

On the BFGS Method for Unconstrained Optimization

Philip E. Gill*

Jeb H. Runnoe*

UCSD Center for Computational Mathematics

Technical Report CCoM-26-1

October 1, 2026

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. An extensive survey and evaluation of the most prominent of these modifications was given by Gill and Runnoe [11]. This paper focuses on two of the most effective approaches considered in the survey and give extensive results from implementations evaluated on a set of unconstrained problems from the CUTEst test collection.

It is shown that a combination of self-scaling and a factored Hessian shows significant and consistent improvement over the conventional BFGS method. The results appear to contradict the widely-held view that self-scaled updates and stable factorization methods provide no additional benefits over a conventional implementation.

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

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

1. Introduction

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

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

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

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

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

*Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (pgill@ucsd.edu, jebrunroe@gmail.com) 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.

x_k . The first quasi-Newton method for the unconstrained minimization problem was proposed in 1959 by Davidon [4]. This method was modified and clarified by Fletcher and Powell [8], 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 $y_k^T d_k$ is a first-order estimate of the curvature $d_k^T \nabla^2 f(x_k) d_k$ and is known as the *approximate curvature*. The matrix H_{k+1} is chosen to satisfy the so-called quasi-Newton condition:

$$H_{k+1} d_k = y_k. \quad (1.2)$$

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

$$d_k^T H_{k+1} d_k = y_k^T d_k. \quad (1.3)$$

The quasi-Newton condition *installs* the approximate curvature $y_k^T d_k$ 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 $f(x_k) + \nabla f(x_k)^T (x - x_k)$. The sufficient-decrease condition may be written as

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k \eta_A \nabla f(x_k)^T p_k, \quad (1.4)$$

(see, e.g., Armijo [1], Ortega and Rheinboldt [22]). Every method considered in this paper satisfies the Armijo condition in conjunction with the strong Wolfe condition

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq \eta_W |\nabla f(x_k)^T p_k|, \quad (1.5)$$

where η_W is a preassigned scalar such that $\eta_W \in (\eta_A, 1)$. (See, e.g., Wolfe [26], Moré and Thuente [15], and Gill et al. [10]).

The quasi-Newton condition (1.2) and identity (1.3) imply that $y_k^T d_k > 0$ 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 (1.5). In particular, for any step satisfying (1.5) it holds that

$$\begin{aligned} y_k^T p_k &= \nabla f(x_{k+1})^T p_k - \nabla f(x_k)^T p_k \\ &\geq \eta_W \nabla f(x_k)^T p_k - \nabla f(x_k)^T p_k \geq (1 - \eta_W) p_k^T H_k p_k. \end{aligned} \quad (1.6)$$

This implies that any α_k satisfying the Wolfe condition (1.5) will give an x_{k+1} such that $y_k^T d_k \geq (1 - \eta_W) \alpha_k p_k^T H_k p_k > 0$.

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

$$H_{k+1} = H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \frac{1}{y_k^T d_k} y_k y_k^T, \quad (1.7)$$

which represents a symmetric, rank-two modification to H_k . The BFGS update has the property of hereditary positive definiteness if $y_k^T d_k > 0$.

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 [25], and Woodbury [27]). If H_k is updated by the BFGS formula (1.7), the associated update to the inverse is given by

$$M_{k+1} = M_k - \frac{1}{y_k^T d_k} (M_k y_k d_k^T + d_k y_k^T M_k) + \frac{y_k^T d_k + y_k^T M_k y_k}{(y_k^T d_k)^2} y_k y_k^T. \quad (1.8)$$

The BFGS update to the inverse may be written in a number of different ways. The symmetrization of the cross-product term of (1.8) gives the update

$$M_{k+1} = M_k - \frac{1}{y_k^T M_k y_k} M_k y_k y_k^T M_k + \frac{1}{y_k^T d_k} d_k d_k^T + (y_k^T M_k y_k) \left(\frac{1}{y_k^T d_k} d_k - \frac{1}{y_k^T M_k y_k} M_k y_k \right) \left(\frac{1}{y_k^T d_k} d_k - \frac{1}{y_k^T M_k y_k} M_k y_k \right)^T. \quad (1.9)$$

In subsequent sections, a line-search method with the BFGS update (1.8) or (1.9) will be referred to as Algorithm **bfgsM**.

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

Overview. The BFGS method is widely perceived to be the best of the quasi-Newton methods proposed in the ten years following the formulation of the DFP method. However, 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. 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. An extensive survey and evaluation of the most prominent of these modifications was given by Gill and Runnøe [11]. The goal of this paper is to implement and test these methods in a uniform, systematic, and consistent way. This paper has focused on two of the most effective approaches considered in the survey. The methods are variations of the BFGS algorithm, and are based on either the standard update formula (1.7) or the inverse update formula (1.8). These methods fall under two broad categories: methods that attempt to improve the condition number of the approximate Hessian; and methods that are intended to improve the overall numerical stability of a given update. The computational performance of each of the methods is evaluated on a set of unconstrained problems from the CUTEst test collection (see Gould, Orban and Toint [13]).

Notation. Given vectors or scalars 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 i th component of that vector, whereas the subscript k is appended to a vector to denote its value during the k th iteration of an algorithm, e.g., x_k represents the value for x during the k th iteration, whereas $[x_k]_i$ denotes the i th 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 to denote 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) and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ the change in gradient. The matrix H_k denotes the approximate Hessian and M_k is the approximate inverse Hessian.

2. The Self-Scaled BFGS Method

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 methods rescale 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 to first order, the spread of the spectrum of $\nabla^2 f(x_k)^{-1}(\gamma_k H_k)$ includes 1. Self-scaled updates were first proposed by Oren and Luenberger [20]. Additional references include Oren [17–19], and Oren and Spedicato [21],

If H_k is replaced by $\gamma_k H_k$ in the BFGS update (1.7), then

$$H_{k+1} = \gamma_k \left(H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k \right) + \frac{1}{y_k^T d_k} y_k y_k^T. \quad (2.1)$$

A suitable value of γ_k may be derived by considering the BFGS method for minimizing the quadratic $c^T x + \frac{1}{2} x^T H x$. In this case, if the eigenvalues of $H^{-1} H_k$ are $0 < \lambda_n \leq \lambda_{n-1} \leq \dots \leq \lambda_1$, then any γ_k such that $1/\lambda_1 \leq \gamma_k \leq 1/\lambda_n$ will give

$$0 < \gamma_k \lambda_n \leq 1 \leq \gamma_k \lambda_1 \quad \text{and} \quad 1 \in [\gamma_k \lambda_n, \gamma_k \lambda_1]. \quad (2.2)$$

In the quadratic case, consider the matrix S_k and vector s_k such that

$$S_k = H^{-1/2} H_k H^{-1/2}, \quad s_k = H^{1/2} d_k,$$

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

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

It follows that in the nonlinear case, if $\gamma_k = y_k^T d_k / d_k^T H_k d_k$ is used in (2.1), then (2.2) 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

$$\begin{aligned} M_{k+1} = & \gamma_k \left(M_k - \frac{1}{y_k^T M_k y_k} M_k y_k y_k^T M_k \right) + \frac{1}{y_k^T d_k} d_k d_k^T \\ & + \gamma_k (y_k^T M_k y_k) \left(\frac{1}{y_k^T d_k} d_k - \frac{1}{y_k^T M_k y_k} M_k y_k \right) \left(\frac{1}{y_k^T d_k} d_k - \frac{1}{y_k^T M_k y_k} M_k y_k \right)^T. \end{aligned} \quad (2.3)$$

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 inverse BFGS update (2.3) will be referred to as **bfgsMS**.

3. Factored BFGS Methods

A significant disadvantage of using 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 be descent directions, ultimately leading to the failure of the line search. In this section a variant of the BFGS method is derived 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.

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 $R_k^T R_k$ can never be indefinite. Moreover, a lower bound on the condition number of H_k is readily available from the identity $\text{cond}(H_k) = \text{cond}(R_k)^2$ and the bound $\text{cond}(R_k) \geq \max |r_{jj}| / \min |r_{jj}|$. It follows that one possible strategy is to reset the (implicit) approximate Hessian H_k if $\text{cond}(R_k)^2$ becomes large.

Methods for updating matrix factorizations are central in the formulation of modern numerically stable optimization methods. Given the Cholesky factorization $H_k = R_k^T R_k$, the equations $H_k p_k = -\nabla f(x_k)$ may be written as $R_k^T R_k p_k = -\nabla f(x_k)$. These equations may be solved using forward and backward substitution on the triangular systems

$$R_k^T q_k = -\nabla f(x_k) \quad \text{and} \quad R_k p_k = q_k. \quad (3.1)$$

A line-search method based on updating the Cholesky factorization will be referred to as **bfgsR**.

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

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

an upper-triangular matrix R_{k+1} is sought such that $H_{k+1} = R_{k+1}^T R_{k+1}$. This matrix may be found by utilizing an alternative form for the BFGS update (1.7). The matrix H_{k+1} may be written in the form

$$H_{k+1} = (I + v_k d_k^T) H_k (I + d_k v_k^T), \quad \text{with} \quad v_k = \pm \frac{1}{(y_k^T d_k)^{1/2} (d_k^T H_k d_k)^{1/2}} y_k - \frac{1}{d_k^T H_k d_k} H_k d_k, \quad (3.2)$$

(see Osborne and Saunders [23], Davidon [5], and Brodlie, Gourlay and Greenstadt [3]). If the factorization $H_k = R_k^T R_k$ is known at the k th iteration, then (3.2) may be used to update R_k via

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

with $\hat{R}_k = R_k (I - d_k v_k^T)$ (see Dennis and Schnabel [6]). Substituting the expression for v_k and expanding gives

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

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

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

or simply $\hat{R}_k = R_k + u_k v_k^T$ with $u_k = R_k p_k / \|R_k p_k\| = q_k / \|q_k\|$ (cf (3.1)) and $v_k = \pm y_k / (y_k^T d_k)^{1/2} - R_k^T u_k$. The magnitudes of intermediate quantities are minimized if the positive sign is used if $y_k^T H_k d_k < 0$ (i.e., $y_k^T \nabla f(x_k) < 0$), and the negative sign is used otherwise.

The matrix $\hat{R}_k = R_k + u_k v_k^T$ has the desired property that $\hat{R}_k^T \hat{R}_k = H_{k+1}$, however it is *not* upper triangular and therefore not a Cholesky factor of H_{k+1} . The principal effort associated with the update procedure is devoted to obtaining a triangular R_{k+1} from $\hat{R}_k$. The numerical results of Section 5 were obtained using the updating method proposed by Goldfarb [12] that is based on the special properties of products of plain rotations considered in Gill et al. [9].

If Q_k is an orthogonal matrix with $Q_k^T Q_k = I$, then

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

The idea is to choose the orthogonal matrix Q_k so that $R_{k+1} = Q_k \hat{R}_k$ is upper triangular. The desired effect is achieved with two sets of 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.

Given any nonzero $z \in \mathbb{R}^n$ and indices $1 \leq i < j \leq n$, it is always possible to find an orthogonal plane rotation P_j^i so that $(P_j^i z)_j = 0$ and $(P_j^i z)_i = \pm \rho$, where $\rho = \sqrt{z_i^2 + z_j^2}$. This matrix is formed by embedding the two-by-two matrix

$$\begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix} \quad \text{such that} \quad c_j = \frac{z_i}{\rho} \quad \text{and} \quad s_j = \frac{z_j}{\rho}$$

in the i th and j th rows and columns of the identity matrix I_n . It follows that the product $P = P_2^1 P_3^2 \cdots P_n^{n-1}$ satisfies $Pz = \pm \rho e_1$ with $\rho = \|z\|$. Gill et al. [9] show that this product is upper-Hessenberg with the special form

$$P = \begin{pmatrix} \bar{\beta}_1 z_1 & \bar{\beta}_1 z_2 & \bar{\beta}_1 z_3 & \cdots & \bar{\beta}_1 z_n \\ -s_1 & \bar{\beta}_2 z_2 & \bar{\beta}_2 z_3 & \cdots & \bar{\beta}_2 z_n \\ & -s_2 & \bar{\beta}_3 z_3 & \cdots & \bar{\beta}_3 z_n \\ & & \ddots & \ddots & \vdots \\ & & & -s_{n-1} & \bar{\beta}_n z_n \end{pmatrix}.$$

The vectors $s = (s_1, s_2, \dots, s_{n-1})$ and $\bar{\beta} = (\bar{\beta}_1, \bar{\beta}_2, \dots, \bar{\beta}_n)$ needed to define P can be computed efficiently using the following algorithm.

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

```

procedure recurrence1( $z$ )
  Set  $\bar{\beta}_n = 1/z_n$ ;
  for  $j = n-1, n-2, \dots, 1$  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  $(\bar{\beta}, s)$ ;
end procedure

```

Returning to the rank-one update $\hat{R} = R + uv^T$, let $\hat{v}$ denote the solution of $R^T \hat{v} = v$. It follows that $\hat{R} = (I + u\hat{v}^T)R$ and Algorithm 1 may be used to compute an orthogonal product of plane rotations Q_1 , chosen so that $Q_1 u = \pm e_1$. Then premultiplying $\hat{R}$ by Q_1 gives

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

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

$$H_u = \begin{pmatrix} \bar{\beta}_1 u_1 + \bar{\sigma}_1 \hat{v}_1 & \bar{\beta}_1 u_2 + \bar{\sigma}_1 \hat{v}_2 & \bar{\beta}_1 u_3 + \bar{\sigma}_1 \hat{v}_3 & \cdots & \bar{\beta}_1 u_n + \bar{\sigma}_1 \hat{v}_n \\ -s_1 & \bar{\beta}_2 u_2 & \bar{\beta}_2 u_3 & \cdots & \bar{\beta}_2 u_n \\ & -s_2 & \bar{\beta}_3 u_3 & \cdots & \bar{\beta}_3 u_n \\ & & \ddots & \ddots & \vdots \\ & & & -s_{n-1} & \bar{\beta}_n u_n \end{pmatrix}. \quad (3.3)$$

The next step is to compute a second sequence of plane rotations Q_2 that eliminate the sub-diagonal elements of H_u . The resulting matrix $\tilde{R} = Q_2 H_u$ will then be upper triangular, with

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

The structure of $\tilde{R}$ facilitates the computation of λ , β , and σ using recurrence relations. Let $\bar{\lambda}_j = \bar{\beta}_j u_j + \bar{\sigma}_j \hat{v}_j$, and consider the two-by-two product

$$\begin{pmatrix} \bar{c}_j & -\bar{s}_j \\ \bar{s}_j & \bar{c}_j \end{pmatrix} \begin{pmatrix} \bar{\lambda}_j & \bar{\beta}_j u_{j+1} + \bar{\sigma}_j \hat{v}_{j+1} \\ -s_j & \bar{\beta}_{j+1} u_{j+1} \end{pmatrix} = \begin{pmatrix} \bar{c}_j \bar{\lambda}_j + \bar{s}_j s_j & (\bar{c}_j \bar{\beta}_j - \bar{s}_j \bar{\beta}_{j+1})u_{j+1} + \bar{c}_j \bar{\sigma}_j \hat{v}_{j+1} \\ \bar{s}_j \bar{\lambda}_j - \bar{c}_j s_j & (\bar{s}_j \bar{\beta}_j + \bar{c}_j \bar{\beta}_{j+1})u_{j+1} + \bar{s}_j \bar{\sigma}_j \hat{v}_{j+1} \end{pmatrix}. \quad (3.4)$$

If the nonzero sub-diagonal of (3.3) is to be removed, then $\bar{s}_j \bar{\lambda}_j - \bar{c}_j s_j = 0$. It is also apparent from (3.4) 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 setting $\bar{c}_j = \bar{\lambda}_j/\lambda_j$ and $\bar{s}_j = s_j/\lambda_j$. The values of β_j , σ_j are the coefficients of u_{j+1} and $\hat{v}_{j+1}$ in the first row of the right side of (3.4), 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 determined from the product (3.4) in a similar way. The entire procedure for computing Q_2 and forming $\tilde{R} = Q_2 H_u$ is accomplished by the following algorithm.

Algorithm 2. Compute $\tilde{R} = Q_2 H_u$.

```

procedure recurrence2( $\bar{\beta}, s, u, \hat{v}$ )
  Set  $\bar{\sigma}_1 = 1/\bar{\beta}_1$ ;  $\bar{\lambda}_1 = \bar{\beta}_1 u_1 + \bar{\sigma}_1 \hat{v}_1$ ;
  for  $j = 1, 2, \dots, n-1$  do
     $\lambda_j = (\bar{\lambda}_j^2 + s_j^2)^{1/2}$ ;
     $\bar{c}_j = \bar{\lambda}_j/\lambda_j$ ;  $\bar{s}_j = s_j/\lambda_j$ ;
     $\beta_j = \bar{c}_j \bar{\beta}_j - \bar{s}_j \bar{\beta}_{j+1}$ ;  $\sigma_j = \bar{c}_j \bar{\sigma}_j$ ;
     $\bar{\beta}_{j+1} = \bar{s}_j \bar{\beta}_j + \bar{c}_j \bar{\beta}_{j+1}$ ;  $\bar{\sigma}_{j+1} = \bar{s}_j \bar{\sigma}_j$ ;  $\bar{\lambda}_{j+1} = \bar{\beta}_{j+1} u_{j+1} + \bar{\sigma}_{j+1} \hat{v}_{j+1}$ ;
  end for
  return ( $\lambda, \beta, \sigma$ );
end procedure

```

The product $R^+ = \tilde{R}R$ may now be formed efficiently. The following algorithm is analogous to the column update proposed by Goldfarb [12], and may also be viewed as a modification of the backward recurrence of Lemma V given in Gill et al. [9].

Algorithm 3. Compute $R^+ = \tilde{R}R$.

```

procedure recurrence3( $R, \beta, u, \sigma, \hat{v}, \lambda$ )
  for  $i = n, n-1, \dots, 1$  do
     $\tilde{u} = u_i r_{ii}$ ;  $\tilde{v} = \hat{v}_i r_{ii}$ ;
    for  $j = i+1, i+2, \dots, n$  do
       $r_{ij}^+ = \lambda_i r_{ij} + \beta_i \tilde{u}_j + \sigma_i \tilde{v}_j$ ;
       $\tilde{u}_j = \tilde{u}_j + u_j r_{ij}$ ;  $\tilde{v}_j = \tilde{v}_j + \hat{v}_j r_{ij}$ ;
    end for
  end for
  return ( $R^+$ );
end procedure

```

Given the upper-triangular factor $R^+ = \tilde{R}R$ and the orthogonal matrix $Q = Q_2 Q_1$, it holds that

$$R^{+\top} R^+ = (\tilde{R}R)^\top \tilde{R}R = R^\top H_u^\top Q_2^\top Q_2 H_u R = \hat{R}^\top Q_1^\top Q_2^\top Q_2 Q_1 \hat{R} = \hat{R}^\top \hat{R} = H^+,$$

which implies that R^+ is the required Cholesky factor for the next iteration. The total work required for updating the Cholesky factors is $6n^2 + 30n - 20$ floating-point operations, which provides an overall reduction of $O(n^2)$ operations compared to using conventional products of plane rotations.

Algorithm 4. BFGS with factored Hessian approximation.

```

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

```

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 $H_k = R_k^T R_k$, 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.

4. Convergence and the Role of the Least Acceptable Curvature

If f is bounded below then a strong Wolfe step exists for all $\eta_w \in (\eta_A, 1)$. It follows that if the line search is guaranteed to compute a strong Wolfe step then the approximate curvature $y_k^T d_k$ is positive if f is bounded below (see the discussion associated with (1.6)). If $y_k^T d_k$ is not positive, then H_{k+1} is indefinite or undefined, and most implementations skip or modify the update.

There is no known proof that the conventional BFGS method is globally convergent on a well-posed unconstrained minimization problem, although the result is widely believed to be true and there is ample practical supporting evidence. The positivity and magnitude of the approximate curvature plays an important role in any theoretical convergence analysis. Making certain assumptions about, or imposing restrictions on, the approximate curvature has allowed several authors to make headway on proving global convergence. If f is a convex function, then it holds that $\|y_k\|^2 / y_k^T d_k \leq M$, where M is an upper bound on the norm of the Hessian. If this inequality is *assumed* to hold for an arbitrary function then global convergence can be established (see Powell [24] for details of the proof). In practice, analogous conditions are *imposed* on the approximate curvature under the assumption that global convergence will be encouraged. A condition based on the Powell's proof is $y_k^T d_k \geq \epsilon_U \|y_k\|^2$, where ϵ_U is a small preassigned positive parameter.

An alternative approach proposed by Li and Fukushima [14] gives a globally convergent

method. In this case, the update is given by

$$H_{k+1} = \begin{cases} H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \frac{1}{y_k^T d_k} y_k y_k^T & \text{if } \frac{y_k^T d_k}{\|d_k\|^2} \geq \epsilon_U \|\nabla f(x_k)\|^c; \\ H_k & \text{otherwise,} \end{cases}$$

where ϵ_U and c are preassigned positive parameters. Li and Fukushima suggest two specific rules for ϵ_U and c :

- rule 1: $\epsilon_U = 10^{-6}$; if $\|\nabla f(x_k)\| \geq 1$ then $c = 10^{-2}$ else $c = 3$;
rule 2: $\epsilon_U = 10^{-6}$; $c = 1$.

Any condition on $y_k^T d_k$ may be considered as representing some “level of caution” associated with accepting a small approximate curvature. In this context, imposing a lower bound of zero, i.e., accepting any value such that $y_k^T d_k > 0$, is regarded as “incautious”. A summary of the most prominent acceptance rules is given in Table 1. The numerical performance of the BFGS method with caution levels C0–C4 is discussed in Section 5.

Caution level	Acceptance test
C0	$y_k^T d_k > 0$
C1	$y_k^T d_k \geq \epsilon_U \ \nabla f(x_k)\ ^c \ d_k\ ^2$
C2	$y_k^T d_k \geq \epsilon_U \ \nabla f(x_k)\ \ d_k\ ^2$
C3	$y_k^T d_k \geq \epsilon_U \ d_k\ ^2$
C4	$y_k^T d_k \geq \epsilon_U \ y_k\ ^2$

Table 1: Five tests for the acceptance of the BFGS update based on the magnitude of the approximate curvature $y_k^T d_k$. In all cases, $\epsilon_U = 10^{-6}$. The value of c for C1 is $c = 10^{-2}$ if $\|\nabla f(x_k)\| \geq 1$, otherwise $c = 3$.

5. Numerical Methods

The results were obtained for optimization problems from the CUTEst test collection (see Bongartz et al. [2] and Gould, Orban and Toint [13]). As of August 1, 2026 there are 291 unconstrained problems in the test set. A subset of 286 problems was selected based on the number of variables. In particular, a problem was chosen if the number of variables was of the order of 5000 or less. The same criterion was used to set the dimension of those problems for which the problem size can be specified. The 5 problems BA-L16LS, BA-L21LS, BA-L49LS, BA-L52LS, and BA-L73LS have more than 23K variables and were excluded from the test set.

Each of the four solvers `bfgsM`, `bfgsMS`, `bfgsR`, and `bfgsRS` was run on all 286 problems. For each run, the recorded information included the number of iterations, the number of function evaluations, the CPU time, and the outcome of the optimization. The outcome was defined by specific tolerances (see Section 5), and classified the result as “optimal”, “near optimal but badly scaled”, “line search failure”, or “too many iterations”.

Hardware and software. The code for each solver was written in the MATLAB programming language. The runs were done using MATLAB version R2024a on a System76 Prime Elite Linux machine running Ubuntu 22.04 with a Ryzen 9 9950X (16 Cores, 32

Threads) and 48GB DDR5 RAM (5600MHz). The data collection, analysis, and plotting software was written in Python using Numpy, Pandas, Matplotlib and Seaborn.

The line search. All methods were implemented using a Wolfe line search with parameters $\eta_w = 0.9$ and $\eta_A = 10^{-4}$ (see the Wolfe conditions (1.4) and (1.5)). These values reflect the assumption that the function evaluations dominate the cost of an iteration.

A Wolfe line search generates a sequence $\alpha_k^{(j)}$ that converges to a point $\alpha_k \in (0, \alpha_{\max}]$, where $\alpha_{\max}$ denotes an infinite change in α . If $\alpha_k \in (0, \alpha_{\max})$ then α_k satisfies the Wolfe conditions (1.4) and (1.5). If $\alpha_k = \alpha_{\max}$ then α_k satisfies the Armijo condition (1.4) but may not satisfy the Wolfe condition (1.5).

The line search involves the specification of two positive parameters δ_{∞} and $\delta_{\max}$. The scalar δ_{∞} is the definition of an infinite change in x , which implies the limit $\alpha_{\max} = \delta_{\infty}/(1 + \|p_k\|)$ on α . The scalar $\delta_{\max}$ is a preassigned limit on the initial step size $\alpha_k^{(0)}$. On a well-scaled problem, the ideal value for $\alpha_k^{(0)}$ is the unit step. However, on a badly-scaled problem, the early search directions may be large in norm and a limit on $\alpha_k^{(0)}$ is necessary. The value specified was

$$\alpha_k^{(0)} = \min \{1, \alpha_L, \alpha_{\max}\}, \quad \text{and} \quad \alpha_L = \begin{cases} \alpha_{\max} & \text{if } \alpha_{\max} \|\hat{p}_k\| \leq \delta_{\max}; \\ \delta_{\max} / \|\hat{p}_k\| & \text{otherwise,} \end{cases}$$

where $\hat{p}_k$ is the normalized search direction $\hat{p}_k = p_k / (1 + \|x_k\|)$.

The termination criteria. A run was considered to be a success based on two criteria. The first was

$$\|\nabla f(x_k)\| \leq \varepsilon_s (1 + |f(x_k)|) \quad \text{or} \quad \|\nabla f(x_k)\| \leq \epsilon_M^{1/2},$$

where ε_s is a preassigned stationarity tolerance and ϵ_M is the relative machine precision. If the problem is badly scaled, a line search failure may occur before these conditions can be satisfied. In this case the run was considered a success if

$$\|\nabla f(x_k)\| \leq \epsilon_M^{1/2} \|\nabla f(x_0)\| \quad \text{or} \quad |f(x_k)| \leq \epsilon_M^{2/3} (1 + |f(x_0)|).$$

The specific values of the parameters used in the runs are given in Table 2.

Parameter	Value	Parameter	Value
Stationarity tolerance, ε_s	1.0e-05	Iteration limit, N	6000
Wolfe line-search tolerance, η_w	9.0e-01	Armijo line-search tolerance, η_A	1.0e-04
Initial step size limit, $\delta_{\max}$	10	Infinite change in x , δ_{∞}	20
Condition number limit	1.0e+16	Unbounded objective	-1.0e+10

Table 2: Constants and tolerances used across all problems and solvers.

Discussion and results. The relative performance of the solvers is summarized using performance profiles. (For a description of how performance profiles should be interpreted, see Dolan and Moré [7].) 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 the optimization literature.

The profiles given in Figures 1–4 indicate that imposing a positive lower bound on the approximate curvature hinders the performance of the BFGS method. Overall, the best

performance is achieved by requiring only that $y_k^T d_k > 0$. This seems to support the belief that the BFGS method is in fact globally convergent, and that restrictions on the approximate curvature are helpful for showing theoretical properties but are not necessary or helpful in practice.

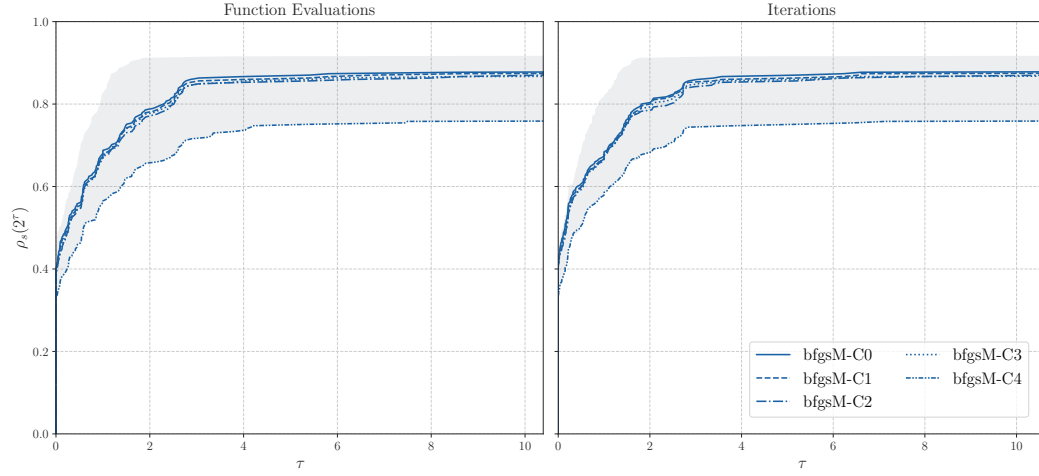


Figure 1: Performance profiles comparing caution levels for **bfgsM**.

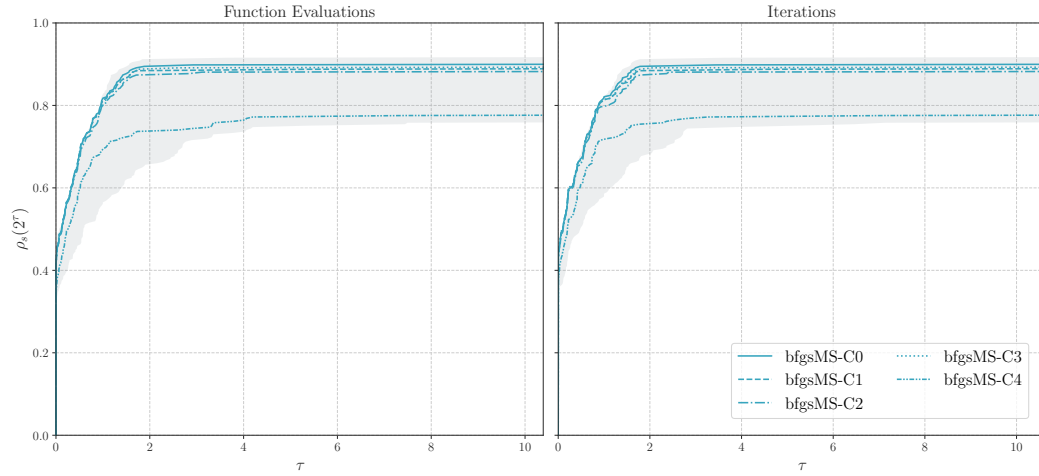
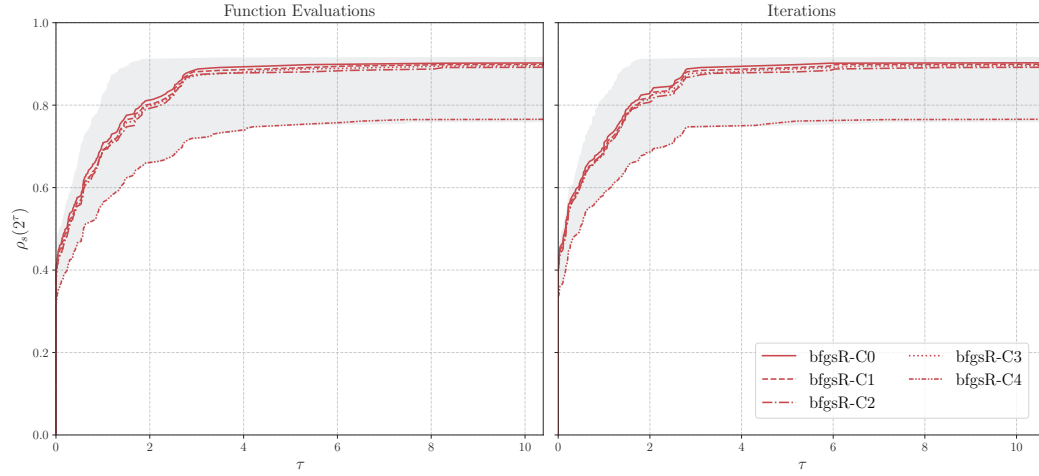
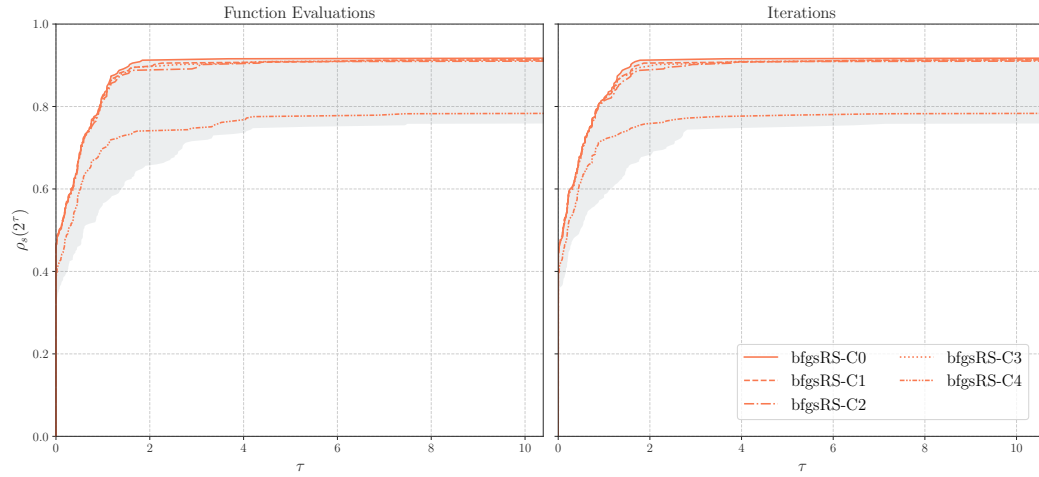


Figure 2: Performance profiles comparing caution levels for **bfgsMS**.

Figure 3: Performance profiles comparing caution levels for **bfgsR**.Figure 4: Performance profiles comparing caution levels for **bfgsRS**.

The comparison of **bfgsM** with the factored variation **bfgsR** highlights the importance of numerical stability in the update formula. The fact that the relation $M_k^{-1} = R_k^T R_k$ 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 5 is owed entirely to the management of numerical factors like conditioning and indefiniteness, made possible by the use of matrix factorization.

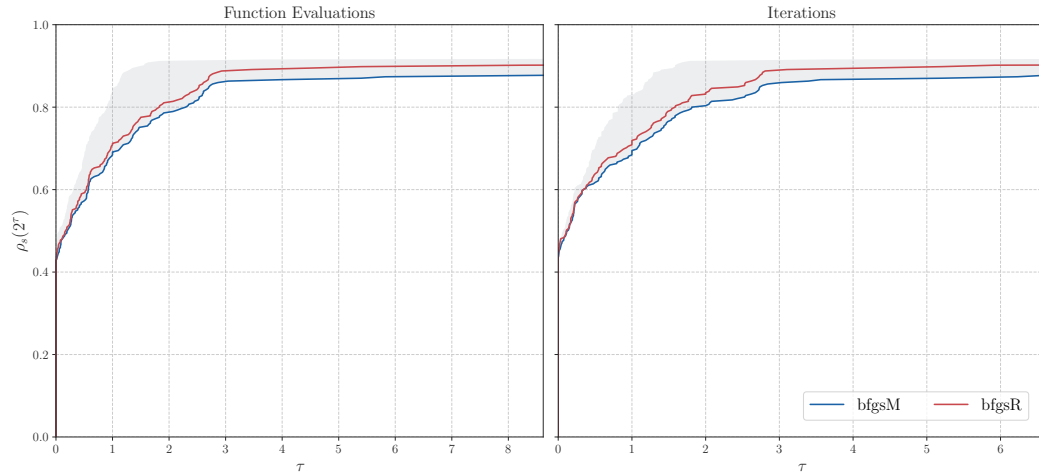


Figure 5: Performance profiles comparing function evaluations and iterations for **bfgsM** and **bfgsR**.

The self-scaled updates, whose purpose is to limit the negative effects of ill-conditioning, provide another example of the benefits of taking numerical 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 6, and 7. Nocedal and Yuan [16] establish the global convergence of self-scaling methods and show that the unit step may not be accepted by the line search near the solution. However, the non-acceptance of the initial step does not appear to affect the efficiency.

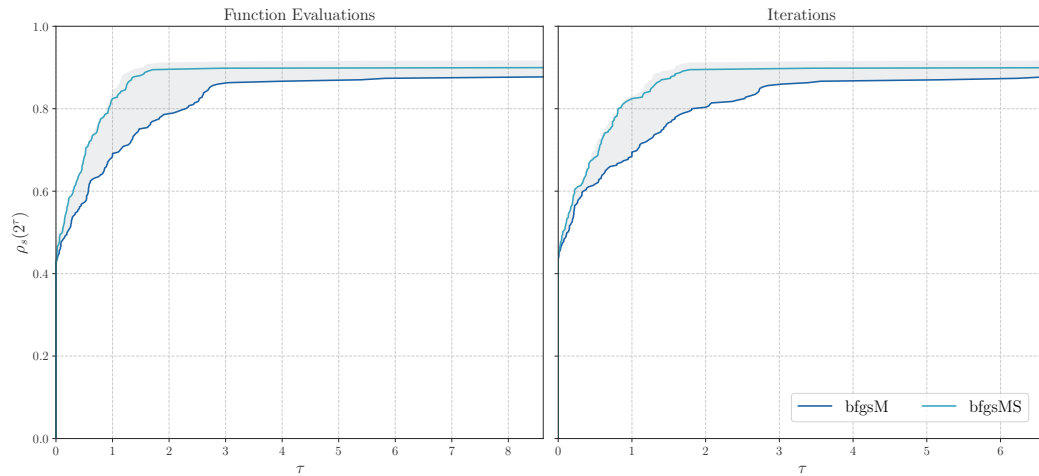


Figure 6: Performance profiles comparing function evaluations and iterations for **bfgsM** and **bfgsMS**.

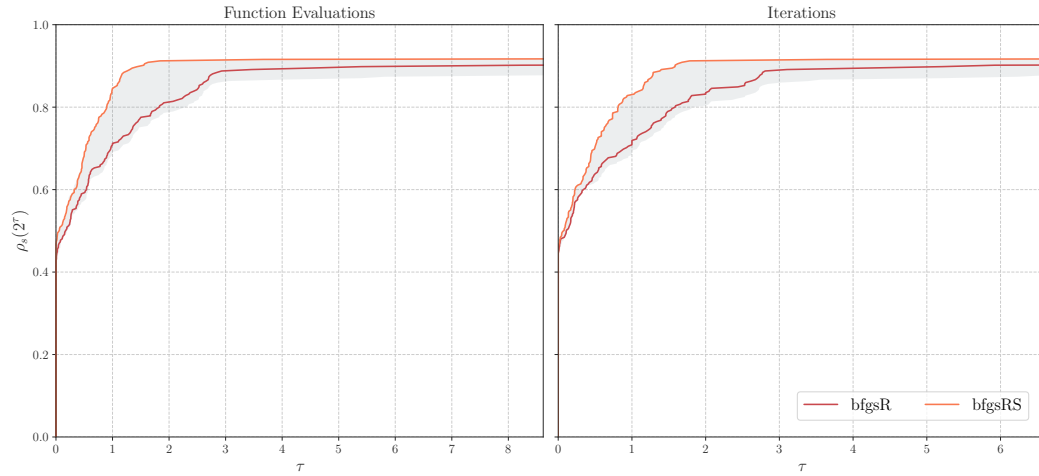


Figure 7: Performance profiles comparing function evaluations and iterations for **bfgsR** and **bfgsRS**.

It is also worth noting that the improvement seen in factored methods is preserved when self-scaling is applied in **bfgsMS** and **bfgsRS** (see Figure 8).

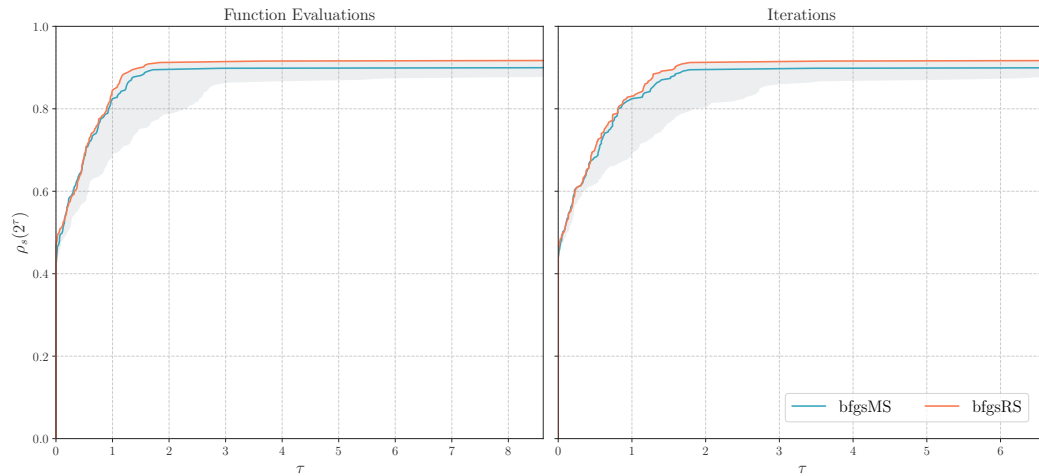


Figure 8: Performance profiles comparing function evaluations and iterations for **bfgsMS** and **bfgsRS**.

The performance profiles indicate that the factored methods **bfgsR** and **bfgsRS** exhibit the best results, with the overall highest performing method being **bfgsRS** (see Figures 5, and 9). Maintaining a factorization allows us to sidestep the issue of error accumulation resulting in failure of hereditary positive definiteness. This is immediately apparent, since $x^T R^T R x = \|R x\|^2 > 0$ 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.

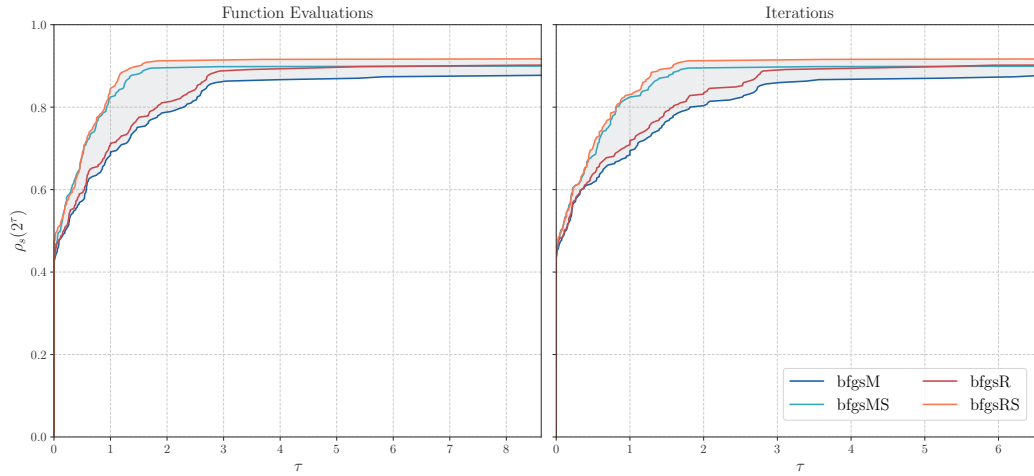


Figure 9: Performance profiles comparing function evaluations and iterations for all solvers.

6. Conclusion

We have shown that a combination of self-scaling and a factored Hessian shows significant and consistent improvement over the conventional BFGS method. The results appear to contradict the widely-held view that self-scaled updates and stable factorization methods provide no additional benefits over a conventional implementation.

References

- [1] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966. [2](#)
- [2] Ingrid Bongartz, Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Trans. Math. Software*, 21(1):123–160, 1995. [10](#)
- [3] Kenneth W. Brodlie, Alexander R. Gourlay, and John Greenstadt. Rank-one and rank-two corrections to positive definite matrices expressed in product form. *J. Inst. Math. Appl.*, 11:73–82, 1973. [5](#)
- [4] William C. Davidon. Variable metric methods for minimization. Report ANL-5990, A.E.C. Research and Development, Argonne National Laboratory, Argonne, IL, 1959. [2](#)
- [5] William C. Davidon. Optimally conditioned optimization algorithms without line searches. *Math. Program.*, 9:1–30, 1975. [5](#)
- [6] John E. Dennis, Jr. and Robert B. Schnabel. A new derivation of symmetric positive definite secant updates. In *Nonlinear Programming, 4 (Proc. Sympos., Special Interest Group on Math. Programming, Univ. Wisconsin, Madison, Wis., 1980)*, pages 167–199. Academic Press, New York, 1981. [5](#)
- [7] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2, Ser. A):201–213, 2002. [11](#)
- [8] Roger Fletcher and Michael J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168, 1963. [2](#)
- [9] Philip E. Gill, Gene H. Golub, Walter Murray, and Michael A. Saunders. Methods for modifying matrix factorizations. *Math. Comput.*, 28:505–535, 1974. [6](#), [8](#)
- [10] 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. [2](#)
- [11] Philip E. Gill and Jeb H. Runnoe. On recent developments in BFGS methods for unconstrained optimization. Center for Computational Mathematics Report CCoM 22-04, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2022. [1](#), [3](#)
- [12] Donald Goldfarb. Factorized variable metric methods for unconstrained optimization. *Math. Comput.*, 30:796–811, 1976. [6](#), [8](#)

-
- [13] 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. [3](#), [10](#)
 - [14] Dong-Hui Li and Masao Fukushima. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM J. Optim.*, 11(4):1054–1064, 2001. [9](#)
 - [15] Jorge J. Moré and David J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software*, 20(3):286–307, 1994. [2](#)
 - [16] Jorge Nocedal and Ya-Xiang Yuan. Analysis of a self-scaling quasi-Newton method. *Math. Program.*, 61(1):19–37, 1993. [14](#)
 - [17] Shmuel S. Oren. Self-scaling variable metric algorithms without line search for unconstrained optimization. *Math. Comp.*, 27(124):873–885, October 1973. [4](#)
 - [18] Shmuel S. Oren. On the selection of parameters in self-scaling variable metric algorithms. *Math. Program.*, 7:351–367, 1974. [4](#)
 - [19] Shmuel S. Oren. Self-scaling variable metric (SSVM) algorithms, Part ii: implementation and experiments. *Management Science*, 20:863–874, 1974. [4](#)
 - [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. [4](#)
 - [21] Shmuel S. Oren and Emilio Spedicato. Optimal conditioning of self-scaling variable metric algorithms. *Math. Program.*, 10:70–90, 1976. [4](#)
 - [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. [2](#)
 - [23] Michael R. Osborne and Michael A. Saunders. Descent methods for minimisation. In R. S. Anderssen, L. S. Jennings, and D. M. Ryan, editors, *Optimisation*, pages 221–237. University of Queensland Press, Brisbane, 1972. [5](#)
 - [24] 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. [9](#)
 - [25] Jack Sherman and Winifred 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](#)
 - [26] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Rev.*, 11:226–235, 1969. [2](#)
 - [27] Max A. Woodbury. Inverting modified matrices. Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950. [3](#)