

where η_W is a preassigned scalar such that $\eta_W \in (\eta_A, 1)$. (See, e.g., Wolfe [27], Moré and Thuente [15], and Gill et al. [9]).

The quasi-Newton condition (1.2) and identity (1.3) imply that $y_k^{\mathrm{T}} d_k > 0$ is a necessary condition for the approximate Hessian H_{k+1} to be positive definite. This condition is satisfied for any x_{k+1} such that α_k satisfies the Wolfe condition (1.4). In particular, for any step satisfying (1.4) it holds that

$$\begin{aligned} y_k^{\mathrm{T}} p_k &= \nabla f(x_{k+1})^{\mathrm{T}} p_k - \nabla f(x_k)^{\mathrm{T}} p_k \\ &\ge \eta_W \nabla f(x_k)^{\mathrm{T}} p_k - \nabla f(x_k)^{\mathrm{T}} p_k \ge (1-\eta_W) p_k^{\mathrm{T}} H_k p_k. \end{aligned}$$

This implies that any α_k satisfying the Wolfe condition (1.4) will give an x_{k+1} such that $y_k^{\mathrm{T}} d_k \ge (1-\eta_W)\alpha_k p_k^{\mathrm{T}} H_k p_k > 0$.

Among the many quasi-Newton updates developed in the ten years following the publication of the DFP method, the *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* update emerged as the best update formula for use in unconstrained minimization. The BFGS update has the form

$$H_{k+1} = H_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k d_k^{\mathrm{T}} H_k + \frac{1}{y_k^{\mathrm{T}} d_k} y_k y_k^{\mathrm{T}}, \tag{1.5}$$

which represents a symmetric, rank-two modification to H_k. The BFGS update has the property of hereditary positive definiteness if $y_k^{\mathrm{T}} d_k > 0$. A line-search method with the BFGS update (1.5) is referred to as Algorithm `bfgsH`.

An alternative to solving the equations $H_k p_k = -\nabla f(x_k)$ at each step is to update the inverse approximate Hessian M_k and form the product $p_k = -M_k \nabla f(x_k)$. Every update to H_k is associated with an equivalent update to H_k^{-1}, which may be derived using the Sherman-Morrison-Woodbury formula (see Sherman and Morrison [26], and Woodbury [28]). If H_k is updated by the BFGS formula (1.5), the associated update to the inverse is given by

$$M_{k+1} = M_k - \frac{1}{y_k^{\mathrm{T}} d_k}(M_k y_k d_k^{\mathrm{T}} + d_k y_k^{\mathrm{T}} M_k) + \frac{y_k^{\mathrm{T}} d_k + y_k^{\mathrm{T}} M_k y_k}{(y_k^{\mathrm{T}} d_k)^2} y_k y_k^{\mathrm{T}}. \tag{1.6}$$

The BFGS update to the inverse may be written in a number of different ways. If the cross-product term of (1.6) is symmetrized, we obtain the update

$$\begin{aligned} M_{k+1} = M_k &- \frac{1}{y_k^{\mathrm{T}} M_k y_k} M_k y_k y_k^{\mathrm{T}} M_k + \frac{1}{y_k^{\mathrm{T}} d_k} d_k d_k^{\mathrm{T}} \\ &+ (y_k^{\mathrm{T}} M_k y_k)\Big(\frac{1}{y_k^{\mathrm{T}} d_k} d_k - \frac{1}{y_k^{\mathrm{T}} M_k y_k} M_k y_k\Big)\Big(\frac{1}{y_k^{\mathrm{T}} d_k} d_k - \frac{1}{y_k^{\mathrm{T}} M_k y_k} M_k y_k\Big)^{\mathrm{T}}. \end{aligned} \tag{1.7}$$

A line-search method with the BFGS update (1.6) or (1.7) is referred to as Algorithm `bfgsM`.

If x_k is not close to a minimizer, the approximate curvature $y_k^{\mathrm{T}} d_k$ may not be positive and H_{k+1} may be indefinite or undefined. In unconstrained optimization, most implementations skip the update if $y_k^{\mathrm{T}} d_k$ is negative, in which case no new information about the curvature is used at that iteration. An alternative strategy is discussed in Section 4.

1.1. Overview

While the BFGS update is widely perceived as the basis for the best methods proposed in the ten years following the formulation of the first quasi-Newton method, the reason for this practical superiority is not fully understood. Over the past 60 years there have been a number of variants of the BFGS method that are intended to enhance those theoretical aspects of the method that are perceived to be the main contributing factors to its practical effectiveness. These modifications are motivated by various assumptions. For example, the use of more accurate curvature information will yield better search directions; or reducing the condition number of the approximate Hessian will provide better numerical stability.

The methods considered in this report are all variations of the BFGS algorithm, and are based on either the standard update (1.5) or the inverse update (1.6) formulas. These variants fall under three broad categories: methods that modify the condition number of the approximate Hessian; methods that focus on improving the numerical accuracy of the update; and methods that use a modified value of y_k in the quasi-Newton condition intended to improve the accuracy of the approximate Hessian.

The difficulty of judging the impact of the many theoretical developments of quasi-Newton methods emphasizes the importance of practical testing to support claims of improved reliability or convergence. Today there are curated collections of test problems that were not available when many of these algorithms were proposed. In this report, we use the Constrained and Unconstrained Testing Environment with safe threads (CUTEst) to compare the performance of these algorithms (see Gould, Orban & Toint [11]).

1.2. Notation

Given vectors x and y, the vector consisting of x augmented by y is denoted by (x, y). The subscript i is appended to vectors to denote the ith component of that vector, whereas the subscript k is appended to a vector to denote its value during the kth iteration of an algorithm, e.g., x_k represents the value for x during the kth iteration, whereas $[\, x_k \,]_i$ denotes the ith component of the vector x_k. The vector e denotes the column vector of ones, and I denotes the identity matrix. The dimensions of e and I are defined by the context. The vector two-norm or its induced matrix norm are denoted by $\|\cdot\|$. The vector p_k is used the search direction, α_k the scalar step length $d_k = \alpha_k p_k = x_{k+1} - x_k$ the step (change in variables) $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ is the change in gradient H_k is the approximate Hessian M_k is the approximate inverse Hessian.

2. Self-Scaled Updates

An important factor in the numerical performance of quasi-Newton methods is the conditioning of each approximate Hessian H_k (i.e., the magnitude of the matrix condition number of H_k). This conditioning is influenced by a number of factors, including the step length α_k, the initial approximate Hessian H_0 and the condition of the exact Hessian at a solution. Self-scaled quasi-Newton methods are designed to limit the bad effects of the choice of the initial approximate Hessian H_0 on the efficiency of a method. Self-scaled updates were first proposed by Oren and Luenberger [20]. Additional references include Oren [17–19], and Oren and Spedicato [21],

2.1. Scaling the BFGS Hessian: quadratic case.

The influence of H_0 may be seen by considering the BFGS method applied to the strictly convex quadratic $f(x) = c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Hx$. If an exact line search is used, then the conjugate-direction property of the approximate Hessian implies that $H_k d_j = Hd_j$, for $0 \le j < k$. As H is nonsingular, it follows that $H^{-1}H_k d_j = d_j$, which implies that $H^{-1}H_k$ has k unit eigenvalues corresponding to the k eigenvectors $\{\, d_j \,\}$, $0 \le j < k$. This property means that the BFGS method moves the eigenvalues of $H^{-1}H_0$ to unity, one at a time as the iterations proceed.

If H_0 is chosen so that the eigenvalues of $H^{-1}H_0$ are all *large* relative to unity, then the first update will make $H^{-1}H_1$ *ill-conditioned.* It follows that any quasi-Newton method is affected by the *spread* of the spectrum of each $H^{-1}H_k$, which is the length of the interval with end-points given by the smallest and largest eigenvalue of $H^{-1}H_k$. The idea is to scale each H_k before applying the quasi-Newton update. This scaling is chosen to make the spread of the spectrum of $H^{-1}H_{k+1}$ no worse than that of $H^{-1}H_k$.

The next result considers the properties of the eigenvalues of $H^{-1}H_k$ and $H^{-1}H_{k+1}$. As H is positive definite, we can write

$$H^{-1}H_j = H^{-1/2}\big(H^{-1/2}H_jH^{-1/2}\big)H^{1/2}, \tag{2.1}$$

which implies that the eigenvalues of $H^{-1}H_j$ are real and positive for all $0 \le j \le k$.

Result 2.1. *Consider the application of the BFGS method to a quadratic with positive-definite Hessian H. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ denote the eigenvalues of $H^{-1}H_k$ ordered so that*

$$0 < \lambda_n \le \lambda_{n-1} \le \cdots \le \lambda_1.$$

Then, if $1 \in [\lambda_n, \lambda_1]$, the eigenvalues of $H^{-1}H_{k+1}$ are all contained in $[\lambda_n, \lambda_1]$.

Proof. Let Q_k denote the matrix $Q_k = H^{-1/2}H_kH^{-1/2}$, which has positive eigenvalues from Sylvester's law of inertia. The identity (2.1) implies that Q_k and $H^{-1}H_k$ have the same eigenvalues. For a quadratic f, the gradient difference satisfies $y_k = Hd_k$, and the updated BFGS matrix (1.5) may be written as an update to the matrix Q_k, i.e.,

$$Q_{k+1} = Q_k - \frac{1}{q_k^{\mathrm{T}}Q_kq_k}Q_kq_kq_k^{\mathrm{T}}Q_k + \frac{1}{q_k^{\mathrm{T}}q_k}q_kq_k^{\mathrm{T}} = P_k + \frac{1}{q_k^{\mathrm{T}}q_k}q_kq_k^{\mathrm{T}},$$

where

$$Q_{k+1} = H^{-1/2} H_{k+1} H^{-1/2}, \quad q_k = H^{1/2} d_k \quad \text{and} \quad P_k = Q_k - \frac{1}{q_k^{\mathrm{T}} Q_k q_k} Q_k q_k q_k^{\mathrm{T}} Q_k.$$

Direct multiplication gives $P_k q_k = 0$ and q_k is an eigenvector of P_k with zero eigenvalue. If the eigenvalues of P_k are denoted by μ_1, μ_2, …, μ_n, then

$$0 = \mu_n \le \mu_{n-1} \le \cdots \le \mu_1,$$

and the eigenvalue interlacing theorem gives

$$0 = \mu_n \le \lambda_n \le \mu_{n-1} \le \cdots \le \mu_1 \le \lambda_1.$$

From the definition of Q_{k+1}, it holds that $Q_{k+1} q_k = q_k$, so that Q_{k+1} has one eigenvector q_k, and $n-1$ eigenvectors that are orthogonal to q_k. It follows that the spectrum of Q_{k+1} consists of a unit eigenvalue and the $n-1$ eigenvalues of P_k. This implies that the eigenvalues of Q_{k+1} are 1, and μ_{n-1}, …, μ_1. But $1 \in [\lambda_n, \lambda_1]$, and $\lambda_n \le \mu_{n-1} \le \cdots \le \mu_1 \le \lambda_1$, which implies that 1, μ_{n-1}, …, $\mu_1 \in [\lambda_n, \lambda_1]$, as required. ∎

As $H^{-1} H_k$ and $H^{-1} H_{k+1}$ have k and $k+1$ unit eigenvalues, respectively, the assumptions for this result hold for all $k > 0$. A simple inductive argument shows that the bound on the eigenvalue ratio of $H^{-1} H_k$ is determined by the eigenvalues λ_1, λ_2, …, λ_n of $H^{-1} H_0$. If H_0 is chosen so that this eigenvalue ratio is small; and $1 \in [\lambda_n, \lambda_1]$, then the ratios for $H^{-1} H_k$ will be small for all subsequent steps.

An appropriate choice of H_0 may be achieved as follows. First, an update pair $(d_1,\ y_1)$ is computed using a given initial H_0. However, before H_0 is updated it is rescaled as $H_0 \leftarrow S_0 H_0$, where S_0 is a positive-definite scaling matrix chosen so that the eigenvalues λ_1, λ_2, …, λ_n of $H^{-1}(S_0 H_0)$ satisfy $1 \in [\lambda_n, \lambda_1]$ and λ_1/λ_n is "small". The optimal choice of S_0 is obviously $S_0 = H$, which is unknown. However, practical rescaling techniques are based on a simple diagonal scaling of the form $S_0 = \gamma I$ for some $\gamma > 0$. If the eigenvalues of $H^{-1} H_0$ are $0 < \lambda_n \le \lambda_{n-1} \le \cdots \le \lambda_1$ and γ is any value such that $1/\lambda_1 \le \gamma \le 1/\lambda_n$, then the eigenvalues of $H^{-1}(\gamma H_0)$ are $0 < \gamma\lambda_n \le \gamma\lambda_{n-1} \le \cdots \le \gamma\lambda_1$, with

$$0 < \gamma\lambda_n \le 1 \le \gamma\lambda_1 \quad \text{and} \quad 1 \in [\gamma\lambda_n, \gamma\lambda_1].$$

These results are based on f being quadratic. In the next section we consider rescaling techniques for the general nonlinear case.

2.2. The self-scaled explicit BFGS method (`bfgsHS`).

For a general nonlinear f, scaling the initial approximate Hessian is not sufficient to provide a favorable eigenvalue distribution in all the subsequent matrices H_k. Instead, it is necessary to rescale every H_k so that H_{k+1} has a favorable *approximate* eigenvalue ratio. The idea is to scale H_k by a scalar γ_k so that $1 \in [\lambda_n, \lambda_1]$, i.e., the spread of the spectrum of $\nabla^2 f(x_k)^{-1}(\gamma_k H_k)$ includes 1 to first order.

If H_k is replaced by $\gamma_k H_k$ in the BFGS update (1.5) we obtain

$$H_{k+1} = \gamma_k \Big(H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k \Big) + \frac{1}{y_k^T d_k} y_k y_k^T. \tag{2.2}$$

A suitable value of γ_k may be derived from the choice of scaling for H_k in the quadratic case. In this case, if the eigenvalues of $H^{-1}H_k$ are $0 < \lambda_n \le \lambda_{n-1} \le \cdots \le \lambda_1$, then any γ_k such that $1/\lambda_1 \le \gamma \le 1/\lambda_n$ will give

$$0 < \gamma_k \lambda_n \le 1 \le \gamma_k \lambda_1 \quad \text{and} \quad 1 \in [\gamma_k \lambda_n, \gamma_k \lambda_1]. \tag{2.3}$$

Recall that, for a quadratic, we can define

$$Q_k = H^{-1/2} H_k H^{-1/2}, \qquad q_k = H^{1/2} d_k,$$

in which case $d_k^T H_k d_k = d_k^T H^{1/2} Q_k H^{1/2} d_k = q_k^T Q_k q_k$ and $y_k^T d_k = d_k^T H d_k = q_k^T q_k$. This implies that $d_k^T H_k d_k / y_k^T d_k = q_k^T Q_k q_k / q_k^T q_k$, giving the bounded Rayleigh quotient

$$\lambda_n \le \frac{d_k^T H_k d_k}{y_k^T d_k} \le \lambda_1 \quad \text{or, equivalently,} \quad \frac{1}{\lambda_1} \le \frac{y_k^T d_k}{d_k^T H_k d_k} \le \frac{1}{\lambda_n}.$$

It follows that in the nonlinear case, if we choose $\gamma_k = y_k^T d_k / d_k^T H_k d_k$ in (2.2) then (2.3) will hold to first order. Note that $d_k^T(\gamma_k H_k) d_k = y_k^T d_k$, which implies that the scaling installs the approximate curvature $y_k^T d_k$ *before* the update.

The equivalent update for the inverse is

$$\begin{aligned} M_{k+1} = {} & \gamma_k \left(M_k - \frac{1}{y_k^T M_k y_k} M_k y_k y_k^T M_k \right) + \frac{1}{y_k^T d_k} d_k d_k^T \\ & + \gamma_k \left(y_k^T M_k y_k \right) \Big(\frac{1}{y_k^T d_k} d_k - \frac{1}{y_k^T M_k y_k} M_k y_k \Big) \Big(\frac{1}{y_k^T d_k} d_k - \frac{1}{y_k^T M_k y_k} M_k y_k \Big)^T. \end{aligned} \tag{2.4}$$

with γ_k chosen as $\gamma_k = y_k^T d_k / y_k^T M_k y_k$.

Line-search methods based on the scaled BFGS update (2.2) and its inverse (2.4) will be referred to as Algorithms `bfgsHS` and `bfgsMS`, respectively.

3. Factored BFGS Methods

3.1. The factored Hessian BFGS method (`bfgsR`)

In this section we derive a variant of the BFGS method that maintains a Cholesky factorization of the approximate Hessian $H_k = R_k^T R_k$, where R_k is upper triangular with positive diagonals. The numerical results of Section 5 indicate that updating the Cholesky factor is substantially more reliable than maintaining the inverse approximation $M_k \approx \nabla^2 f(x_k)^{-1}$. Though it is generally regarded as more computationally expensive, a technique is described at the end of this section that increases efficiency, making the operation-complexity comparable to that of using the inverse approximation without sacrificing any of the reliability.

A significant drawback to the explicit and inverse BFGS methods is the inability to efficiently monitor the conditioning or positive definiteness of their respective approximations. The theoretical property of hereditary positive definiteness does not always hold in finite-precision arithmetic and the approximations may become indefinite, in which case the search directions obtained from forming $p_k = -M_k\nabla f(x_k)$ or solving $H_k p_k = -\nabla f(x_k)$ may not even be descent directions, ultimately leading to line search failure.

If the approximations become ill-conditioned or indefinite they can be reset. Unfortunately, it is difficult to determine when this happens. In general it is too expensive to compute the singular values of H_k or M_k. However, if a method based upon updating a factorization is being used, only the factor R_k is stored and updated, which implies that $R_k^{\mathrm{T}} R_k$ can never be indefinite. Moreover, a lower bound on the condition number of H_k is readily available from the identity $\mathrm{cond}(H_k) = \mathrm{cond}(R_k)^2$ and the bound $\mathrm{cond}(R_k) \ge \max|r_{jj}|/\min|r_{jj}|$. It follows that one possible strategy is to reset the (implicit) approximate Hessian H_k if $\mathrm{cond}(R_k)^2$ becomes large.

Although the details of updating matrix factorizations can be tedious, the procedures themselves are central in the formulation of modern numerically stable optimization methods. Given the Cholesky factorization $H_k = R_k^{\mathrm{T}} R_k$, the equations $H_k p_k = -\nabla f(x_k)$ may be written as

$$R_k^{\mathrm{T}} R_k p_k = -\nabla f(x_k).$$

To obtain the solution p_k, first solve the lower-triangular system $R_k^{\mathrm{T}} q_k = -\nabla f(x_k)$ followed by the upper-triangular system $R_k p_k = q_k$.

Updating the Cholesky factorization. Suppose that the Cholesky factor R_k of the current matrix H_k is available, so that $H_k = R_k^{\mathrm{T}} R_k$, where R_k is a nonsingular upper triangle. Given that

$$H_{k+1} = H_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k d_k^{\mathrm{T}} H_k + \frac{1}{y_k^{\mathrm{T}} d_k} y_k y_k^{\mathrm{T}},$$

we seek an upper-triangular matrix R_{k+1} such that $H_{k+1} = R_{k+1}^{\mathrm{T}} R_{k+1}$. This matrix may be found by utilizing an alternative form for the BFGS update (1.5). Brodlie, Gourlay and Greenstadt [3] show that

$$\begin{aligned} &H_{k+1} = (I + v_k d_k^{\mathrm{T}}) H_k (I + d_k v_k^{\mathrm{T}}), \quad \text{with} \\ &\qquad v_k = \pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} y_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k. \end{aligned} \tag{3.1}$$

If the factorization $H_k = R_k^{\mathrm{T}} R_k$ is known at the kth iteration then

$$H_{k+1} = (I - v_k d_k^{\mathrm{T}}) R_k^{\mathrm{T}} R_k (I - d_k v_k^{\mathrm{T}}) = \widehat{R}_k^{\mathrm{T}} \widehat{R}_k,$$

with $\widehat{R}_k = R_k(I - d_k v_k^{\mathrm{T}})$. Substituting the expression for v_k and expanding gives

$$\widehat{R}_k = R_k + \frac{1}{(d_k^{\mathrm{T}} H_k d_k)^{1/2}} R_k d_k \left(\pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2}} y_k - \frac{1}{(d_k^{\mathrm{T}} H_k d_k)^{1/2}} H_k d_k \right)^{\mathrm{T}}.$$

Notice that $d_k^{\mathrm{T}} H_k d_k = d_k^{\mathrm{T}} R_k^{\mathrm{T}} R_k d_k = (R_k d_k)^{\mathrm{T}}(R_k d_k) = \|R_k d_k\|^2$. The expression for $\widehat{R}_k$ then becomes

$$\widehat{R}_k = R_k + \frac{1}{\|R_k d_k\|} R_k d_k \left(\pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2}} y_k - \frac{1}{\|R_k d_k\|} H_k d_k \right)^{\mathrm{T}},$$

or simply $\widehat{R}_k = R_k + u_k v_k^{\mathrm{T}}$ with $u_k = q_k/\|q_k\|$ and $v_k = \pm y_k/(y_k^{\mathrm{T}} d_k)^{1/2} - R_k^{\mathrm{T}} u_k$. The choice of sign indicates that both

$$v_k^+ = \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2}} y_k - R_k^{\mathrm{T}} u_k \quad \text{and} \quad v_k^- = -\frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2}} y_k - R_k^{\mathrm{T}} u_k$$

are appropriate values for v_k in the rank-one update. The magnitudes of intermediate quantities will be minimized if the sign is chosen to make the norm of the update as small as possible. Using that fact that u_k is a unit vector, as well as the basic properties of the two-norm, we have $\|u_k v_k^{\mathrm{T}}\|_2 = \|u_k\|_2 \|v_k\|_2 = \|v_k\|_2$. As

$$v_k^{\mathrm{T}} v_k = \frac{1}{y_k^{\mathrm{T}} d_k} y_k^{\mathrm{T}} y_k \mp \frac{2}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} y_k^{\mathrm{T}} H_k d_k + \frac{1}{d_k^{\mathrm{T}} H_k d_k} \|H_k d_k\|_2^2,$$

it follows that the value of v_k with the least norm is v_k^+ if $y_k^{\mathrm{T}} H_k d_k < 0$ (i.e., $y_k^{\mathrm{T}} \nabla f(x_k) < 0$), and v_k^- otherwise.

The matrix $\widehat{R}_k = R_k + u_k v_k^{\mathrm{T}}$ has the desired property that $\widehat{R}_k^{\mathrm{T}} \widehat{R}_k = H_{k+1}$, however it is *not* upper triangular and therefore not a Cholesky factor of H_{k+1}. Hence the principal effort associated with the update procedure is devoted to obtaining a triangular R_{k+1} from $\widehat{R}_k$. If Q_k is an orthogonal matrix with $Q_k^{\mathrm{T}} Q_k = I$, we have

$$H_{k+1} = \widehat{R}_k^{\mathrm{T}} \widehat{R}_k = \widehat{R}_k^{\mathrm{T}} Q_k^{\mathrm{T}} Q_k \widehat{R}_k.$$

It remains to choose the orthogonal matrix Q_k so that $R_{k+1} = Q_k \widehat{R}_k$ *is* upper triangular. The desired effect is achieved with two products, or *sweeps*, of plane rotations. To simplify the notation the suffix k is omitted and the symbol $+$ is used to denote quantities associated with the $(k+1)$th iteration.

Let e_n denote the nth coordinate vector. The first sweep S_1 has two properties. First, S_1 should reduce the vector u to a multiple of e_n. In particular, we must have $S_1 u = \|w\|_2 e_n = e_n$, because orthogonal transformations preserve Euclidean length, and u is a unit vector. The result of applying S_1 to the rank-one matrix uv^{T} is then a specially structured rank-one matrix $e_n v^{\mathrm{T}}$, whose first $n-1$ rows are all zero and whose last row is simply v^{T}. The second property of S_1 is to leave the upper-triangular structure of R as intact as possible. This is achieved by choosing the *order* of the rotations in S_1 so that they leave the upper-triangular structure of R unaltered except for creating a "horizontal spike" of nonzeros in the last row. These properties are achieved by constructing S_1 from rotations in the $(n, n-1)$, $(n, n-2)$, ..., $(n, 1)$ planes that give $S_1 u = e_n$.

As $S_1 uv^{\mathrm{T}}$ contains nonzeros only in its last row, the combined matrix

$$\widetilde{R} = S_1 \widehat{R} = S_1 (R + uv^{\mathrm{T}})$$

has the same structure as S_1R, i.e., is upper-triangular except for a nonzero spike in the last row. To achieve the desired upper-triangular form for $S_2\widetilde{R}$, S_2 is taken as a sweep of rotations in the planes $(1, n)$, $(2, n)$, …, $(n-1, n)$, designed to annihilate the last element of each column of $\widetilde{R}$.

The required updated Cholesky factor R_+ is given by $R_+ = S_2S_1\bar{R}$. The total work required for this procedure, including finding the solution of two triangular systems is, $7n^2 + n + 32$ floating point operations.

Recurrence relations with improved efficiency. If we wish to derive a method for determining Q and applying it to $\widehat{R}$ that is both numerically stable and computationally efficient, we need to understand and utilize the special structure of the matrices representing plane rotations and their products. Given any $z \in \mathbb{R}^n$ and for any $1 \le i < j \le n$, it is always possible to find an orthogonal plane rotation P_j^i so that $(P_j^i\, z)_j = 0$ and $(P_j^i\, u)_i = \pm\tau$, where $\tau = \|(z_i,\ z_j)^{\mathrm{T}}\|_2$. This matrix is formed by embedding the two-by-two matrix

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \qquad \text{such that} \qquad c = \frac{z_i}{\rho} \quad \text{and} \quad s = \frac{z_j}{\rho}$$

in the ith and jth rows and columns of the identity matrix I_n (see [8] for details). It follows that the product $P = P_2^1P_3^2\cdots P_n^{n-1}$ satisfies $Pz = \pm\rho e_1$ with $\rho = \|z\|$. Gill et al. [8] show that this product is upper-Hessenberg and has the form

$$P = \begin{pmatrix} \bar{\beta}_1z_1 & \bar{\beta}_1z_2 & \bar{\beta}_1z_3 & \cdots & \bar{\beta}_1z_n \\ -s_1 & \bar{\beta}_2z_2 & \bar{\beta}_2z_3 & \cdots & \bar{\beta}_2z_n \\ & -s_2 & \bar{\beta}_3z_3 & \cdots & \bar{\beta}_3z_n \\ & & \ddots & \ddots & \vdots \\ & & & -s_{n-1} & \bar{\beta}_nz_n \end{pmatrix}.$$

Knowing this special structure means that in order to determine and work with P we need only compute s and $\bar{\beta}$, which can be computed efficiently using the following recurrence relation.

Algorithm Recurrence Relation 1: Compute $P = P_2^1P_3^2\cdots P_n^{n-1}$.

```
Input $z$;
Set $\bar{\beta}_n = 1/z_n$;
**for** $j = n-1,\ n-2,\ \ldots,\ 1$ **do**
    $\rho_j = \bar{\beta}_{j+1}z_j$;
    $s_j = 1/(\rho_j^2 + 1)^{1/2}$;   $c_j = \rho_j s_j$;
    $\bar{\beta}_j = s_j\bar{\beta}_{j+1}$;   $\bar{\beta}_{j+1} = c_j\bar{\beta}_{j+1}$;
**end for**
**return** $\bar{\beta}$, $s$;
```

Returning to our rank-one update $\widehat{R} = R + uv^{\mathrm{T}}$, we let $\widehat{v}$ be the solution of $R^{\mathrm{T}}\widehat{v} = v$ so that we can write $\widehat{R} = (I + u\widehat{v}^{\mathrm{T}})R$. We can use recurrence relation 1 to compute an orthogonal product of plane rotations Q_1, chosen so that $Q_1 u = \pm e_1$. Then premultiplying gives

$$Q_1 \widehat{R} = (Q_1 + \bar{\sigma}_1 e_1 \widehat{v}^{\mathrm{T}})R = H_u R,$$

where $\bar{\sigma}_1 = \frac{1}{\bar{\beta}_1}$. The matrix $e_1 \widehat{v}^{\mathrm{T}}$ is just the zero matrix with $\widehat{v}^{\mathrm{T}}$ installed in the first row. This means that H_u is Q_1 with $\bar{\sigma}_1 \widehat{v}^{\mathrm{T}}$ added to the first row. It follows that H_u is an upper-Hessenberg matrix of the form

$$H_u = \begin{pmatrix} \bar{\beta}_1 u_1 + \bar{\sigma}_1 \widehat{v}_1 & \bar{\beta}_1 u_2 + \bar{\sigma}_1 \widehat{v}_2 & \bar{\beta}_1 u_3 + \bar{\sigma}_1 \widehat{v}_3 & \cdots & \bar{\beta}_1 u_n + \bar{\sigma}_1 \widehat{v}_n \\ -s_1 & \bar{\beta}_2 u_2 & \bar{\beta}_2 u_3 & \cdots & \bar{\beta}_2 u_n \\ & -s_2 & \bar{\beta}_3 u_3 & \cdots & \bar{\beta}_3 u_n \\ & & \ddots & \ddots & \vdots \\ & & & -s_{n-1} & \bar{\beta}_n u_n \end{pmatrix}. \quad (3.2)$$

Next we need another orthogonal matrix Q_2 that is composed of a sequence of plane rotations chosen to annihilate the sub-diagonal elements of H_u. The resulting matrix $\widetilde{R} = Q_2 H_u$ will then be upper triangular and of the form

$$\widetilde{R} = \begin{pmatrix} \lambda_1 & \beta_1 u_2 + \sigma_1 \widehat{v}_2 & \beta_1 u_3 + \sigma_1 \widehat{v}_3 & \cdots & \beta_1 u_n + \sigma_1 \widehat{v}_n \\ & \lambda_2 & \beta_2 u_3 + \sigma_2 \widehat{v}_3 & \cdots & \beta_2 u_n + \sigma_2 \widehat{v}_n \\ & & \lambda_3 & \cdots & \beta_3 u_n + \sigma_3 \widehat{v}_n \\ & & & \ddots & \vdots \\ & & & & \lambda_n \end{pmatrix}.$$

Due to the structure of $\widetilde{R}$ we can also compute λ, β, and σ using recurrences. Let $\bar{\lambda}_j = \bar{\beta}_j u_j + \bar{\sigma}_j \widehat{v}_j$, and consider the 2×2 product

$$\begin{pmatrix} \bar{c}_j & -\bar{s}_j \\ \bar{s}_j & \bar{c}_j \end{pmatrix} \begin{pmatrix} \bar{\lambda}_j & \bar{\beta}_j u_{j+1} + \bar{\sigma}_j \widehat{v}_{j+1} \\ -s_j & \bar{\beta}_{j+1} u_{j+1} \end{pmatrix}$$
$$= \begin{pmatrix} \bar{c}_j \bar{\lambda}_j + \bar{s}_j s_j & (\bar{c}_j \bar{\beta}_j - \bar{s}_j \bar{\beta}_{j+1}) u_{j+1} + \bar{c}_j \bar{\sigma}_j \widehat{v}_{j+1} \\ \bar{s}_j \bar{\lambda}_j - \bar{c}_j s_j & (\bar{s}_j \bar{\beta}_j + \bar{c}_j \bar{\beta}_{j+1}) u_{j+1} + \bar{s}_j \bar{\sigma}_j \widehat{v}_{j+1} \end{pmatrix}. \quad (3.3)$$

To delete the sub-diagonal of (3.2), we require $\bar{s}_j \bar{\lambda}_j - \bar{c}_j s_j = 0$. It is also apparent from (3.3) that we need $\lambda_j = \bar{c}_j \bar{\lambda}_j + \bar{s}_j s_j = (\bar{\lambda}_j^2 + s_j^2)^{1/2}$, which is accomplished by putting $\bar{c}_j = \bar{\lambda}_j / \lambda_j$ and $\bar{s}_j = s_j / \lambda_j$. The values of β_j, σ_j are the coefficients of u_{j+1} and $\widehat{v}_{j+1}$ in the first row of the right side of (3.3), namely $\beta_j = \bar{c}_j \bar{\beta}_j - \bar{s}_j \bar{\beta}_{j+1}$ and $\sigma_j = \bar{c}_j \bar{\sigma}_j$. The intermediate values of $\bar{\beta}$, $\bar{\sigma}$, and $\bar{\lambda}$ are also read off of product (3.3) in a similar way. The entire procedure for computing Q_2 and forming $\widetilde{R} = Q_2 H_u$ is accomplished by the following recurrence relation.

Algorithm Recurrence Relation 2: Compute $\widetilde{R} = Q_2 H_u$.

```
Input $\bar{\beta}$, $s$, $u$, and $\widehat{v}$;
Set $\bar{\sigma}_1 = 1/\bar{\beta}_1$;   $\bar{\lambda}_1 = \bar{\beta}_1 u_1 + \bar{\sigma}_1 \widehat{v}_1$;
for $j = 1, 2, \ldots, n-1$ do
    $\lambda_j = (\bar{\lambda}_j^2 + s_j^2)^{1/2}$;
    $\bar{c}_j = \dfrac{\bar{\lambda}_j}{\lambda_j}$;   $\bar{s}_j = \dfrac{s_j}{\lambda_j}$;
    $\beta_j = \bar{c}_j \bar{\beta}_j - \bar{s}_j \bar{\beta}_{j+1}$;   $\sigma_j = \bar{c}_j \bar{\sigma}_j$;
    $\bar{\beta}_{j+1} = \bar{s}_j \bar{\beta}_j + \bar{c}_j \bar{\beta}_{j+1}$;   $\bar{\sigma}_{j+1} = \bar{s}_j \bar{\sigma}_j$;   $\bar{\lambda}_{j+1} = \bar{\beta}_{j+1} u_{j+1} + \bar{\sigma}_{j+1} \widehat{v}_{j+1}$;
end for
return $\lambda$, $\beta$, and $\sigma$;
```

We can now form the product $R^+ = \widetilde{R}R$ efficiently. The following recurrence relation is analogous to the column update proposed by Goldfarb [10], and can also be viewed as a modification of the backward recurrence of Lemma V given in Gill et al. [8].

Algorithm Recurrence Relation 3: Compute $R^+ = \widetilde{R}R$.

```
Input $R$, $\beta$, $u$, $\sigma$, $\widehat{v}$, and $\lambda$.
for $i = n, n-1, \ldots, 1$ do
    $\tilde{u} = u_i r_{ii}$;   $\widetilde{v} = \widehat{v}_i r_{ii}$;
    for $j = i+1, i+2, \ldots, n$ do
        $r_{ij}^+ = \lambda_i r_{ij} + \beta_i \tilde{u}_j + \sigma_i \widetilde{v}_j$;
        $\tilde{u}_j = \tilde{u}_j + u_j r_{ij}$;   $\widetilde{v}_j = \widetilde{v}_j + \widehat{v}_j r_{ij}$;
    end for
end for
return $R^+$;
```

We now have the upper triangular factor $R^+ = \widetilde{R}R$ and the orthogonal matrix $Q = Q_2 Q_1$. It follows that

$$R^{+\mathrm{T}} R^+ = (\widetilde{R}R)^{\mathrm{T}} \widetilde{R}R = R^{\mathrm{T}} H_u^{\mathrm{T}} Q_2^{\mathrm{T}} Q_2 H_u R = \widehat{R}^{\mathrm{T}} Q_1^{\mathrm{T}} Q_2^{\mathrm{T}} Q_2 Q_1 \widehat{R} = \widehat{R}^{\mathrm{T}} \widehat{R} = H^+,$$

which implies that R^+ is the required Cholesky factor for the next iteration. Note that this method for updating $H_k = R_k^{\mathrm{T}} R_k$ is equivalent to the method for updating $H_k = L_k D_k L_k^{\mathrm{T}}$ proposed by Goldfarb [10], the only difference being transposition and that D_k is updated separately.

Although this technique requires solving the additional triangular system $R_k^{\mathrm{T}} \widehat{v}_k = v_k$, the overall efficiency is improved. The total work required is $6n^2 + 30n - 20$ floating-point operations, thus the improvement gained from using recurences is $O(n^2)$.

Algorithm `bfgsR`: BFGS with factored Hessian approximation.

```
Choose $x_0 \in \mathbb{R}^n$;   $k \leftarrow 0$;
while $\|\nabla f(x_k)\| > \varepsilon$ and $k \le N$ do
    Solve $R_k^{\mathrm{T}} q_k = -\nabla f(x_k)$;   Solve $R_k p_k = q_k$;
    $\alpha_k = \mathtt{Wolfe}(x_k, f(x_k), \nabla f(x_k), p_k)$;
    $d_k = x_k + \alpha_k p_k$;   $x_{k+1} = x_k + d_k$;   $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
    if $y_k^{\mathrm{T}} d_k > 0$ then
        $u_k = \dfrac{1}{\|q_k\|} q_k$;   $v_k = \pm \dfrac{1}{(y_k^{\mathrm{T}} d_k)^{1/2}} y_k - R_k^{\mathrm{T}} u_k$;
        Solve $R_k^{\mathrm{T}} \widehat{v}_k = v_k$;
        $(\bar{\beta}_k,\ s_k) = \mathtt{recurrence1}(u_k)$;
        $(\lambda_k,\ \beta_k,\ \sigma_k) = \mathtt{recurrence2}(\bar{\beta}_k, s_k, u_k, \widehat{v}_k)$;
        $R_{k+1} = \mathtt{recurrence3}(R_k, \beta_k, u_k, \sigma_k, \widehat{v}_k, \lambda_k)$;
    else
        $R_{k+1} = R_k$;
    end if
    $k = k + 1$;
end while
```

3.2. The self-scaled factored BFGS method (`bfgsRS`).

All that is needed to apply the scaling described in Section 2 is to use the square root of the scaling factor. As $H_k = R_k^{\mathrm{T}} R_k$, then H_k may be scaled implicitly as $R_k \leftarrow \gamma_k^{1/2} R_k$ just before the triangular factor is updated. This scaling requires $\frac{1}{2}n(n+1)$ multiplications, which does not increase the operation count significantly.

3.3. Conjugate factored inverse BFGS (`bfgsZ`)

In contrast to the preceding method, this variant of BFGS is based on maintaining a factorization of the inverse Hessian approximation. If Z_k is a nonsingular matrix such that

$$M_k = Z_k Z_k^{\mathrm{T}}, \tag{3.4}$$

then Z_k is said to be a conjugate factor of M_k. Note that Z_k is not unique because if Z_k is a conjugate factor, then so is $Z_k \Omega$ for any orthogonal matrix Ω. The term conjugate factorization is used because the columns of Z_k satisfy the conjugacy condition with respect to H_k, i.e., $z_i^{\mathrm{T}} H_k z_j = 0$ if $i \neq j$, and $z_j^{\mathrm{T}} H_k z_j = 1$, where z_i denotes the ith column of Z_k. This condition implies that (3.4) may be written as $Z_k^{\mathrm{T}} H_k Z_k = I$.

Result 3.1. *The BFGS update for M_k may be written in the form*

$$M_{k+1} = \big(I + d_k w_k^{\mathrm{T}}\big) M_k \big(I + w_k d_k^{\mathrm{T}}\big),$$

$$\text{with} \quad w_k = \pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} H_k d_k - \frac{1}{y_k^{\mathrm{T}} d_k} y_k.$$

Proof. From (3.1) the BFGS update for M_k $(= H_k^{-1})$ may be written as

$$M_{k+1} = (I + d_k v_k^{\mathrm{T}})^{-1} M_k (I + v_k d_k^{\mathrm{T}})^{-1},$$
$$\text{with } \; v_k = \pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} y_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k.$$

The Sherman-Morrison-Woodbury formula implies that

$$(I + v_k d_k^{\mathrm{T}})^{-1} = I + \beta_k v_k d_k^{\mathrm{T}}, \;\; \text{with } \; \beta_k = -\frac{1}{1 + d_k^{\mathrm{T}} v_k},$$

so that

$$M_{k+1} = (I + \beta_k d_k v_k^{\mathrm{T}}) M_k (I + \beta_k v_k d_k^{\mathrm{T}}),$$
$$\text{with } \; v_k = \pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} y_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k.$$

Then

$$\begin{aligned} 1 + d_k^{\mathrm{T}} v_k &= 1 + d_k^{\mathrm{T}} \left(\pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} y_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k \right) \\ &= 1 \pm \frac{(d_k^{\mathrm{T}} y_k)^{1/2}}{(d_k^{\mathrm{T}} H_k d_k)^{1/2}} - 1 = \pm \frac{(d_k^{\mathrm{T}} y_k)^{1/2}}{(d_k^{\mathrm{T}} H_k d_k)^{1/2}}. \end{aligned}$$

It follows that

$$\beta_k = -\frac{1}{1 + d_k^{\mathrm{T}} v_k} = \mp \frac{(d_k^{\mathrm{T}} H_k d_k)^{1/2}}{(y_k^{\mathrm{T}} d_k)^{1/2}}.$$

If we write $w_k = \beta_k v_k$, then

$$M_{k+1} = (I + d_k w_k^{\mathrm{T}}) M_k (I + w_k d_k^{\mathrm{T}}),$$

with

$$\begin{aligned} w_k = \beta_k v_k &= \mp \frac{(d_k^{\mathrm{T}} H_k d_k)^{1/2}}{(d_k^{\mathrm{T}} y_k)^{1/2}} \left(\pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} y_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k \right) \\ &= -\frac{1}{y_k^{\mathrm{T}} d_k} y_k \pm \frac{1}{(y_k^{\mathrm{T}} d_k)^{1/2} (d_k^{\mathrm{T}} H_k d_k)^{1/2}} H_k d_k, \end{aligned}$$

as required. ∎

If M_k is written in the form $M_k = Z_k Z_k^{\mathrm{T}}$, the inverse update may be written in the product form

$$M_{k+1} = (I + d_k w_k^{\mathrm{T}}) Z_k Z_k^{\mathrm{T}} (I + w_k d_k^{\mathrm{T}}) = \big((I + d_k w_k^{\mathrm{T}}) Z_k \big) \big((I + d_k w_k^{\mathrm{T}}) Z_k \big)^{\mathrm{T}},$$

or, equivalently, $M_{k+1} = Z_{k+1}Z_{k+1}^{\mathrm{T}}$, with $Z_{k+1} = (I + d_k w_k^{\mathrm{T}})Z_k$.

For the moment we will assume all variables are associated with iteration k so that the suffix may be omitted. Powell [25] first replaces the factor Z with $\widehat{Z} = Z\Omega$, where Ω is orthogonal. The matrix Ω is chosen so that the first column of $\widehat{Z}$ is a multiple d. Note that

$$d = -\alpha M\nabla f(x) = -\alpha ZZ^{\mathrm{T}}\nabla f(x) = Zs, \quad \text{where} \quad s = -\alpha Z^{\mathrm{T}}\nabla f(x).$$

Let Ω be the product of plane rotations $\Omega^{\mathrm{T}} = P_2^1 P_3^2 \cdots P_n^{n-1}$, where P_{i+1}^i is a rotation in the $(i, i+1)$ plane such the $(i+1)$th component of the product $P_{i+1}^i P_{i+2}^{i+1} \cdots P_n^{n-1} s$ is zero. This choice of Ω gives

$$\Omega^{\mathrm{T}} s = P_2^1 P_3^2 \cdots P_n^{n-1} s = \|s\| e_1,$$

in which case

$$d = Zs = Z\Omega\Omega^{\mathrm{T}} s = \widehat{Z}\Omega^{\mathrm{T}} s = \widehat{Z}(\|s\|e_1) = \|s\|\widehat{z}_1,$$

and $\widehat{z}_1$ is a multiple of d.

The next step is to compute the product $\bar{Z} = (I + dw^{\mathrm{T}})\widehat{Z}$. The first column of $\bar{Z}$ is $\bar{z}_1 = (I + dw^{\mathrm{T}})\widehat{Z}e_1 = (I + dw^{\mathrm{T}})\widehat{z}_1 = \widehat{z}_1 + dw^{\mathrm{T}}\widehat{z}_1$. If we substitute the value of w from Result 3.1 we obtain

$$\begin{aligned}\widehat{z}_1 + dw^{\mathrm{T}}\widehat{z}_1 &= \frac{1}{\|s\|}d + d\left(\frac{1}{(y^{\mathrm{T}}d)^{1/2}(d^{\mathrm{T}}Hd)^{1/2}}d^{\mathrm{T}}H - \frac{1}{y^{\mathrm{T}}d}y^{\mathrm{T}}\right)\frac{1}{\|s\|}d \\ &= \left(\frac{1}{\|s\|}d - \frac{1}{\|s\|}d\right) + \frac{(d^{\mathrm{T}}Hd)^{1/2}}{(y^{\mathrm{T}}d)^{1/2}\|s\|}d = \frac{1}{(y^{\mathrm{T}}d)^{1/2}}d,\end{aligned}$$

because $d^{\mathrm{T}}Hd = s^{\mathrm{T}}(Z^{\mathrm{T}}HZ)s = \|s\|^2$. For columns $j = 2, 3, \ldots, n$ we have

$$\bar{z}_j = \bar{Z}e_j = \widehat{z}_j + dw^{\mathrm{T}}\widehat{z}_j = \widehat{z}_j - \left(\frac{y^{\mathrm{T}}\widehat{z}_j}{y^{\mathrm{T}}d}\right)d.$$

This follows from the conjugacy property $\widehat{Z}^{\mathrm{T}}H\widehat{Z} = I$ and the fact that d is a multiple of $\widehat{z}_1$, i.e., $d^{\mathrm{T}}H\widehat{z}_j = 0$ for $j = 2, 3, \ldots, n$. We now have a practical formula for computing the conjugate factor $\bar{Z} = (I + dw^{\mathrm{T}})\widehat{Z}$, i.e.,

$$\bar{z}_j = \begin{cases} \dfrac{1}{(y^{\mathrm{T}}d)^{1/2}}d & \text{if } j = 1; \\[2ex] \widehat{z}_j - \left(\dfrac{y^{\mathrm{T}}\widehat{z}_j}{y^{\mathrm{T}}d}\right)d & \text{if } j = 2, 3, \ldots, n. \end{cases}$$

The algorithm for this conjugate factored variant of the BFGS method is given in Algorithm `bfgsZ`.

Algorithm `bfgsZ`: Conjugate factored inverse BFGS.

```
Choose $x_0 \in \mathbb{R}^n$;   $k \leftarrow 0$;
while $\|\nabla f(x_k)\| > \varepsilon$ and $k \le N$ do
    $q_k = Z_k^{\mathrm{T}} \nabla f(x_k)$;   $p_k = -Z_k q_k$;
    $\alpha_k = \mathtt{Wolfe}(x_k, f(x_k), \nabla f(x_k), p_k)$;
    $d_k = \alpha_k p_k$;   $x_{k+1} = x_k + d_k$;   $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
    if $y_k^{\mathrm{T}} d_k > 0$ then
        Compute $\Omega_k$;   $\widehat{Z} = Z_k \Omega_k$;
        $\bar{z}_j = \begin{cases} \dfrac{1}{(y_k^{\mathrm{T}} d_k)^{1/2}} d_k & \text{if } j = 1 \\ \widehat{z}_j - \left(\dfrac{y_k^{\mathrm{T}} \widehat{z}_j}{y_k^{\mathrm{T}} d_k}\right) d_k & \text{if } j = 2, 3, \dots, n \end{cases}$;
        $Z_{k+1} = \bar{Z}$;
    else
        $Z_{k+1} = Z_k$;
    end if
    $k \leftarrow k + 1$;
end while
```

Powell provides an efficient method for computing $Z_k \Omega_k$ without the need to form Ω_k explicitly. The method is summarized in Algorithm `modZ`.

Algorithm `modZ`: Compute $\widehat{Z} = Z\Omega$.

```
Set $j = \max\{1 \le i \le n : s_i \ne 0\}$;
Set $\sigma = s_j z_j$;   Set $\tau = s_j^2$;
for $i = j : -1 : 2$ do
    $\widehat{z}_i = \Big(\dfrac{\tau}{s_{i-1}^2 + \tau}\Big)^{1/2} \Big( - z_{i-1} + \dfrac{s_{i-1}}{\tau} \sigma \Big)$;
    $\sigma \leftarrow \sigma + s_{i-1} z_{i-1}$;   $\tau \leftarrow \tau + s_{i-1}^2$;
end for
```

Note that in the above routine s_j is the jth component of s, while z_j is the jth column of Z.

Automatic rescaling The conjugacy property satisfied by the columns of Z_k can introduce numerical difficulies when Z_0 is nearly singular. For example, suppose $\operatorname{rank}(Z_0) = n - 1$ and the method is applied to a quadratic objective function in infinite precision. The first $n - 1$ columns of Z_n will be mutually orthogonal, and therefore span the $n - 1$ dimensional column space of Z_0. The last column of Z_n must simultaneously be in the range of, and orthogonal to, $z_1, \dots, z_{n-1}$, and therefore identically zero. If Z_0. If the objective function is well-approximated by a quadratic near the kth iterate, as is usually the case, and Z_k is nearly singular from an accumulation of errors, then the same line of reasoning suggests the method would produce a final column of Z_{k+n} that is disproportionately small. Powell [25] recommends a technique to automatically rescale the columns of Z_k to prevent this issue, that extends well to general nonlinear objective functions.

Keep a running record of the minimum norm of the first column observed up to and including the kth iteration. Then scale each of the remaining columns, if necessary, so their norm is no less than the running record. This technique is summarized in the following algorithm.

Algorithm `autoscale`: Automatic Rescaling of Z_k

```
if $k = 0$ then
    $\sigma_k = +\infty$;
else
    $\sigma_k = \min\{\sigma_{k-1}, \|z_1\|\}$
end if
for $j = 2 : n$ do
    $\beta = \|z_j\|$;
    if $\beta < \sigma_k$ then
        $z_j = \frac{\sigma}{\beta} z_j$;
    end if
end for
```

It should be noted that the automatic scaling procedure technically separates `bfgsZ`from the other solvers because once auto-scaling occurs, the resulting Hessian approximation need not satisfy the quasi-Newton condition 1.2.

4. Modified BFGS Methods

In the next section we derive and analyze several modifications of the BFGS method. These take the form of either changing the update formula while still satisfying the conventional quasi-Newton condition, or by changing the quasi-Newton condition itself and then finding a suitable update that satisfies the modified condition. These modifications are motivated by various assumptions. For example, incorporating more accurate curvature information will yield better search directions; or reducing the condition number of the approximate Hessian will provide better numerical stability.

The first modified BFGS method was proposed by Powell [24] in the context of constrained optimization. In this method, y_k is replaced by a vector $\widetilde{y}_k$ in the update (1.5) when $y_k^{\mathrm{T}} d_k$ is not sufficiently positive. This requires the definition of a "least acceptable curvature". A popular strategy is to regard $y_k^{\mathrm{T}} d_k$ as being sufficiently positive if

$$y_k^{\mathrm{T}} d_k \geq (1-\mu) d_k^{\mathrm{T}} H_k d_k, \tag{4.1}$$

where μ is a constant parameter such $0 \leq \mu \leq 1$. Nocedal and Wright [16, p. 540] propose the value $\mu = 0.8$.

Powell proposed that $\widetilde{y}_k$ be computed as a convex combination of y_k and $H_k y_k$, i.e.,

$$\widetilde{y}_k = \theta_k y_k + (1-\theta_k) H_k d_k,$$

where

$$\theta_k = \begin{cases} 1 & \text{if } y_k^{\mathrm{T}} d_k \geq (1-\mu) d_k^{\mathrm{T}} H_k d_k; \\ \dfrac{\mu d_k^{\mathrm{T}} H_k d_k}{d_k^{\mathrm{T}} H_k d_k - y_k^{\mathrm{T}} d_k} & \text{if } y_k^{\mathrm{T}} d_k < (1-\mu) d_k^{\mathrm{T}} H_k d_k. \end{cases}$$

The Powell modification is *always* well defined, which implies that it is always applied—even when it might be unwarranted because f has negative curvature.

Cautious updates The positivity and magnitude of the approximate curvature plays an important role in theoretical convergence analysis. Showing that the BFGS method is globally convergent for general well-posed minimization problems has proved very difficult and has not yet been accomplished. However, it is widely believed to be true and there is ample practical support for this claim. Making certain assumptions about, or imposing restrictions on, the approximate curvature has allowed several authors to make headway on proving global convergence.

The inequality

$$\frac{\|y_k\|^2}{y_k^{\mathrm{T}} d_k} \leq M, \tag{4.2}$$

where $M > 0$ is constant, is not difficult to verify when f is convex. Of course, this inequality need not hold in general. However, if 4.2 is *assumed* to hold for all iterations k, then global convergence can be shown. See Powell [23] for the details of the proof. This result is extended to the convex Broyden class of updates with parameter $\phi \in [0,\ 1)$ by Byrd et al. [4].

An alternative approach that is well-suited to practical application is proposed by Li and Fukushima [14]. They suggest the update rule

$$H_{k+1} = \begin{cases} H_k - \dfrac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k d_k^{\mathrm{T}} H_k + \dfrac{1}{y_k^{\mathrm{T}} d_k} y_k y_k^{\mathrm{T}} & \dfrac{y_k^{\mathrm{T}} d_k}{\|d_k\|^2} \geq \epsilon \|g_k\|^{\alpha} \\ H_k & \text{otherwise,} \end{cases}$$

where ϵ and α are positive parameters that can be chosen. The authors were able to prove global convergence when this update rule is used, and suggest $\epsilon =$ `1.00e-06` and either $\alpha = 1$ or $\alpha = 0.01$ when $\|g_k\| \geq 1$ and $\alpha = 3$ otherwise.

Our numerical experiments show that imposing these update restrictions hinders the performance of the BFGS method and the overall best performance is achieved by only requiring $y_k^{\mathrm{T}} d_k > 0$. This seems to support the belief that the BFGS method is in fact globally convergent, and these assumptions and restrictions on the approximate curvature are helpful for showing theoretical properties but are not necessary or helpful in practice. See the numerical results in Section 5 for more details.

4.1. Function interpolation (bfgsY)

The search direction obtained by solving $H_k p_k = -\nabla f(x_k)$ is the solution of the quadratic subproblem $\min_{p\in\mathbb{R}^n} \; q_k(p)$. where $q_k(p)$ is the quadratic model (1.1). For small values of $\|p\|$, $q_k(p)$ approximates $f(x_k+p)$, in particular, $q_k(p)$ satisfies the interpolation conditions.

$$q_k(0) = f(x_k), \qquad \nabla q_k(0) = \nabla f(x_k), \quad \text{and} \quad \nabla^2 q_k(0) = H_k. \tag{4.3}$$

Yuan [30] gives a modified BFGS method in which the modified gradient difference $\widetilde{y}_k$ is obtained by imposing an additional interpolation condition on the quadratic model $q_k(p)$ of (1.1). If the quasi-Newton condition $H_{k+1}d_k = y_k$ holds at x_{k-1} then the interpolation conditions (4.3) imply

$$\begin{aligned}\nabla q_k(x_{k-1}-x_k) &= \nabla f(x_k) + H_k(x_{k-1}-x_k)\\ &= \nabla f(x_k) - \big(\nabla f(x_k) - \nabla f(x_{k-1})\big) = \nabla f(x_{k-1}).\end{aligned}$$

These properties imply that $q_k(x-x_k)$ is a quadratic interpolant of $f(x)$ at x_k and x_{k-1}.

We want to introduce an extra parameter γ_k and update the Hessian approximation using the scaled update

$$H_{k+1} = H_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k d_k^{\mathrm{T}} H_k + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} y_k y_k^{\mathrm{T}}. \tag{4.4}$$

In order to determine γ_k we need to impose an extra condition. The quasi-Newton variant derived by Yuan [30] is based on the condition

$$q_k(x_{k-1}-x_k) = f(x_{k-1}). \tag{4.5}$$

Notice that this requires the quadratic model $q_k(x-x_k)$ to interpolate $f(x)$ at $x = x_{k-1}$. It is similar to the relation $\nabla q_k(x_{k-1}-x_k) = \nabla f(x_{k-1})$ that followed naturally from the interpolation conditions (4.3) in that the same x values are considered, but the interpolation condition is imposed on q_k rather than its gradient.

To derive a usable method from this condition, observe that if k is increased by one in (4.5) the condition becomes $q_{k+1}(x_k - x_{k+1}) = f(x_k)$. Rewriting the quadratic model with k also increased by one yields $q_{k+1}(p) = f(x_{k+1}) + p^{\mathrm{T}}\nabla f(x_{k+1}) + \frac{1}{2}p^{\mathrm{T}} H_{k+1} p$. Substituting $p = x_k - x_{k+1}$ into the model and setting the result equal to $f(x_k)$, i.e., applying (4.5), gives

$$q_{k+1}(-d_k) = f(x_{k+1}) - d_k^{\mathrm{T}}\nabla f(x_{k+1}) + \tfrac{1}{2} d_k^{\mathrm{T}} H_{k+1} d_k = f(x_k).$$

This can be solved for $d_k^{\mathrm{T}} H_{k+1} d_k$ to get the relation

$$d_k^{\mathrm{T}} H_{k+1} d_k = 2\left(f(x_k) - f(x_{k+1}) + \nabla f(x_{k+1})^{\mathrm{T}} d_k\right). \tag{4.6}$$

We want to update the approximate Hessian H_k using

$$H_{k+1} = H_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k d_k^{\mathrm{T}} H_k + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} y_k y_k^{\mathrm{T}},$$

Algorithm `bfgsY`: BFGS with function interpolation.

```
Choose $x_0 \in \mathbb{R}^n$;   $k \leftarrow 0$;
while $\|\nabla f(x_k)\| > \varepsilon$ and $k \le N$ do
    Solve $H_k p_k = -\nabla f(x_k)$;
    $\alpha_k = \texttt{Wolfe}(x_k, f(x_k), \nabla f(x_k), p_k)$;
    $d_k = \alpha_k p_k$;   $x_{k+1} = x_k + d_k$;   $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
    if $y_k^{\mathrm{T}} d_k > 0$ then
        $\gamma_k = \frac{2}{y_k^{\mathrm{T}} d_k}\big(f(x_k) - f(x_{k+1}) + \nabla f(x_{k+1})^{\mathrm{T}} d_k\big)$;   $\widetilde{y}_k = \gamma_k y_k$;
        $H_{k+1} = H_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k d_k^{\mathrm{T}} H_k + \frac{1}{\widetilde{y}_k^{\mathrm{T}} d_k} \widetilde{y}_k \widetilde{y}_k^{\mathrm{T}}$;
    else
        $H_{k+1} = H_k$;
    end if
    $k = k + 1$;
end while
```

so that the expression for $d_k^{\mathrm{T}} H_{k+1} d_k$ (4.6) is true. To determine what value of γ_k will accomplish this, left- and right-multiply the scaled BFGS update (4.4) by d_k to get

$$\begin{aligned} d_k^{\mathrm{T}} H_{k+1} d_k &= d_k^{\mathrm{T}} H_k d_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} d_k^{\mathrm{T}} H_k d_k d_k^{\mathrm{T}} H_k d_k + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} d_k^{\mathrm{T}} y_k y_k^{\mathrm{T}} d_k \\ &= d_k^{\mathrm{T}} H_k d_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} (d_k^{\mathrm{T}} H_k d_k)^2 + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} (y_k^{\mathrm{T}} d_k)^2 = \gamma_k y_k^{\mathrm{T}} d_k. \end{aligned}$$

Solving for γ_k and substituting (4.6) into the result gives the scaling factor

$$\gamma_k = \frac{1}{y_k^{\mathrm{T}} d_k} d_k^{\mathrm{T}} H_k d_k = \frac{2}{y_k^{\mathrm{T}} d_k}\big(f(x_k) - f(x_{k+1}) + \nabla f(x_{k+1})^{\mathrm{T}} d_k\big).$$

Algorithm `bfgsY` is based on using this value of γ_k in the scaled update (4.4).

4.2. Adaptive scaling (`bfgsN`)

The method proposed by Andrei [1] is derived as a combination of the conjugate-gradient method and the scaled BFGS method. It aims to scale the gradient difference y_k by a scaling factor that corrects large eigenvalues of the approximate inverse Hessian M_k and satisfies certain conjugacy conditions.

The conjugate-direction property for the BFGS method applied to a quadratic function $f(x) = c^{\mathrm{T}} x + \frac{1}{2} x^{\mathrm{T}} H x$ gives

$$d_i^{\mathrm{T}} H d_j = 0, \qquad i \ne j, \tag{4.7}$$

with H symmetric and positive definite. This condition can be extended to general nonlinear twice continuously differentiable functions. By the mean value theorem there exists some $\xi \in (0,1)$ for which

$$\nabla f(x_k + d_k) - \nabla f(x_k) = \nabla^2 f(x_k + \xi d_k) d_k.$$

Noting that the left-hand side is y_k and taking the inner product with d_{k+1} gives

$$d_{k+1}^{\mathrm{T}} y_k = d_{k+1}^{\mathrm{T}} \nabla^2 f(x_k + \xi d_k) d_k.$$

If this is to satisfy the conjugacy condition (4.7) then it may be reasonable to seek to achieve $d_{k+1}^{\mathrm{T}} y_k = 0$. As $\alpha_{k+1} \neq 0$, it follows that

$$p_{k+1}^{\mathrm{T}} y_k = 0. \tag{4.8}$$

The quasi-Newton condition $d_k = M_{k+1} y_k$ for the inverse approximate Hessian update can be combined with (4.8) to write

$$p_{k+1}^{\mathrm{T}} y_k = \big(- M_{k+1} \nabla f(x_{k+1}) \big)^{\mathrm{T}} y_k = -\nabla f(x_{k+1})^{\mathrm{T}} (M_{k+1} y_k) = -\nabla f(x_{k+1})^{\mathrm{T}} d_k.$$

The idea is to find a scaling factor γ_k so that if y_k is scaled as $\gamma_k y_k$, the magnitude of $-\gamma_k \nabla f(x_{k+1})^{\mathrm{T}} d_k$ is minimized.

An important consideration in the performance of inverse quasi-Newton methods is the conditioning of the inverse Hessian $\nabla^2 f(x_k)^{-1}$ and its approximation M_k. If M_k is ill-conditioned then the search direction obtained from $p_k = -M_k \nabla f(x_k)$ may be a poor choice or possibly not even a descent direction resulting in a line-search failure. To prevent this we hope to find γ_k so that the diagonal matrix $\gamma_k I$, which is always well-conditioned, is such that $\gamma_k I \approx \nabla^2 f(x_{k+1})$. In this case it would want $\|M_{k+1} y_k - \gamma_k I y_k\|$ as small as possible, i.e., γ_k is an eigenvalue of M_{k+1} with eigenvector y_k. Therefore another objective is to find γ_k for which $\|d_k - \gamma_k y_k\|^2$ is minimized and $\gamma_k \leq 1$.

As these two objectives cannot be satisfied simultaneously in general, the factor γ_k is chosen as

$$\gamma_k = \underset{\gamma \leq 1}{\operatorname{argmin}} \ \|d_k - \gamma y_k\|^2 + \gamma^2 |d_k^{\mathrm{T}} \nabla f(x_{k+1})|,$$

which yields

$$\gamma_k = \min\left(\frac{y_k^{\mathrm{T}} d_k}{\|y\|^2 + |d_k^{\mathrm{T}} \nabla f(x_{k+1})|}, 1 \right).$$

Once γ_k has been computed, the adaptive-scaled update to the Hessian approximation is

$$H_{k+1} = H_k - \frac{1}{d_k^{\mathrm{T}} H_k d_k} H_k d_k d_k^{\mathrm{T}} H_k + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} y_k y_k^{\mathrm{T}}. \tag{4.9}$$

An important property of the BFGS update is that if H_k is positive definite and $y_k^{\mathrm{T}} d_k$ is positive, then so is H_{k+1}. In the next result shows that the proposed scaling preserves hereditary positive definiteness

Result 4.1. (`bfgsN`: Hereditary Positive Definiteness) *If the step length α_k is determined by the Wolfe line search, H_k is positive definite, and $\gamma_k > 0$, then H_{k+1} as given by the scaled inverse update (4.9) is also positive definite.*

Proof. If $v \neq 0$ then by the Cauchy-Schwarz inequality implies that

$$(d_k^{\mathrm{T}} H_k v)^2 \leq (d_k^{\mathrm{T}} H_k d_k)(v^{\mathrm{T}} H_k v).$$

Conjugating the scaled inverse update (4.9) with v gives

$$\begin{aligned} v^{\mathrm{T}} H_{k+1} v &= v^{\mathrm{T}} H_k v - \frac{1}{d_k^{\mathrm{T}} H_k d_k} v^{\mathrm{T}} H_k d_k d_k^{\mathrm{T}} H_k v + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} v^{\mathrm{T}} y_k y_k^{\mathrm{T}} v \\ &= v^{\mathrm{T}} H_k v - \frac{1}{d_k^{\mathrm{T}} H_k d_k} (v^{\mathrm{T}} H_k d_k)^2 + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} (v^{\mathrm{T}} y_k)^2 \\ &\geq v^{\mathrm{T}} H_k v - \frac{1}{d_k^{\mathrm{T}} H_k d_k} (d_k^{\mathrm{T}} H_k d_k)(v^{\mathrm{T}} H_k v) + \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} (v^{\mathrm{T}} y_k)^2 \\ &= \gamma_k \frac{1}{y_k^{\mathrm{T}} d_k} (v^{\mathrm{T}} y_k)^2 > 0. \end{aligned}$$

It follows that $v^{\mathrm{T}} H_{k+1} v > 0$ for all $v \neq 0$, in which case H_{k+1} is positive definite. ∎

The Sherman-Morrison-Woodbury formula gives the corresponding rank-two update to the approximate inverse Hessian as

$$M_{k+1} = M_k - \frac{1}{\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k} M_k \widetilde{y}_k \widetilde{y}_k^{\mathrm{T}} M_k + \frac{1}{\widetilde{y}_k^{\mathrm{T}} d_k} d_k d_k^{\mathrm{T}} + (\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k) w_k w_k^{\mathrm{T}},$$

with $\widetilde{y}_k = \gamma_k y_k$.

Algorithm `bfgsN`: Adaptive, scaled BFGS.

```
Choose $x_0 \in \mathbb{R}^n$;   $k \leftarrow 0$;
while $\|\nabla f(x_k)\| > \varepsilon$ and $k \leq N$ do
    $p_k = -M_k \nabla f(x_k)$;
    $\alpha_k = \texttt{Wolfe}(x_k, f(x_k), \nabla f(x_k), p_k)$;
    $d_k = \alpha_k p_k$;   $x_{k+1} = x_k + d_k$;   $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
    if $y_k^{\mathrm{T}} d_k > 0$ then
        $\gamma_k = \min\left( \frac{y_k^{\mathrm{T}} d_k}{\|y\|^2 + |d_k^{\mathrm{T}} \nabla f(x_{k+1})|}, 1 \right)$;
        $\widetilde{y}_k = \gamma_k y_k$;   $w_k = \frac{1}{\widetilde{y}_k^{\mathrm{T}} d_k} d_k - \frac{1}{\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k} M_k \widetilde{y}_k$;
        $M_{k+1} = M_k - \frac{1}{\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k} M_k \widetilde{y}_k \widetilde{y}_k^{\mathrm{T}} M_k + \frac{1}{\widetilde{y}_k^{\mathrm{T}} d_k} d_k d_k^{\mathrm{T}} + (\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k) w_k w_k^{\mathrm{T}}$;
    else
        $M_{k+1} = M_k$;
    end if
    $k = k + 1$;
end while
```

4.3. Multistep quasi-Newton equations (`bfgsI`)

The next method to be considered is the result of deriving modified quasi-Newton equations, rather than a new method designed to satisfy the quasi-Newton condition $H_{k+1}d_k = y_k$. At each iteration, the data

$$f(x_k), \quad f(x_{k+1}), \quad \nabla f(x_k), \quad \text{and} \quad \nabla f(x_{k+1}), \tag{4.10}$$

are all available, but are not fully utilized by other methods. The approach developed by Zhang, Deng and Xu [31] is designed to provide a better approximation to $\nabla^2 f(x_k)d_k$ by using the data (4.10) to scale y_k to some $\widetilde{y}_k$, so that the quasi-Newton condition $H_{k+1}d_k = \widetilde{y}_k$ gives a more accurate Hessian approximation.

The path from x_k to x_{k+1} may be parameterized in terms of the parameter t by defining

$$x(t) = x_k + t\left(\frac{1}{\|d_k\|}d_k\right).$$

To understand how the gradient of f is changing at x_{k+1}, take the derivative with respect to t and evaluate at $t = \|d_k\|$. This gives us

$$\begin{aligned}\frac{d}{dt}\nabla f\big(x(t)\big)\Big|_{t=\|d_k\|} &= \nabla^2 f\big(x(t)\big)\frac{d}{dt}x(t)\Big|_{t=\|d_k\|}\\ &= \frac{1}{\|d_k\|}\nabla^2 f\big(x(t)\big)d_k\Big|_{t=\|d_k\|}\\ &= \frac{1}{\|d_k\|}\nabla^2 f(x_{k+1})d_k.\end{aligned}$$

This implies that $\nabla^2 f(x_{k+1})d_k = \|d_k\|\dfrac{d}{dt}\nabla f\big(x(t)\big)$. It seems reasonable to approximate $\nabla f\big(x(t)\big)$ with a quadratic polynomial

$$h(t) = at^2 + bt + c,$$

for some a, b, $c \in \mathbb{R}^n$. For h to interpolate $\nabla f\big(x(t)\big)$ at $t = 0$ and $t = \|d_k\|$ it is required that $h(0) = \nabla f\big(x(0)\big) = \nabla f(x_k)$ and $h(\|d_k\|) = \nabla f\big(x(\|d_k\|)\big) = \nabla f(x_{k+1})$. As h is meant to approximate $\nabla f\big(x(t)\big)$, it should satisfy the identity

$$\int_0^{\|d_k\|} \nabla f\big(x(t)\big)^{\mathrm{T}} x'(t)\,dt = \int_0^{\|d_k\|} \nabla f\big(x(t)\big)^{\mathrm{T}}\,dx(t) = f(x_{k+1}) - f(x_k),$$

which gives the third condition

$$\int_0^{\|d_k\|} h(t)^{\mathrm{T}} x'(t)\,dt = f(x_{k+1}) - f(x_k).$$

Putting these together gives the following conditions that h is required to satisfy

$$\left.\begin{aligned} h(0) &= \nabla f(x_k),\\ h(\|d_k\|) &= \nabla f(x_{k+1}),\\ \int_0^{\|d_k\|} h(t)^{\mathrm{T}} x'(t)\,dt &= f(x_{k+1}) - f(x_k). \end{aligned}\right\} \tag{4.11}$$

Requirements (4.11) have several implications. The first is that $c = \nabla f(x_k)$, which follows from evaluating h at $t = 0$. The second is that $\|d_k\|^2 a + \|d_k\| b = y_k$ and comes from evaluating h at $t = \|d_k\|$. Lastly we have that

$$\|d_k\|^2 a^{\mathrm{T}} d_k = 3\big(\nabla f(x_k) + \nabla f(x_{k+1})\big)^{\mathrm{T}} d_k - 6\big(f(x_{k+1} - f(x_k)\big),$$

which follows from

$$\begin{aligned}\int_0^{\|d_k\|} h(t)^{\mathrm{T}} x'(t)\,dt &= \int_0^{\|d_k\|} \frac{1}{\|d_k\|}(at^2 + bt + c)^{\mathrm{T}} d_k\,dt \\ &= \frac{1}{\|d_k\|}\left(\tfrac{1}{3}at^3 + \tfrac{1}{2}bt^2 + ct\right)^{\mathrm{T}} d_k \Big|_0^{\|d_k\|} \\ &= \tfrac{1}{6}\left(3y_k - a\|d_k\|^2 + 6c\right)^{\mathrm{T}} d_k.\end{aligned}$$

Now define

$$\gamma_k = 3(\nabla f(x_k) + \nabla f(x_{k+1}))^{\mathrm{T}} d_k - 6(f(x_{k+1}) - f(x_k)).$$

The simplest choice for a that satisfies $\|d_k\|^2 a^{\mathrm{T}} d_k = \gamma_k$ is $a = \gamma d_k / \|d_k\|^4$. Now we can finally give an approximation for $\nabla^2 f(x_{k+1}) d_k$ in terms of the data available at any given iteration:

$$\begin{aligned}\nabla^2 f(x_{k+1}) d_k = \|d_k\| \frac{d}{dt} \nabla f(x(t))\Big|_{t=\|d_k\|} \approx \|d_k\| \frac{d}{dt} h(t)\Big|_{t=\|d_k\|} &= \|d_k\|(2at + b)\Big|_{t=\|d_k\|} \\ &= 2\|d_k\|^2 a + \|d_k\| b \\ &= y + \|d_k\|^2 a \\ &= y + \frac{\gamma_k}{\|d_k\|^2} d_k.\end{aligned}$$

This leads to a modified quasi-Newton condition $H_{k+1} d_k = \widetilde{y}_k$ with

$$\left.\begin{aligned}\widetilde{y}_k &= y_k + \frac{\gamma_k}{\|d_k\|^2} d_k, \\ \gamma_k &= 3\big(\nabla f(x_k) + \nabla f(x_{k+1})\big)^{\mathrm{T}} d_k - 6\big(f(x_{k+1}) - f(x_k)\big).\end{aligned}\right\} \tag{4.12}$$

Observe that all four data items (4.10) are incorporated by the method if the modified quasi-Newton condition $H_{k+1} d_k = \widetilde{y}_k$ is enforced with $\widetilde{y}_k$ given by (4.12).

One important consideration is preserving the property $y_k^{\mathrm{T}} d_k > 0$. If y_k is scaled by the new condition (4.12) then

$$\widetilde{y}_k^{\mathrm{T}} d_k = \left(y_k + \frac{\gamma_k}{\|d_k\|^2} d_k\right)^{\mathrm{T}} d_k = y_k^{\mathrm{T}} d_k + \gamma_k.$$

For numerical stability, if $y_k^{\mathrm{T}} d_k + \gamma_k < \tilde{\varepsilon}\|d_k\|^2$ then γ_k is set to zero and the unscaled y_k is used for the remainder of the iteration. The value $\tilde{\varepsilon} = 10^{-18}$ proposed by Zhang, Deng and Xu [31] was used in our implementation (see Algorithm `bfgsI`). For additional information, see Zhang and Xu [32] and Xu and Zhang [29].

Algorithm `bfgsI`: Multistep, scaled BFGS.

```
Choose $x_0 \in \mathbb{R}^n$;   $k \leftarrow 0$;
while $\|\nabla f(x_k)\| > \varepsilon$ and $k \le N$ do
    $p_k = -M_k \nabla f(x_k)$;
    $\alpha_k = \texttt{Wolfe}(x_k, f(x_k), \nabla f(x_k), p_k)$;
    $d_k = \alpha_k p_k$;   $x_{k+1} = x_k + d_k$;   $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
    if $y_k^{\mathrm{T}} d_k > 0$ then
        $\gamma_k = 3(\nabla f(x_{k+1}) + \nabla f(x_k))^{\mathrm{T}} - 6(f(x_{k+1}) + f(x_k))$;
        if $y_k^{\mathrm{T}} d_k + \gamma_k < \tilde{\varepsilon}\|d_k\|^2$ then
            $\gamma_k = 0$;
        end if
        $\widetilde{y}_k = y_k + \gamma_k \frac{1}{\|d_k\|^2} d_k$;   $w_k = \frac{1}{\widetilde{y}_k^{\mathrm{T}} d_k} d_k - \frac{1}{\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k} M_k \widetilde{y}_k$;
        $M_{k+1} = M_k - \frac{1}{\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k} M_k \widetilde{y}_k \widetilde{y}_k^{\mathrm{T}} M_k + \frac{1}{\widetilde{y}_k^{\mathrm{T}} d_k} d_k d_k^{\mathrm{T}} + (\widetilde{y}_k^{\mathrm{T}} M_k \widetilde{y}_k) w_k w_k^{\mathrm{T}}$;
    else
        $M_{k+1} = M_k$;
    end if
    $k = k + 1$;
end while
```

5. Numerical Methods

We consider a set $\mathcal{S}$ of $n_{\mathcal{S}}$ solvers run on a problem set $\mathcal{P}$ of $n_{\mathcal{P}}$ problems. In this project $n_{\mathcal{S}} = 9$ and $n_{\mathcal{P}} = 275$. In the interest of complete objectivity, every single unconstrained test problem of dimension $n \in [2,\ 5000]$ available in the CUTEst environment at the time of writing was included and run by each solver. Data of interest is collected for each solver $s \in \mathcal{S}$ as it is run on $\mathcal{P}$. We recorded the number of iterations, the number of function evaluations, the CPU time, and the outcome. The outcome is categorical, defined by specific tolerances (see Section 5.2), and classifies the result as optimal, near optimal but badly scaled, line search failure, or too many iterations. Once the body of data is generated, a systematic method of comparison is needed.

5.1. Performance profiles

One approach is to use the average or cumulative total metric value over the entire problem set. However, the most difficult problems can potentially dominate the results and eliminate the ability to make fine comparisons. Averaging also necessitates discarding problems that were not solved. In this case the failed problem can be removed for all solvers or only for those that failed, both of which bias the results *against* more robust solvers. Another tactic is to rank the solvers, i.e., recording the number of times a solver came in kth place for $k = 1, 2, \ldots, n_{\mathcal{S}}$. This avoids the pitfalls described above, but fails to measure the magnitude of the improvement. In this report we use *performance profiles*, described in [6].

For each $p \in \mathcal{P}$ and $s \in \mathcal{S}$, define $t_{p,s}$ to be the metric value recorded for solver s on problem p. This could be the number of iterations, function evaluations, CPU time, or any other measure of performance. To establish a baseline for comparison, define the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} \mid s \in \mathcal{S}\}},$$

that is, the ratio of the particular solver's metric value against the best value of any solver on this particular problem. This means that $r_{p,s} \geq 1$ and the best possible value is 1. To deal with solve failures, we define

$$r_M = 2\max\{r_{p,s} \mid p \in \mathcal{P}, s \in \mathcal{S}\},$$

and set $r_{p,s} = r_M$ if solver s fails to solve problem p.

In order to measure how each solver does on the entire problem set *relative to* the other solvers, define

$$\rho_s(\tau) = \frac{\left|\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}\right|}{n_{\mathcal{P}}}$$

to be the performance profile for solver s. The function $\rho_s :_R\rightarrow [0,1]$ can be interpreted as follows. For a given value of $\tau \in [1,\ r_M]$, $\rho_s(\tau)$ is the number of problems for which the solver's performance ratio is within a factor of τ of the best possible value (relative to solver set $\mathcal{S}$), out of the total number of problems $n_{\mathcal{P}}$. With this in mind, $\rho_s(\tau)$ behaves like a cumulative distribution function for the solver's performance ratio. The two extremes $\rho_s(1)$ and $\rho_s(r_M)$ give the proportion of problems on which solver s had the best possible value and the proportion of problems solver s was able to solve respectively.

5.2. Hardware and software

The code for each solver was written in MATLAB version R2019b. Each of the solvers makes use of a number of constants and tolerances that determine termination, optimality, unboundedness, line-search failure, etc.. These values are described in Table 1. The data collection, analysis, and plotting software was written

Parameter	**Value**	**Parameter**	**Value**
Stationary tolerance ε	`1.00e-04`	Iteration limit N	`3000`
Gradient tolerance η_W	`9.00e-01`	Maximum Δx	`100`
Function tolerance η_A	`1.00e-04`	Line-search function limit	`20`
Condition number limit	`1.00e+16`		
Unbounded objective	`-1.00e+09`		

Table 1: Constants and tolerances used across all problems and solvers.

in Python using Numpy, Pandas, Matplotlib and Seaborn. All computations were carried out on a 2017 MacBook Pro with a 2.3 GHz Intel Core i5 processor and 8GB 2133 MHz LPDDR3 memory.

5.3. Discussion and results

There is a trade off between the use of iterations and function evaluations, and which one is minimized depends on their relative cost. If iterations are computationally expensive in comparison to function evaluations, then the goal is to use as many function evaluations as needed to get the most out of each iteration. What this means in practice is the tolerances used in the line search are adjusted to demand a greater reduction in the objective value. Conversely, if function evaluations are expensive we accept a smaller decrease in the objective in the interest of limiting the number of evaluations used per iteration. Except for some specific circumstances, the assumption that functions are more expensive than iterations is reasonable. In this project we assume this to be the case and so the function evaluation profiles are the primary tool for comparison. Iteration profiles are also provided to provide additonal context. We use the BFGS method that updates the inverse Hessian approximation (`bfgsM`) as a benchmark with which to compare other methods, because it is commonly regarded as the "gold standard" in optimization literature.

The comparison of `bfgsM` with the factored variation `bfgsR` highlights the importance of numerical stability in the update formula. The fact that the relation ${M_k}^{-1} = R_k^{\mathrm{T}} R_k$ is an *equality* means that in infinite precision arithmetic `bfgsM` and `bfgsR` should perform identically. In practice they do not, and the difference in their performance seen in Figure 1 is owed entirely to the management of numerical factors like conditioning and indefiniteness, made possible by the use of matrix factorization.

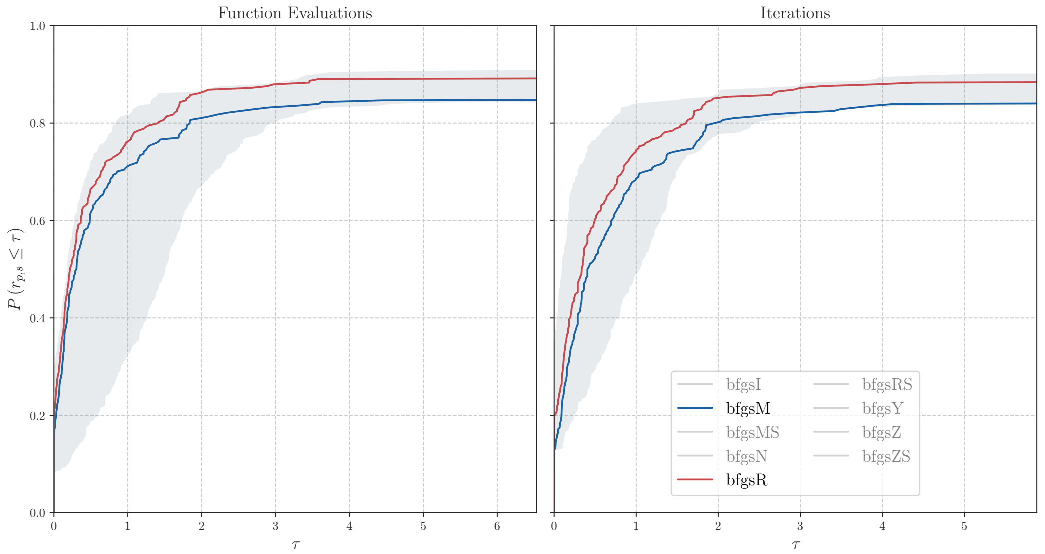

Figure 1: Performance profiles comparing function evaluations and iterations for `bfgsM` and `bfgsR`.

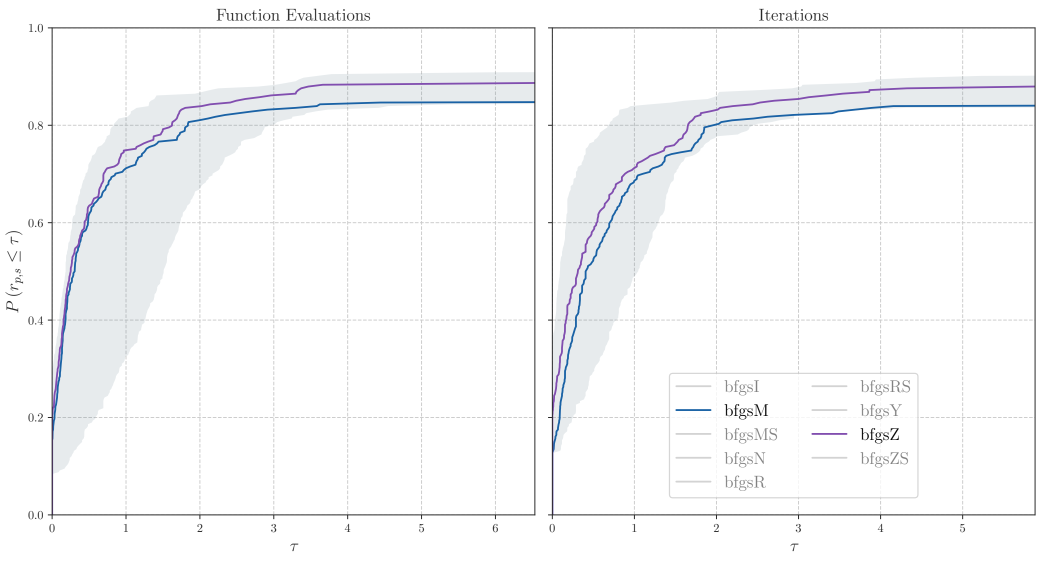

Figure 2: Performance profiles comparing function evaluations and iterations for `bfgsM` and `bfgsZ`.

The self-scaled updates, whose purpose is to limit the negative effects of ill-conditioning, provide another example of the benefits of taking stability into consideration. Whether the inverse, factored inverse, or factored explicit Hessian approximation is used as the base method, self-scaling improves overall performance across the board (see Figures 3, 4, and 5).

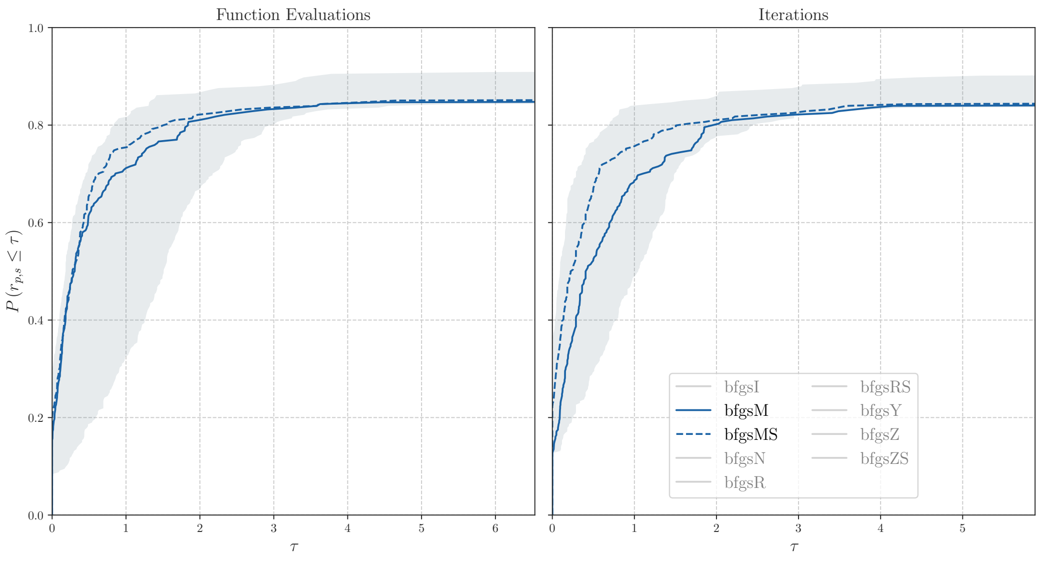

Figure 3: Performance profiles comparing function evaluations and iterations for `bfgsM` and `bfgsMS`.

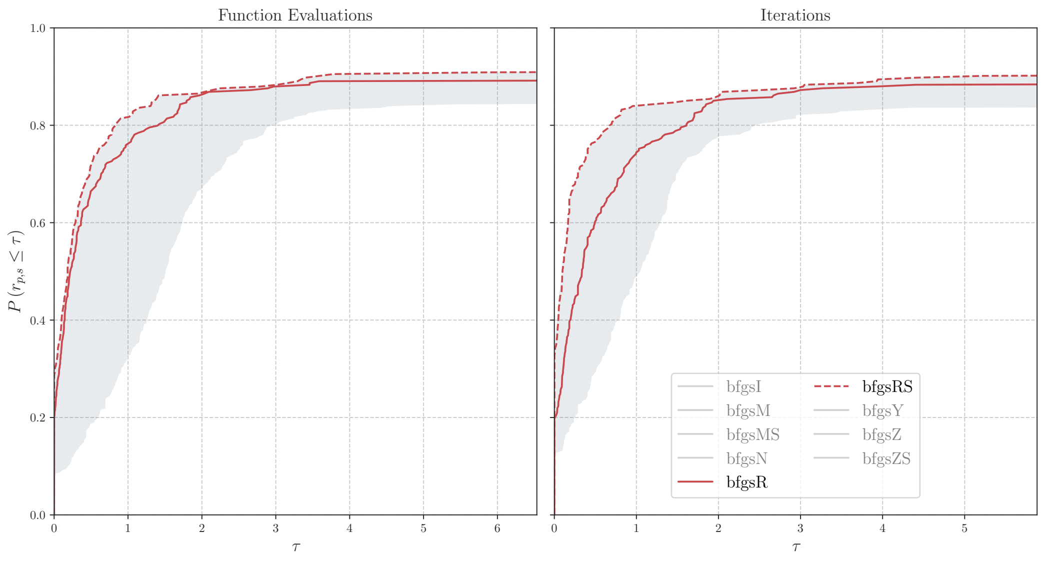

Figure 4: Performance profiles comparing function evaluations and iterations for `bfgsR` and `bfgsRS`.

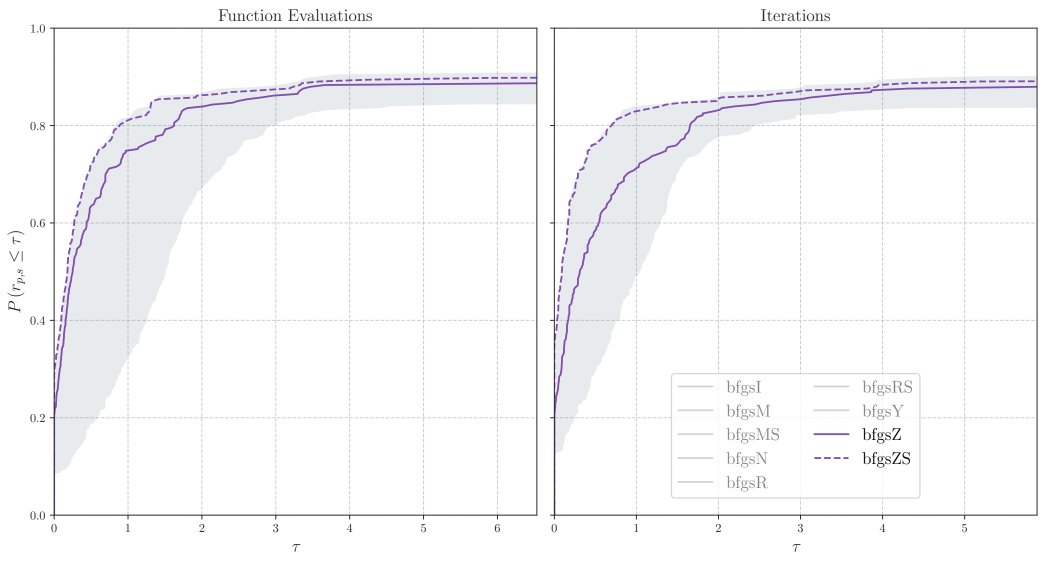

Figure 5: Performance profiles comparing function evaluations and iterations for `bfgsZ` and `bfgsZS`.

It is also worth noting that the improvement seen in factored methods is preserved when self-scaling is applied in `bfgsMS`, `bfgsRS`, and `bfgsZS` (see Figures 6 and 7).

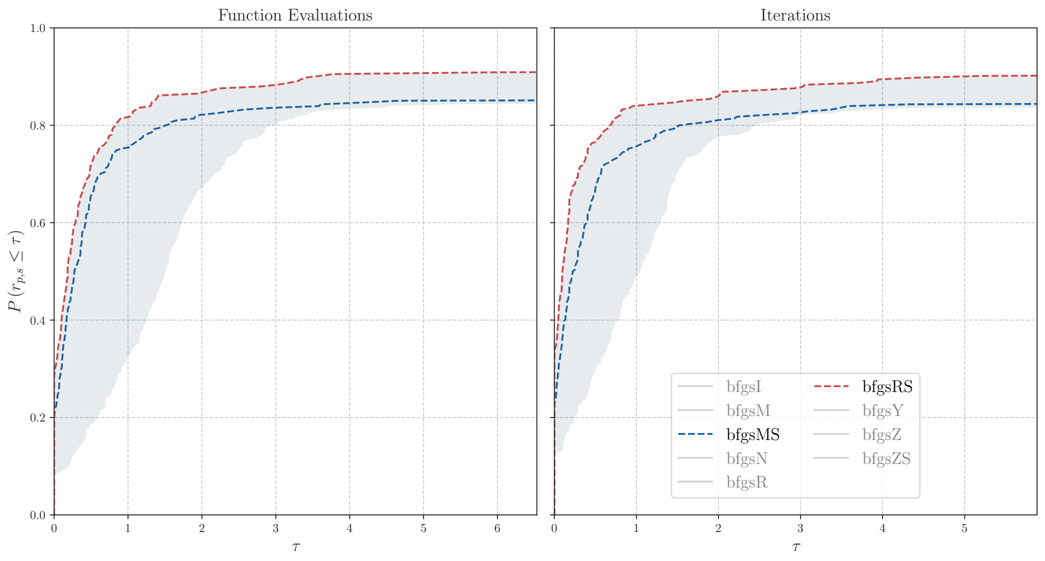

Figure 6: Performance profiles comparing function evaluations and iterations for `bfgsMS` and `bfgsRS`.

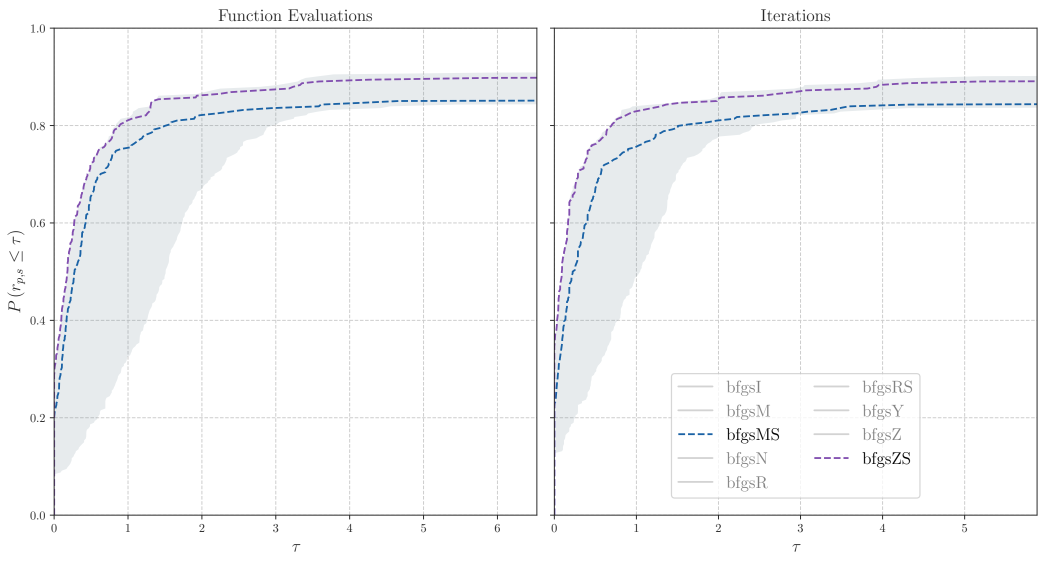

Figure 7: Performance profiles comparing function evaluations and iterations for `bfgsMS` and `bfgsZS`.

Out of all the variants considered in this report, the factored methods `bfgsR`, `bfgsRS`, `bfgsZ`, and `bfgsZS` exhibit the best results, with the overall highest performing method being `bfgsRS` (see Figures 2, 1, and 8). Maintaining a factorization allows us to sidestep the issue of error accumulation resulting in the failure of hereditary positive definiteness. This is immediately apparent because $x^{\mathrm{T}}R^{\mathrm{T}}Rx = \|Rx\|^2 > 0$ for nonzero x and nonsingular R. As `bfgsRS` makes use of both self-scaling to manage conditioning, and resetting if R_k is nearly singular, this supports the idea that numerical stability makes a real difference.

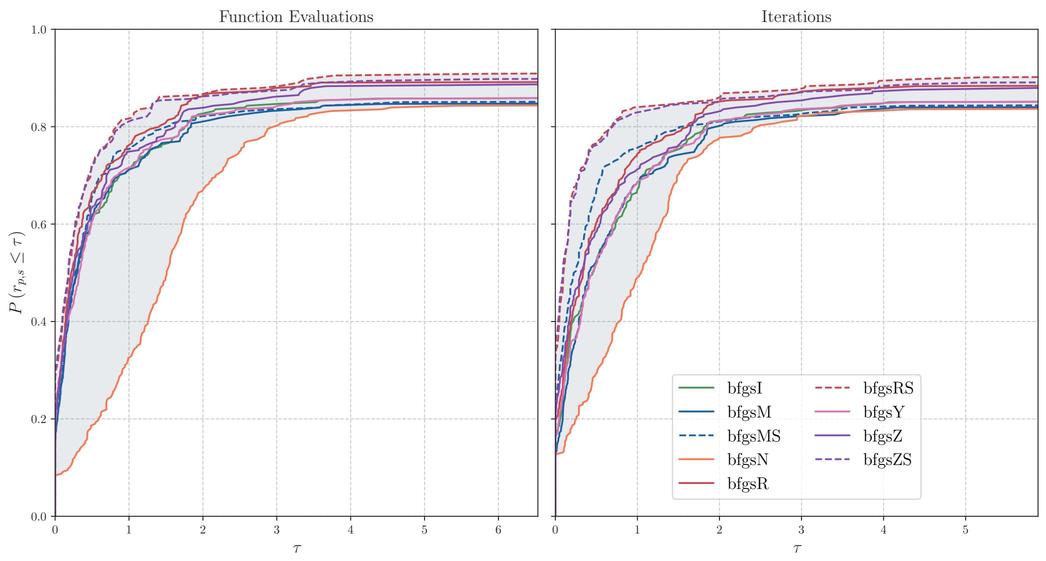

Figure 8: Performance profiles comparing function evaluations and iterations for all solvers.

Methods based on scaling or shifting y_k rather than the Hessian do not appear to be as effective as those based on scaling the Hessian. The more recent variations `bfgsY`, `bfgsN`, and `bfgsI` are all implemented by scaling or shifting y_k. Looking at 9, it seems imposing an extra interpolation condition that is met by scaling y_k seems to have little measurable effect.

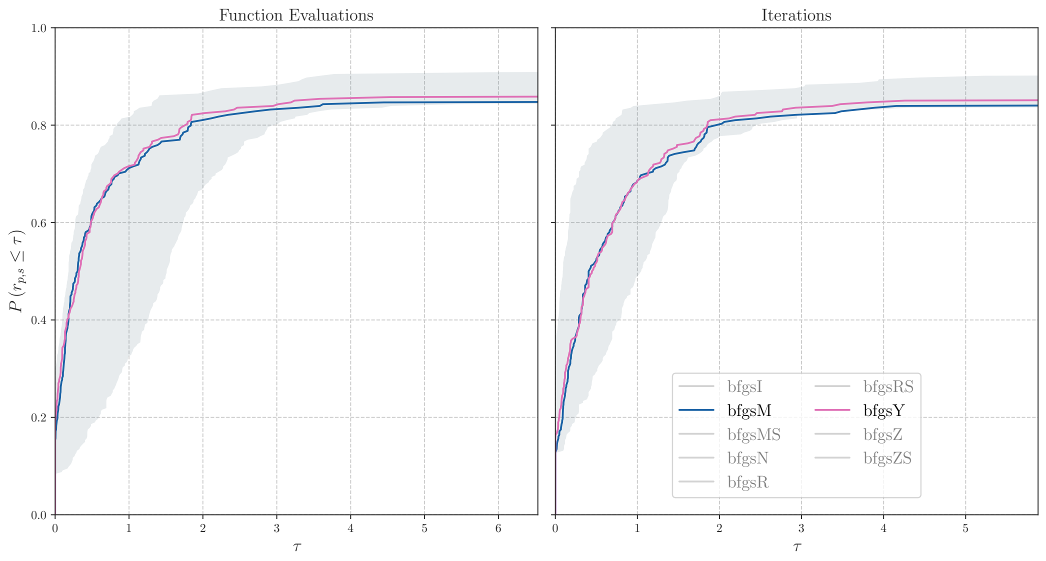

Figure 9: Performance profiles comparing function evaluations and iterations for `bfgsM` and `bfgsY`.

Using the modified quasi-Newton condition and corresponding update also seems to reduce the number of function evaluations and iterations used, but only slightly (see Figure 10).

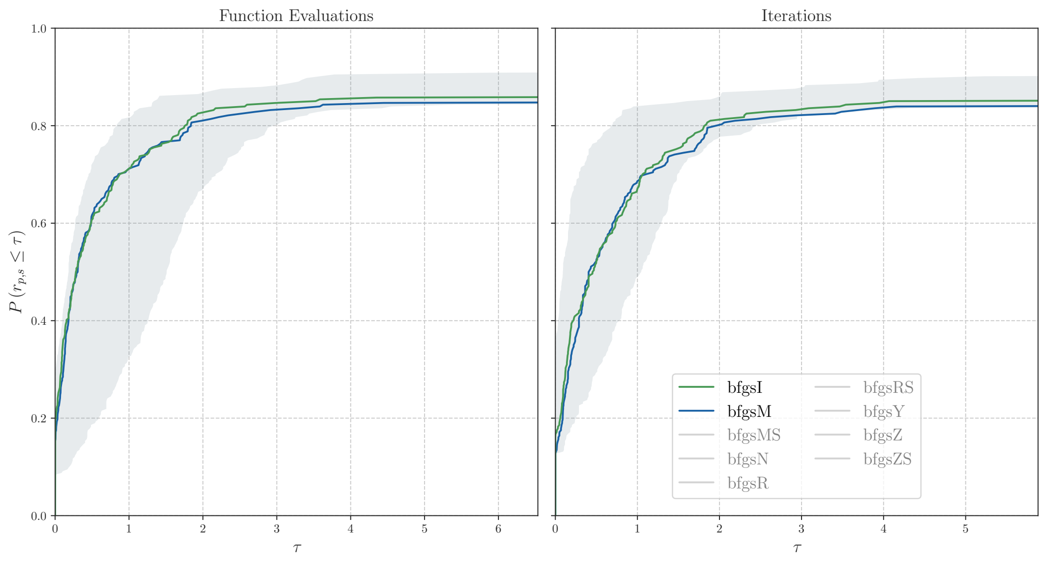

Figure 10: Performance profiles comparing function evaluations and iterations for `bfgsM` and `bfgsI`.

It is important to emphasize the need for uniform analysis and systemic numerical testing in order to draw meaningful conclusions about these algorithms' relative performance. Without this kind of rigorous comparison, results have the potential to be misleading. To illustrate, looking at the references in [1], our problem sets have 38 problems in common. If we restrict our test set to the common problems and consider only the CPU time metric (as in [1]), then `bfgsN` appears to be superior. However, if the test set is expanded to the full 275 problems and the more relevant metrics are measured then it is clear from Figure 11 that the suggested method is actually harmful to the methods performance.

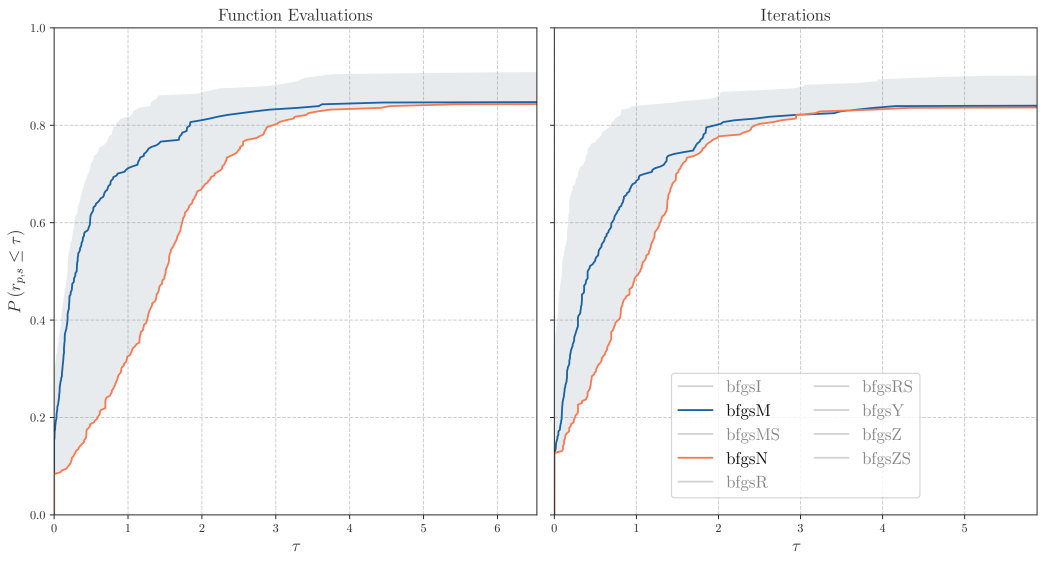

Figure 11: Performance profiles comparing function evaluations and iterations for `bfgsM` and `bfgsN`.

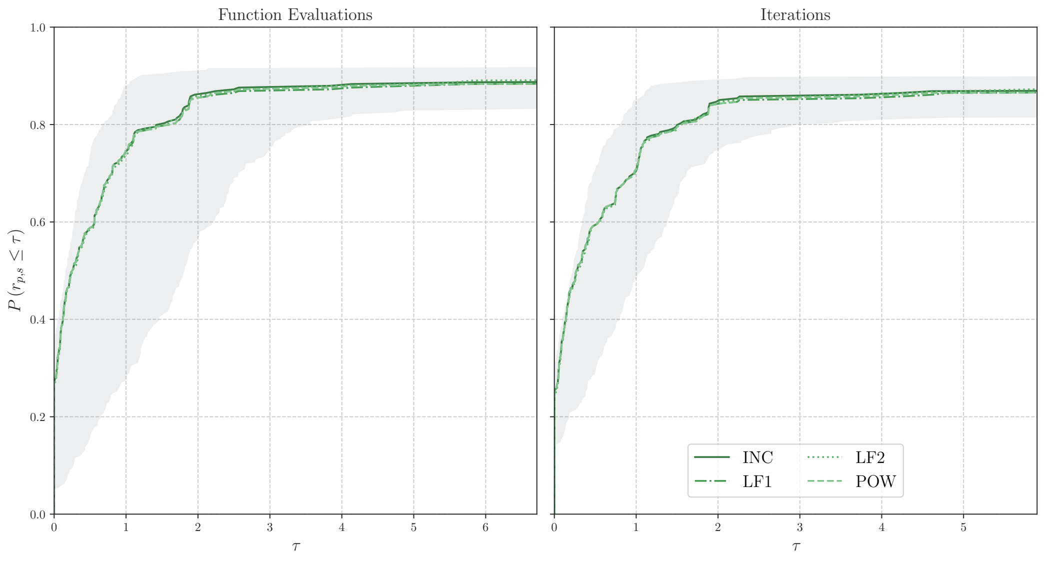

Figure 12: Performance profiles comparing caution levels for `bfgsI`.

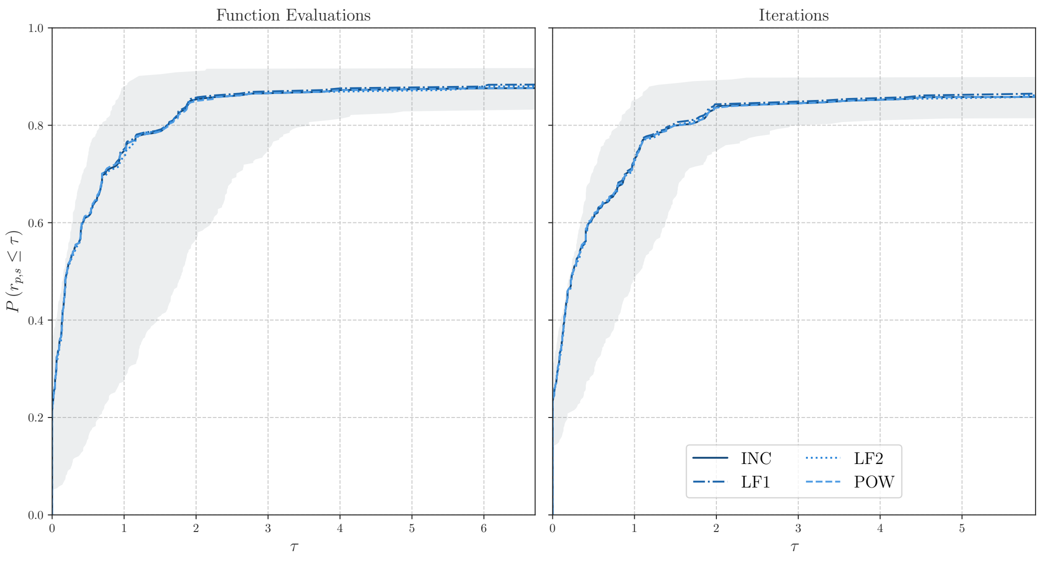

Figure 13: Performance profiles comparing caution levels for `bfgsM`.

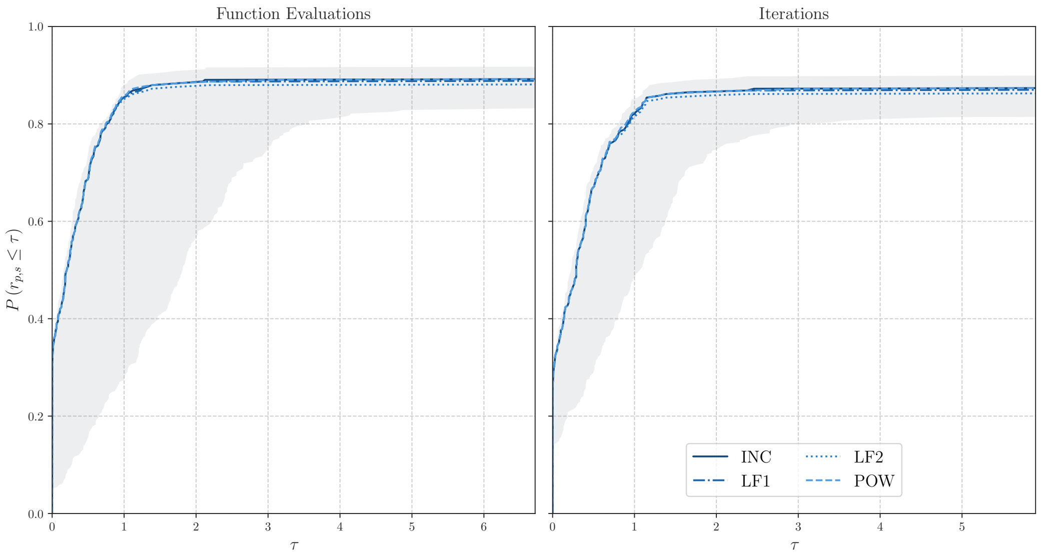

Figure 14: Performance profiles comparing caution levels for `bfgsMS`.

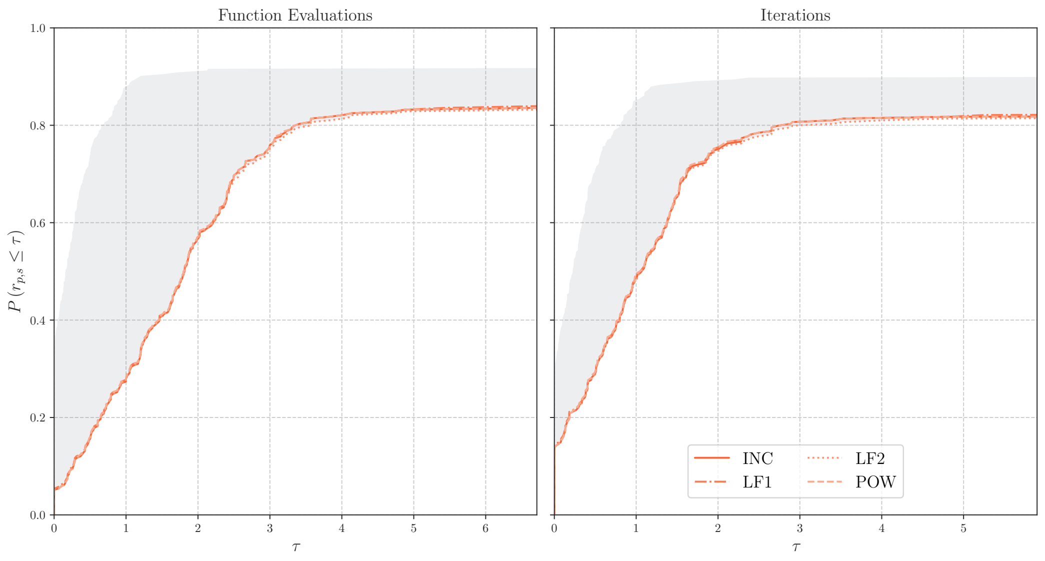

Figure 15: Performance profiles comparing caution levels for `bfgsN`.

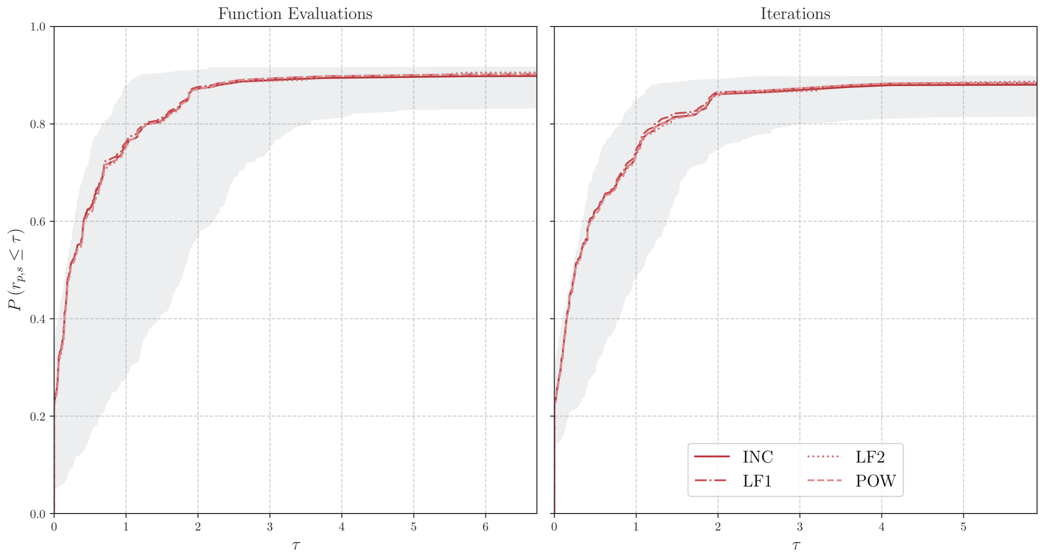

Figure 16: Performance profiles comparing caution levels for `bfgsR`.

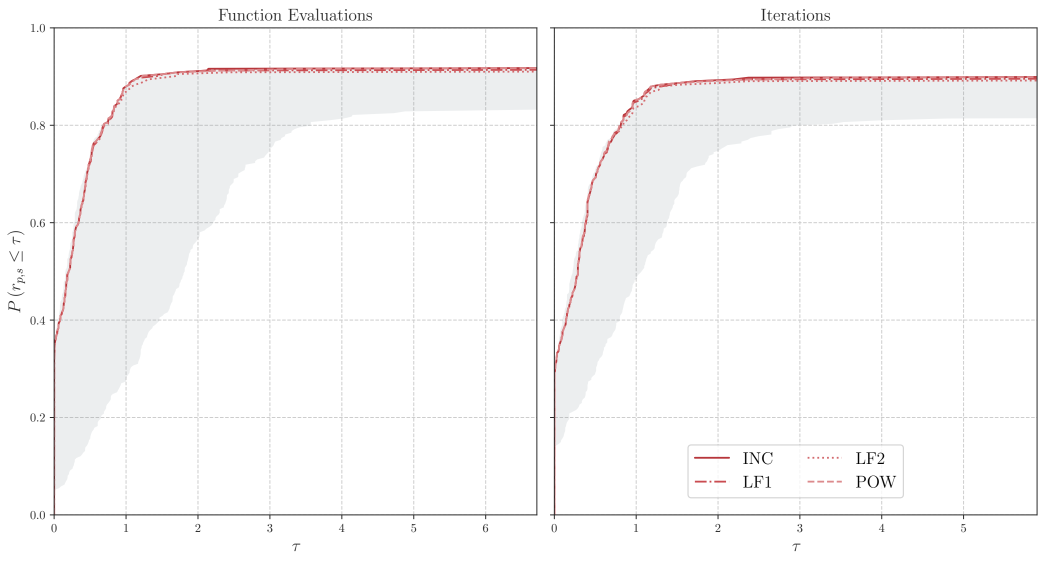

Figure 17: Performance profiles comparing caution levels for `bfgsRS`.

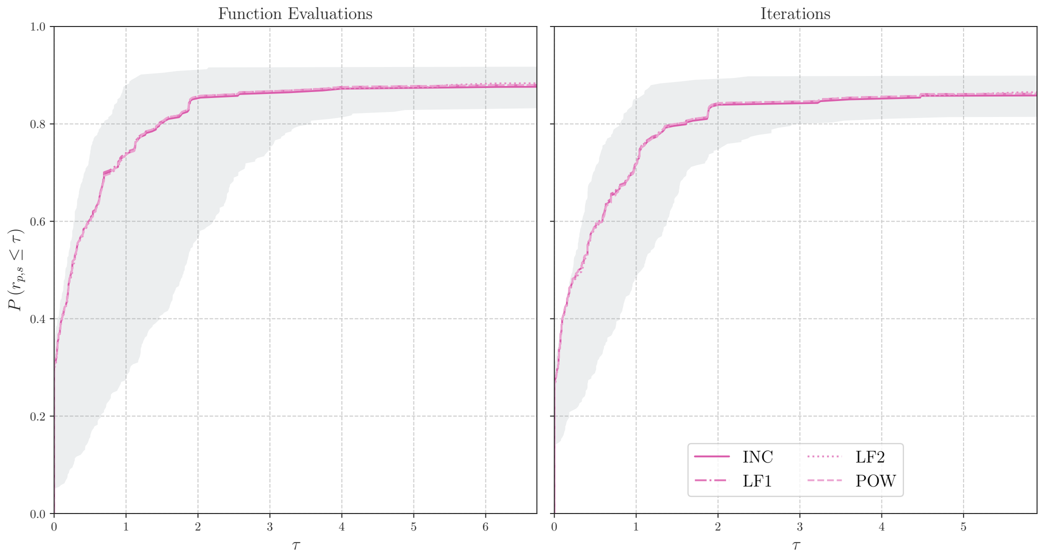

Figure 18: Performance profiles comparing caution levels for `bfgsY`.

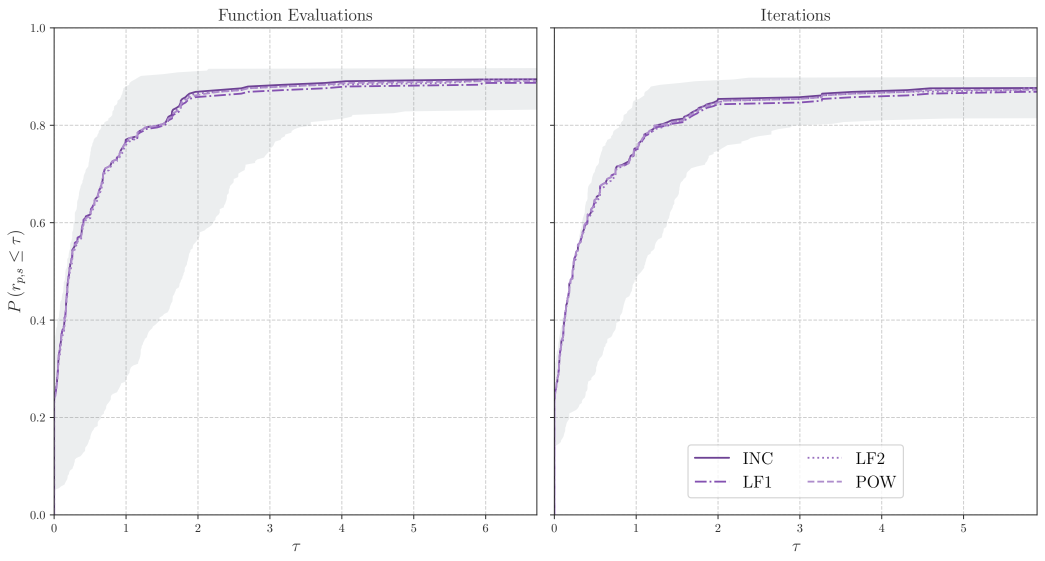

Figure 19: Performance profiles comparing caution levels for `bfgsZ`.

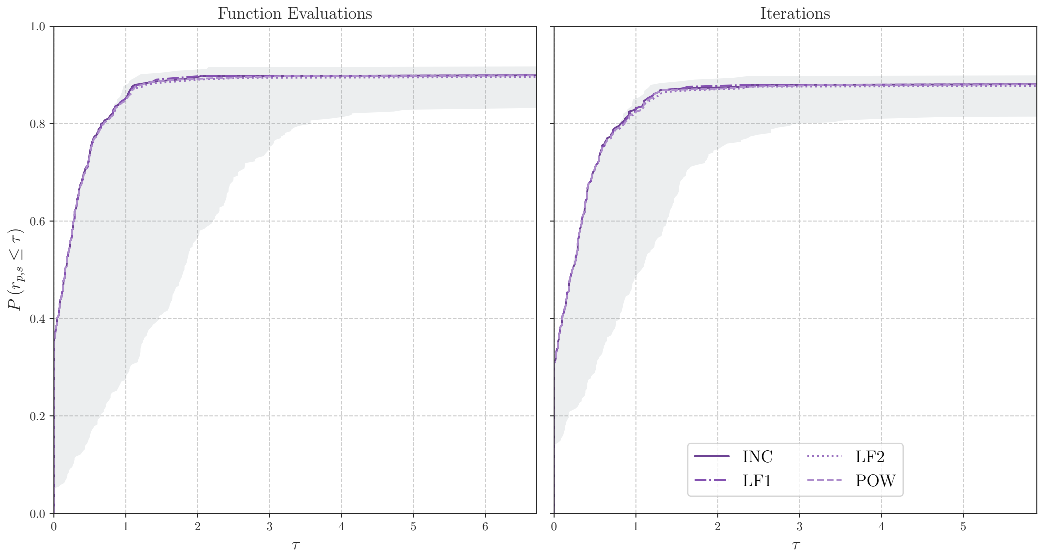

Figure 20: Performance profiles comparing caution levels for `bfgsZS`.

6. Conclusion

In this report we developed the theory, derived the algorithms, and analyzed the performance of variations of quasi-Newton methods for unconstrained optimization.

A systematic and rigorous comparison has revealed that some newer modifications show little or no improvement, while a novel combination of self-scaling and a factored Hessian shows significant and consistent improvement. (Surprisingly, most authors in the optimization community have dismissed factored Hessian methods; see, e.g., Grandinetti [12, 13] Nocedal and Wright [16, p. 201].)

Variations of quasi-Newton methods that focus on managing the condition number and preserving positive definiteness result in a marked improvement in function evaluations and iterations. In contrast, those based on imposing additional conditions and then scaling the update to satisfy them seem to have little or no measurable effect. These results server as a reminder that numerical stability must be taken into account in numerical optimization. A method may have wonderful theoretical properties, but if it fails to deal with numerical issues these properties simply cannot be realized.

References

[1] Neculai Andrei. An adaptive scaled BFGS method for unconstrained optimization. *Numerical Algorithms*, 77(2):413–432, 2018. 20, 32

[2] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966. 3

[3] K. W. Brodlie, A. R. Gourlay, and J. Greenstadt. Rank-one and rank-two corrections to positive definite matrices expressed in product form. *J. Inst. Math. Appl.*, 11:73–82, 1973. 8

[4] Richard H Byrd, Jorge Nocedal, and Ya-Xiang Yuan. Global convergence of a class of quasi-newton methods on convex problems. *SIAM J. Numer. Anal.*, 24:1171–1190, 1987. 18

[5] William C. Davidon. Variable metric methods for minimization. Report ANL-5990, A.E.C. Research and Development, Argonne National Laboratory, Argonne, IL, 1959. 2

[6] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2, Ser. A):201–213, 2002. 25

[7] Roger Fletcher and Michael J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168, 1963. 2

[8] Philip E. Gill, Gene H. Golub, Walter Murray, and Michael A. Saunders. Methods for modifying matrix factorizations. *Math. Comput.*, 28:505–535, 1974. 10, 12

[9] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. A note on a sufficient-decrease criterion for a nonderivative step-length procedure. *Math. Programming*, 23(3):349–352, 1982. 3

[10] Donald Goldfarb. Factorized variable metric methods for unconstrained optimization. *Math. Comput.*, 30:796–811, 1976. 12

[11] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.*, 60(3):545–557, 2015. 4

[12] Lucio Grandinetti. Factorization versus non-factorization in quasi-newtonian algorithms for differentiable optimization. In W. Oettli and F. Steffens, editors, *Third Symposium on Operations Research (Univ. Mannheim, Mannheim, 1978), Section I, Operations Res. Verfahren, 31*, pages 255–274, Königstein/Ts, 1979. Hain. 37

[13] Lucio Grandinetti. Factorized variable metric algorithms for unconstrained optimization. In K. Iracki, K. Malanowski, and S. Walukiewicz, editors, *Optimization Techniques (Proc. 9th IFIP Conf., Warsaw, September 4-8, 1979), Part 2*, Lecture Notes in Control and Information Sciences, pages 52–61, Berlin, Heidelberg, 1980. Springer Berlin Heidelberg. 37

[14] Dong-Hui Li and Masao Fukushima. On the global convergence of the bfgs method for non-convex unconstrained optimization problems. *SIAM J. Optim.*, 11:1054–1064, 2001. 18

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

[16] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer-Verlag, New York, 1999. 17, 37

[17] Shmuel S. Oren. Self-scaling variable metric algorithms without line search for unconstrained optimization. *Math. Comp.*, 27(124):873–885, October 1973. 5

[18] Shmuel S. Oren. On the selection of parameters in self-scaling variable metric algorithms. *Math. Program.*, 7:351–367, 1974. 5

[19] Shmuel S. Oren. Self-scaling variable metric (SSVM) algorithms, Part ii: implementation and experiments. *Management Science*, 20:863–874, 1974. 5

[20] Shmuel S. Oren and David G. Luenberger. Self-scaling variable metric (SSVM) algorithms, Part I: Criteria and sufficient conditions for scaling a class of algorithms. *Management Science*, 20:845–862, 1974. 5

[21] Shmuel S. Oren and Emilio Spedicato. Optimal conditioning of self-scaling variable metric algorithms. *Math. Program.*, 10:70–90, 1976. 5

[22] James M. Ortega and Werner C. Rheinboldt. *Iterative solution of nonlinear equations in several variables.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original. 3

[23] Michael J. D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In R. W. Cottle and C. E. Lemke, editors, *SIAM-AMS Proceedings*, volume IX, Philadelphia, 1976. SIAM Publications. 18

[24] Michael J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, editor, *Numerical Analysis, Dundee 1977*, number 630 in Lecture Notes in Mathematics, pages 144–157, Heidelberg, Berlin, New York, 1978. Springer Verlag. 17

[25] Michael J. D. Powell. Updating conjugate directions by the BFGS formula. *Math. Program.*, 38:693–726, 1987. 15, 17

[26] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Ann. Math. Statist.*, 20:621–625, 1949. 3

[27] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Rev.*, 11:226–235, 1969. 3

[28] Max A. Woodbury. Inverting modified matrices. Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950. 3

[29] Chengxian Xu and Jianzhong Zhang. A survey of quasi-Newton equations and quasi-Newton methods for optimization. *Ann. Oper. Res.*, 103(1–4):213–234, 2001. 24

[30] Ya-Xiang Yuan. A modified BFGS algorithm for unconstrained optimization. *IMA J. Numer. Anal.*, 11(3):325–332, 1991. 19

[31] J. Z. Zhang, N. Y. Deng, and L. H. Chen. New quasi-Newton equation and related methods for unconstrained optimization. *Journal of Optimization Theory and Applications*, 102(1):147–167, 1999. 23, 24

[32] Jianzhong Zhang and Chengxian Xu. Properties and numerical performance of quasi-newton methods with modified quasi-newton equations. *Journal of Computational and Applied Mathematics*, 137:269–278, 2001. 24