Quasi-Newton Methods for Unconstrained Optimization

Jeb H. Runnoe
Advisor: Philip E. Gill
Department of Mathematics
University of California, San Diego

April 24th, 2020

Abstract

Many techniques for solving general nonlinear unconstrained optimization problems involve iteratively minimizing a model function that satisfies certain interpolation conditions. These conditions provide a model that behaves like the objective function in the neighborhood of the current iterate. The model functions often involve second-order derivatives of the objective function, which can be expensive to calculate. The fundamental idea behind quasi-Newton methods is to maintain an approximation to the Hessian matrix. The practical success of quasi-Newton methods has spurred a great deal of interest and research that has resulted in a considerable number of variations of this idea. The analytical difficulties associated with characterizing the performance of these algorithms means there is a real need for practical testing to support theoretical claims. The goal of this project is to describe, implement, and test these methods in a way that is uniform, systematic, and consistent. In the first part of the paper, we derive several classical quasi-Newton methods, discuss their relative benefits, and show how to implement them. In the second part, we investigate more recent variations, explain their motivation and theory, and analyze their performance.

Contents

1. Introduction

In this paper we will consider optimization problems of the form

$$\min_{x \in \mathbb{R}^n} f(x). \tag{1.1}$$

Since there are no constraints on the allowable values of x, (1.1) is referred to as an unconstrained optimization problem. Our task is, given a function $f : \mathbb{R}^n \to \mathbb{R}$, to find a point $x^* \in \mathbb{R}^n$ for which

$$f(x^*) \leq f(x) \quad \text{for all} \quad x \in \mathbb{R}^n \quad \text{with} \quad \|x - x^*\| < \delta, \tag{1.2}$$

for some $\delta > 0$. The problem of *global* optimization, i.e. finding a point that satisfies (1.2) for all x regardless of its proximity to x^*, is often intractable and will not be considered in this paper.

One of the most foundational methods for solving (1.1) is Newton's method. A basic result of calculus is that a necessary condition for optimality is $\nabla f(x^*) = 0$. The aim of Newton's method is to find a zero of the gradient function. It is an iterative process in which, at each iteration, the function gradient is approximated near the kth iterate using a linear approximation

$$\nabla f(x_k + p_k) \approx \nabla f(x_k) + \nabla^2 f(x_k) p_k.$$

Treating this as an equality and setting the left-hand side to zero gives the Newton equations $\nabla^2 f(x_k) p_k = -\nabla f(x_k)$. The solution p_k is the direction to the next iterate x_{k+1}.

This method relies on the availability of the Hessian $\nabla^2 f(x_k)$ at each iteration. Often the Hessian is expensive to compute or simply not available. The basic idea of *quasi-Newton methods* is to approximate the Hessian and use the approximation $H_k \approx \nabla^2 f(x_k)$ to determine the search direction. Rather than solve the Newton equations, we solve the system $H_k p_k = -\nabla f(x_k)$. The main question then is to determine how to construct H_{k+1} from H_k in a way that incorporates information about the objective function f that is obtained by moving from x_k to x_{k+1}.

Many variations of this fundamental idea have been developed over the past 60 years. Among them is the well-known BFGS method, which seems to perform best in practice for reasons that are not fully understood. This speaks to the difficulty of theoretically characterizing how these methods perform and the importance of practical testing to support claims of improved reliability or convergence.

Today we have numerical methods that were not available when these algorithms were conceived. We will leverage the Constrained and Unconstrained Testing Environment with safe threads (CUTEst) (see [10]) to compare these algorithms performance.

1.1. Notation

In general, notation will be defined as needed throughout this paper. However the notation summarized in 1.1 is used consistently and without further definition.

Notation	Meaning
p_k	Search direction
α_k	Step length
$d_k = \alpha_k p_k = x_{k+1} - x_k$	Step
$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$	Change in gradient
$H_k \approx \nabla^2 f(x_k)$	Hessian approximation
$M_k \approx (\nabla^2 f(x_k))^{-1}$	Inverse Hessian approximations
$\|\cdot\|$	Euclidean vector or induced matrix norm

Table 1: Common notation.

1.2. Background

Almost all methods for unconstrained optimization generate a sequence of iterates $\{ x_k \}_{k=0}^{\infty}$ such that x_{k+1} 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 $f(x_k) + \nabla f(x_k)^T(x - x_k)$. If x_{k+1} is computed as $x_{k+1} = x_k + \alpha_k p_k$, where p_k is a vector and α_k is a scalar step, then the sufficient-decrease condition may be written as

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \alpha_k \eta_A \nabla f(x_k)^T p_k, \tag{1.3}$$

(see, e.g., Armijo [2], Ortega and Rheinboldt [19]). Many practical methods satisfy the Armijo condition in conjunction with a condition on the directional derivative $\nabla f(x_k + \alpha_k p_k)^T p_k$. In particular, the strong Wolfe conditions require that α_k satisfy both (1.3) and

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

where η_W is a preassigned scalar such that $\eta_W \in (\eta_A, 1)$. (See, e.g., Wolfe [20], Moré and Thuente [12], and Gill et al. [8]). Alternatively, the weak Wolfe conditions involve the Armijo condition (1.3) in conjunction with the one-sided condition

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq \eta_W \nabla f(x_k)^T p_k \tag{1.5}$$

instead of (1.4). The strong Wolfe conditions allow η_W to be chosen to vary the accuracy of the step. If η_A is fixed at a value close to zero (e.g., 10^{-4}), then a value of η_W close to η_A gives a "tighter" or more accurate step with respect to closeness to a critical point of $\nabla f(x_k + \alpha p_k)^T p_k$. A value of η_W close to one results in a "looser" or more approximate step. If the Wolfe conditions are used in conjunction with a quasi-Newton method, there is the additional benefit that the conditions (1.4) and (1.5) guarantee a positive-definite approximate Hessian H_k can be updated to give a positive definite H_{k+1} for the next iterate. If α_k satisfies either the strong Wolfe condition (1.4) or the weak Wolfe condition (1.5), then $y_k^T d_k > 0$, where $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, and $d_k = x_{k+1} - x_k$. The inequality $y_k^T d_k > 0$ is a necessary condition for the updated quasi-Newton Hessian to be positive definite.

A typical Wolfe line search may be viewed as a two-stage process. The first stage involves the determination of an interval containing a Wolfe step, if one exists. The second stage is to locate a Wolfe step in this interval using safeguarded

polynomial interpolation. If the first stage fails, then the objective function is necessarily unbounded below. The key principle that drives the first stage is that certain conditions may be formulated that determine if an interval contains a Wolfe step. Much of the discussion in this section is based on the work of Morè and Thuente [12]. More information may be found in Wolfe [21] and Nocedal and Wright [13]. In order to simplify the notation we omit the suffix k and consider the univariate function $\phi(\alpha) = f(x + \alpha p)$. With this notation the Wolfe conditions (1.3) and (1.4) may be written in the form

$$\phi(\alpha) \leq \phi(0) + \alpha\eta_A\phi'(0), \quad \text{and} \quad |\phi'(\alpha)| \leq \eta_W|\phi'(0)|.$$

Much of the theory associated with a Wolfe line search is based on the properties of the auxiliary function

$$\omega(\alpha) = \phi(\alpha) - \big(\phi(0) + \alpha\eta_A\phi'(0)\big), \quad \text{with} \quad \omega'(\alpha) = \phi'(\alpha) - \eta_A\phi'(0).$$

Moré and Sorensen [11] show that a minimizer of this function at which ω is negative satisfies the Wolfe conditions. The function ω and its relationship to ϕ are depicted in Figure 1.

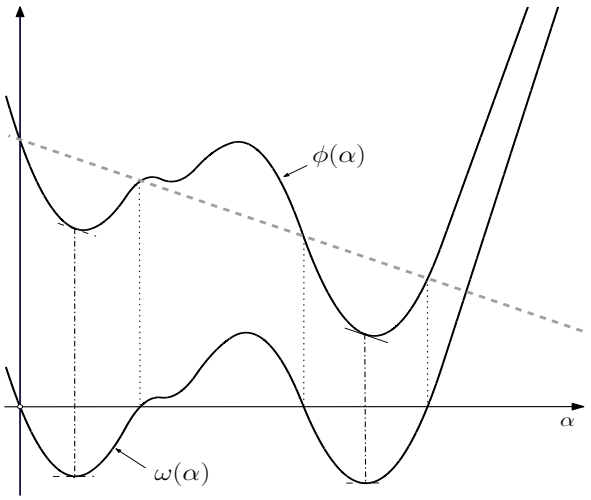

Figure 1: The graph depicts $\phi(\alpha) = f(x + \alpha p)$ as a function of positive α, with the shifted function $\omega(\alpha) = \phi(\alpha) - \big(\phi(0) + \alpha\eta_A\phi'(0)\big)$ superimposed. The dashed line represents the affine function $\phi(0) + \alpha\eta_A\phi'(0)$.

2. Quasi-Newton Methods

The class of *quasi-Newton methods* constitutes one of the great breakthroughs in numerical optimization. The first quasi-Newton method was proposed in 1959 by W. C. Davidon [3], in a technical report published at the Argonne National Laboratory. A famous paper in 1963 by R. Fletcher and M. J. D. Powell [6], published

in *The Computer Journal* improved on Davidon's derivation and was largely responsible for making these methods widely known. Their subsequent success in practice has essentially revolutionized optimization for problems in which only first derivatives are available. In this section we introduce a simple quasi-Newton update proposed by C. G. Broyden that may be considered as a particular type of secant update.

2.1. The quasi-Newton conditions

Using the $(n+1)$-point secant method as a starting point (see Gill and Wright [9, Chapter 3]), let D_k be the matrix whose columns are composed of the previous n values of $d_k = x_{k+1} - x_k$, and assume for simplicity that the columns of the matrix D_k occur in the order d_{k-1}, d_{k-2}, ..., d_{k-n}. The $(n+1)$-point secant method gives an approximate Hessian H_{k+1} that may be written in form

$$H_{k+1} = H_k + U_k = H_k + \frac{1}{v_k^T d_k}(y_k - H_k d_k)v_k^T, \quad v_k^T d_k \neq 0,$$

with H_{k+1} satisfying the secant conditions

$$\begin{aligned} &H_{k+1}d_j = H_k d_j, \quad j = k-1, \ldots, k-n+1, \\ &H_{k+1}d_k = y_k. \end{aligned} \tag{2.1}$$

This states that U_k leaves H_k "untouched" along the $n-1$ previous directions d_{k-1}, d_{k-2}, ..., d_{k-n+1}, and makes H_{k+1} behave like the exact Hessian along the "new" direction d_k. The last condition is justified by the expansion

$$\begin{aligned} H_{k+1}d_k = y_k &= \nabla f(x_k + d_k) - \nabla f(x_k) \\ &= \nabla^2 f(x_k)d_k + \int_0^1 \left(\nabla^2 f(x_k + td_k) - \nabla^2 f(x_k)\right) d_k \, dt \\ &= \nabla^2 f(x_k)d_k + O(L\|d_k\|^2), \end{aligned}$$

where L is the Lipschitz constant. If ∇f is an affine function—i.e., $\nabla f(x) = c + Hx$ for some constant H and c, then the last relation (2.1) is exact, with

$$H_{k+1}d_k = y_k = \nabla f(x_{k+1}) - \nabla f(x_k) = c + Hx_{k+1} - (c + Hx_k) = Hd_k = \nabla^2 f(x_k)d_k.$$

If the n directions d_k, d_{k-1}, ..., d_{k-n+1} are linearly independent, then the conditions (2.1) are sufficient to define U_k uniquely. This means that some of the conditions must be relaxed in order to define methods based on alternative choices for U_k. The standard strategy is to relax the conditions $H_{k+1}d_j = H_k d_j$, $j = k-1$, $k-2$,..., $k-n$, but keep the last condition

$$H_{k+1}d_k = y_k, \tag{2.2}$$

which is known as the *quasi-Newton condition*. The simplest possible form for U_k is the *rank-one matrix*

$$U_k = u_k v_k^T, \tag{2.3}$$

for some vectors u_k and v_k to be determined. Substituting (2.3) in (2.2), we have

$$H_{k+1}d_k = (H_k + u_k v_k^T)d_k = y_k, \quad \text{or} \quad u_k(v_k^T d_k) = y_k - H_k d_k, \tag{2.4}$$

which has several interesting implications. First, H_k itself satisfies the quasi-Newton condition if $H_k d_k = y_k$. Second, the rows of U_k are orthogonal to d_k (i.e., $U_k d_k = 0$) if $v_k^T d_k = 0$; if $H_k d_k \neq y_k$, the updated matrix H_{k+1} can satisfy the quasi-Newton condition only if $v_k^T d_k \neq 0$. Finally, assuming that $H_k d_k \neq y_k$ and $v_k^T d_k \neq 0$, the quasi-Newton condition will be satisfied by the rank-one matrix $U_k = u_k v_k^T$ only if $(H_k + U_k)d_k = H_k d_k + u_k(v_k^T d_k) = y_k$, so that u_k must be a multiple of $y_k - H_k d_k$, with

$$H_{k+1} = H_k + \frac{1}{v_k^T d_k}(y_k - H_k d_k)v_k^T. \tag{2.5}$$

One obvious choice of v_k is $v_k = d_k$, which makes U_k well-defined whenever d_k is not zero. In this case,

$$H_{k+1} = H_k + \frac{1}{d_k^T d_k}(y_k - H_k d_k)d_k^T, \tag{2.6}$$

which is known by several names, including the *Broyden update*, the *good Broyden update*, and *Broyden's second update*.

Each iteration of a quasi-Newton method requires solution of the $n \times n$ linear system $H_k p_k = -\nabla f(x_k)$, and a "naive" implementation would require $O(n^3)$ flops[1] to perform this calculation. In Section 3 we show how the special relationship between consecutive Hessian approximations allows a *factorization* of the Hessian to be updated in a numerically stable way (see Gill and Murray [7] and Dennis and Schnabel [4]). Once these factors are known, the direction p_k can be computed in only $O(n^2)$ flops.

Methods for modifying matrix factorization were not available when quasi-Newton methods were being developed, and so the earliest quasi-Newton updates were derived in terms of the *inverse* Hessian. With exact arithmetic, every update to H_k is associated with an equivalent update to H_k^{-1}, using the Sherman-Morrison-Woodbury formula. For example, if M_k denotes H_k^{-1}, the inverse form of the general update (2.6) is

$$M_{k+1} = M_k + \frac{1}{v_k^T M_k y_k}(d_k - M_k y_k)v_k^T M_k,$$

from which the inverse form of the Broyden update

$$M_{k+1} = M_k + \frac{1}{d_k^T M_k y_k}(d_k - M_k y_k)d_k^T M_k \tag{2.7}$$

is obtained by setting $v_k = d_k$ as before.

However, it is possible to derive quasi-Newton updates for the inverse Hessian *directly* by considering update matrices U_k that satisfy the quasi-Newton condition

$$(M_k + U_k)y_k = M_{k+1}y_k = d_k,$$

[1] floating-point operations

(2.2). Using a formula analogous to (2.4) we can write all admissible rank-one updates in the form

$$M_{k+1} = M_k + \frac{1}{w_k^T y_k}(d_k - M_k y_k) w_k^T,$$

where w_k is arbitrary, except for the fact that it must satisfy $w_k^T y_k \neq 0$. In this case, the "obvious" formula is defined by setting $w_k = y_k$, giving

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

The derivation of this update predates the Broyden update (2.6) because of the perceived additional expense of solving the equations $H_k p_k = -\nabla f(x_k)$ rather than forming the product $p_k = -M_k \nabla f(x_k)$ at each step. This formula is not the same as the Broyden update (2.6), and is known as *Broyden's first update* or the *bad Broyden update*. It is easy to see that we can write down this formula as an update to H_k by simply interchanging "M" for "B" and "d" for "y" in the inverse formula (2.7), giving

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

2.2. The quasi-Newton Hessian

A *quasi-Newton line-search method* (or, when the meaning is clear, simply a *quasi-Newton method*) is typically defined by the sequence of iterates $x_{k+1} = x_k + \alpha_k p_k$, where p_k is the solution of

$$H_k p_k = -\nabla f(x_k), \tag{2.8}$$

α_k satisfies appropriate sufficient decrease conditions, and H_k is updated at every iteration using a quasi-Newton update. When H_k is symmetric and positive definite, p_k of (2.8) is the unique minimizer of the quadratic model $q_k(x)$ (2.9), whose Hessian *changes* at every iteration—hence the name *variable metric* methods originally suggested by Davidon and still used by some authors.

Up to this point, we have not made use of the fact that the Hessian H is actually the second derivative of a scalar-valued function f. The two fundamental properties of H that we plan to exploit are *symmetry* and *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. A crucial property of symmetric positive-definite approximate Hessians is they may be used directly to define a *quadratic model* whose minimizer can be used to define the next iterate. By analogy with a Newton-based line-search method, a quasi-Newton approximation H_k may be used to define a local quadratic model of f:

$$q_k(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T H_k (x - x_k), \tag{2.9}$$

which has a unique minimizer only if H_k is symmetric positive definite.

The *curvature* of f plays a particularly important role in the development and understanding of quasi-Newton methods for minimization. The curvature of f along d_k at an iterate x_k is given by $d_k^T \nabla^2 f(x_k) d_k$, which cannot be computed exactly because H is (by assumption) unknown. However, a Taylor-series expansion of the gradient $\nabla f(x_k + d_k)$ gives

$$(\nabla f(x_k + d_k) - \nabla f(x_k))^T d_k = d_k^T \nabla^2 f(x_k) d_k + \int_0^1 d_k^T (\nabla^2 f(x_k + t d_k) - \nabla^2 f(x_k)) d_k \, dt,$$

and hence, if $\|d_k\|$ is "small enough", the curvature at x_k should be "close" to the curvature at the intermediate point $x_k + t d_k$, and

$$y_k^T d_k \approx d_k^T \nabla^2 f(x_k) d_k.$$

The quantity $y_k^T d_k$, defined from only first-order information, is known as the *approximate curvature* of f at x_k. An important property of the quasi-Newton condition is that it forces the new quadratic model q_{k+1} to have curvature $y_k^T d_k$ along d_k; i.e.,

$$d_k^T H_{k+1} d_k = y_k^T d_k. \tag{2.10}$$

We say that the quasi-Newton condition *installs* the approximate curvature $y_k^T d_k$ as the *exact* curvature of the new quadratic model. With this interpretation, the approximate Hessian H_k, represents (in some sense) curvature information that has been accumulated at iterates preceding x_k. The move from x_k to x_{k+1} provides further information about curvature that may be incorporated in a new Hessian approximation H_{k+1}.

2.3. The Symmetric Rank-One Update

The rank-one update of the form

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

satisfies the quasi-Newton condition $H_{k+1} d_k = y_k$ for all d_k such that $y_k^T d_k$ is nonzero (see (2.5)). For this formula, hereditary symmetry can be achieved only if v_k is a multiple of $y_k - H_k d_k$. It follows that there is a *unique* rank-one update with hereditary symmetry:

$$H_{k+1} = H_k + \frac{1}{(y_k - H_k d_k)^T d_k} (y_k - H_k d_k)(y_k - H_k d_k)^T, \tag{2.11}$$

called (not surprisingly) the *symmetric rank-one (SR1) update*. The SR1 update is defined only if $(y_k - H_k d_k)^T d_k \neq 0$.

The picture becomes more complicated if we seek symmetric quasi-Newton updates with hereditary positive-definiteness. It is easy to see that relation (2.10) implies that a positive value for the approximate curvature is a *necessary condition* for the existence of a positive definite H_{k+1}.

Unfortunately, positive approximate curvature is not a *sufficient* condition for H_{k+1} to be positive definite. For example, the symmetric rank-one update (2.11) does *not* possess hereditary positive-definiteness, even if the approximate curvature is positive. For example, consider the SR1 update with

$$H_k = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \quad d_k = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \text{ and } y_k = \begin{pmatrix} -3 \\ 2 \end{pmatrix}.$$

In this case, H_k is positive-definite and $y_k^T d_k = 1$, yet the new Hessian is the indefinite matrix

$$H_{k+1} = \begin{pmatrix} 2 & 1 \\ 1 & -3 \end{pmatrix}.$$

2.4. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Update

As the SR1 update is the only rank-one update with the property of hereditary symmetry, update matrices U_k with rank greater than one must be used to obtain hereditary positive definiteness. Arguably the simplest possible U_k (other than a matrix of rank one) is a matrix of rank two, with

$$U_k = \gamma_k u_k u_k^T + \delta_k v_k v_k^T,$$

for vectors u_k, v_k, and scalars γ_k, δ_k to be determined. Substituting U_k in the quasi-Newton condition $H_{k+1} d_k = (H_k + U_k) d_k = y_k$, yields

$$y_k - H_k d_k = U_k d_k = \gamma (u_k^T d_k) u_k + \delta_k (v_k^T d_k) v_k,$$

which implies that suitable u_k and v_k can be found as linear combinations of y_k and $H_k d_k$. Of course, there are an infinite number of possible choices for u_k and v_k. However, as we saw in our derivation of the Broyden update, it usually pays to "keep it simple". In this case, one of the simplest possible choices is $u_k = H_k d_k$ and $v_k = y_k$, giving

$$U_k = \gamma_k H_k d_k d_k^T H_k + \delta_k y_k y_k^T.$$

It remains to find γ_k and δ_k, which are used to ensure that the quasi-Newton condition $H_{k+1} d_k = y_k$ is satisfied. Using the form of U_k above, we have

$$H_{k+1} d_k - y_k = (H_k + U_k) d_k - y_k = H_k d_k (1 + \gamma_k d_k^T H_k d_k) - y_k (1 - \delta_k y_k^T d_k).$$

An obvious choice of γ_k and δ_k that makes $H_{k+1} d_k - y_k = 0$ is $\gamma_k = -1/d_k^T H_k d_k$, and $\delta_k = 1/y_k^T d_k$, which leads to the well-known *BFGS (Broyden-Fletcher-Goldfarb-Shanno) update*:

$$H_{k+1} = H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \frac{1}{y_k^T d_k} y_k y_k^T, \tag{2.12}$$

which is a symmetric, rank-two update to H_k.

The BFGS update has the property of hereditary positive definiteness if $y_k^T d_k > 0$. There are a number of ways to show this. It is easily verified by direct multiplication that

$$H_{k+1} = (I + v_k d_k^T) H_k (I + d_k v_k^T), \quad \text{with} \tag{2.13}$$
$$v_k = \pm \frac{1}{(y_k^T d_k)^{1/2} (d_k^T H_k d_k)^{1/2}} y_k - \frac{1}{d_k^T H_k d_k} H_k d_k.$$

This identity implies that H_{k+1} is positive definite if H_k is positive definite and $I + d_k v_k^T$ is nonsingular (i.e., if $v_k^T d_k \neq -1$). As H_{k+1} satisfies the quasi-Newton condition, we have from (2.10) and (2.13) that

$$y_k^T d_k = d_k^T H_{k+1} d_k = d_k^T (I + v_k d_k^T) H_k (I + d_k v_k^T) d_k = (1 + v_k^T d_k)^2 d_k^T H_k d_k.$$

As $y_k^T d_k > 0$ by assumption, it must hold that $(1 + v_k^T d_k)^2 > 0$, which implies that $I + d_k v_k^T$ must be nonsingular. The initial matrix H_0 is usually defined as the identity, in which case the first quasi-Newton direction is identical to a steepest-descent direction.

The conditions imposed on the step length α_k in a quasi-Newton line-search method are more stringent than in a Newton-based method, because of the requirement that $y_k^T d_k > 0$ for every k to retain positive-definiteness. An important advantage of using a line search based on the Wolfe conditions is that $y_k^T d_k$ is always positive. This property is a consequence of the Wolfe step satisfying the inequality $\nabla f(x_{k+1})^T p_k \geq \eta_W \nabla f(x_k)^T p_k$, which is implicitly imposed via the first Wolfe condition. The definition of $y_k^T d_k$ yields

$$y_k^T d_k = \alpha_k \left(\nabla f(x_{k+1})^T p_k - \nabla f(x_k)^T p_k \right) \geq -\alpha_k (1 - \eta_W) \nabla f(x_k)^T p_k > 0.$$

This property does *not* necessarily hold for the Armijo backtracking condition.

2.5. The Davidon-Fletcher-Powell (DFP) Update

The earliest quasi-Newton methods were based on updating an explicit approximation of the inverse Hessian. Let M_k represent an approximation to the *inverse* Hessian. Rather than solving $H_k p_k = -\nabla f(x_k)$ at each iteration, p_k is computed by forming the matrix-vector product

$$p_k = -M_k \nabla f(x_k).$$

where $M_k \approx \left(\nabla^2 f(x_k) \right)^{-1}$. By analogy with (2.2), the quasi-Newton condition for the updated approximate inverse M_{k+1} is

$$M_{k+1} y_k = d_k,$$

and M_{k+1} is obtained by updating M_k. If we work with the inverse and apply the same "simple" derivation used for the BFGS formula we obtain the formula

$$M_{k+1} = M_k - \frac{1}{y_k^T M_k y_k} M_k y_k y_k^T M + \frac{1}{d_k^T y_k} d_k d_k^T,$$

which is the original *Davidon-Fletcher-Powell (DFP) update*.

As we would expect, this update is suggestive of the BFGS update, with d_k replaced by y_k, y_k replaced by d_k and H_k replaced by M_k. In fact, a whole family of interesting relationships can be derived between so-called *complementary* updates. With exact arithmetic, every update to H_k is associated with an equivalent update to M_k that may be computed using the following version of the *Sherman-Morrison-Woodbury formula* (specialized to the symmetric case):

$$(M + \alpha uu^T)^{-1} = M^{-1} - \frac{\alpha}{1 + \alpha u^T M^{-1} u} M^{-1} u u^T M^{-1}.$$

Some tedious manipulation of this formula gives the DFP update in terms of H_k and H_{k+1} as

$$\begin{aligned} H_{k+1} = H_k &- \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \frac{1}{y_k^T d_k} y_k y_k^T \\ &+ \left(d_k^T H_k d_k\right)\left(\frac{1}{y_k^T d_k} y_k - \frac{1}{d_k^T H_k d_k} H_k d_k\right)\left(\frac{1}{y_k^T d_k} y_k - \frac{1}{d_k^T H_k d_k} H_k d_k\right)^T. \end{aligned} \quad (2.14)$$

Given any approximate Hessian H_k, the BFGS and DFP updates to H_k are related by the expression

$$H_{k+1}^{\text{DFP}} = H_{k+1}^{\text{BFGS}} + (d_k^T H_k d_k) w_k w_k^T, \quad \text{where} \quad w_k = \frac{1}{y_k^T d_k} y_k - \frac{1}{d_k^T H_k d_k} H_k d_k. \quad (2.15)$$

This observation is crucial to the discussion of the next section.

2.6. The Broyden class of updates

In this section we consider the definition of a one-parameter family of quasi-Newton updates. It can be verified by direct multiplication that the vector w_k of (2.15) satisfies $w_k^T d_k = 0$. Hence, the symmetric rank-one matrix $w_k w_k^T$ may be added to any matrix H_{k+1} without changing the satisfaction of the quasi-Newton condition $H_{k+1} d_k = y_k$ (2.2). This observation leads to the *Broyden class* of updates

$$H_{k+1} = H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \frac{1}{y_k^T d_k} y_k y_k^T + \phi_k \left(d_k^T H_k d_k\right) w_k w_k^T, \quad (2.16)$$

where the scalar ϕ_k may depend on y_k and $H_k d_k$. The set of updates (2.16) is sometimes called the *Broyden one-parameter family*.

If $\phi_k = 0$, (2.16) is the BFGS update (2.12). If $\phi_k = 1$, (2.16) is the DFP (2.14). If $\phi_k = y_k^T d_k / (y_k^T d_k - d_k^T H_k d_k)$, (2.16) is the symmetric rank-one update (2.11).

2.7. The convex class of updates

The *convex Broyden class* of quasi-Newton updates includes all convex combinations of the BFGS and DFP updates, and is usually written as a function of the parameter φ:

$$H_{k+1}(\varphi) = (1 - \varphi) H_{k+1}^{\text{BFGS}} + \varphi H_{k+1}^{\text{DFP}}, \quad (2.17)$$

where $0 \leq \varphi \leq 1$; $\varphi = 0$ corresponds to the BFGS update, and $\varphi = 1$ corresponds to the DFP update. All updates $H_{k+1}(\varphi)$ (2.17) have the property of hereditary positive-definiteness.

We now consider a specific example to see the effects of different updates. Let

$$H_k = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}, \quad d_k = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \text{ and } y_k = \begin{pmatrix} -3 \\ 2 \end{pmatrix}.$$

The matrix H_k is symmetric and positive definite (with eigenvalues 1 and 3). In addition, $y_k^T d_k > 0$. The (unsymmetric) Broyden update (2.6) is

$$H_{k+1} = H_k + \begin{pmatrix} 0 & 0 \\ -2.5 & -2.5 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ -1.5 & -0.5 \end{pmatrix}.$$

The symmetric rank-one (SR1) update (2.11) gives

$$H_{k+1} = H_k + \begin{pmatrix} 0 & 0 \\ 0 & -5 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & -3 \end{pmatrix},$$

which is symmetric but indefinite (with eigenvalues 2.19 and -3.19). In contrast, the BFGS update (2.12) is

$$H_{k+1} = H_k + \begin{pmatrix} 7.5 & -7.5 \\ -7.5 & 2.5 \end{pmatrix} = \begin{pmatrix} 9.5 & -6.5 \\ -6.5 & 4.5 \end{pmatrix},$$

which is positive definite (with eigenvalues .036 and 13.96). Finally, the DFP update (2.14) is

$$H_{k+1} = H_k + \begin{pmatrix} 45 & -45 \\ -45 & 40 \end{pmatrix} = \begin{pmatrix} 47 & -44 \\ -44 & 42 \end{pmatrix},$$

which is also positive definite (with eigenvalues 0.43 and 88.6). For interest, it is worth verifying that each of these updated matrices H_{k+1} satisfies $H_{k+1} d_k = y_k$.

2.8. Relationships between the updates

An interesting historical note about quasi-Newton methods is that a vast amount of effort was devoted during the late 1960's and early 1970's to devising "new" quasi-Newton updates. Naturally, the researchers developing the updates argued vehemently about which update was "best", and cited extensive numerical experiments to prove that a certain update (usually, their own) was superior to the others. All this controversy ceased with an important theorem, proved by L. C. W. Dixon in 1972, which showed that, under certain conditions, *all* updates in the Broyden class generate *identical iterates* when applied to the same function f, with the same x_0 and H_0, when the step length α_k is taken as the step to the first minimizer of f along p_k. The observed differences in performance were thus attributable to variations in the step length selection strategy rather than to properties of the updates.

Theorem 2.1. (Equivalence of members of the Broyden class) *Let $f(x) \in C^2$, and assume that x_0 and H_0 are given. Let $\{x_k\}$, $\{H_k\}$, $\{p_k\}$ and $\{\alpha_k\}$ denote the sequences generated by the BFGS line-search method, with $\{x_k^\phi\}$, $\{H_k^\phi\}$, $\{p_k^\phi\}$, and $\{\alpha_k^\phi\}$ the corresponding values for the line-search method in which H_{k+1} is defined by any member of the Broyden class (2.16). If each of the sequences $\{H_k\}$ and $\{H_k^\phi\}$ is well-defined, and, for all k, α_k and α_k^ϕ are the minimizers of $f(x_k + \alpha p_k)$ that are nearest to the point $\alpha = 0$, then*

$$x_k^\phi = x_k, \quad \alpha_k^\phi p_k^\phi = \alpha_k p_k \ \text{ and } \ H_k = H_k^\phi + \left(\frac{\phi_{k-1}}{\nabla f(x_{k-1})^T p_k^\phi}\right) \nabla f(x_k) \nabla f(x_k)^T. \quad (2.18)$$

Proof. To limit clutter we use g_k to denote the gradient of f at x_k. The proof is by induction. The use of an exact line search for both methods implies that $y_k^T d_k = -g_k^T d_k$ and $y_k^T d_k^\phi = -g_k^T d_k^\phi$. Using these expressions and the definition $H_k p_k = -g_k$, in the update (2.16) gives

$$H_{k+1}^\phi = H_k^\phi + \frac{1}{g_k^T p_k^\phi} g_k g_k^T - \frac{1}{\alpha_k^\phi g_k^T p_k^\phi} y_k y_k^T - \phi_k \frac{1}{g_k^T p_k^\phi} g_{k+1} g_{k+1}^T. \quad (2.19)$$

As $H_0^\phi \equiv H_0$, it follows that $p_0^\phi = p_0$ and $\alpha_0^\phi = \alpha_0$, and hence that $x_1^\phi = x_1$. The updated matrix (2.19) for $k = 0$ satisfies

$$H_1^\phi = H_1 - \left(\frac{\phi_0}{g_0^T p_0^\phi}\right) g_1 g_1^T,$$

so that (2.18) holds for $k = 1$.

Assume that the result is true at iteration k. The BFGS update is given by (2.19) with $\phi_k = 0$, i.e.,

$$H_{k+1} = H_k + \frac{1}{g_k^T p_k} g_k g_k^T - \frac{1}{\alpha_k g_k^T p_k} y_k y_k^T.$$

The induction hypothesis allows us to substitute for H_k from (2.18), so that

$$H_{k+1} = H_k^\phi + \frac{1}{g_k^T p_k} g_k g_k^T - \frac{1}{\alpha_k g_k^T p_k} y_k y_k^T + \left(\frac{\phi_{k-1}}{g_{k-1}^T p_{k-1}^\phi}\right) g_k g_k^T.$$

Re-arranging terms and using (2.16), we obtain

$$\begin{aligned} H_{k+1} = H_k^\phi + \frac{1}{g_k^T p_k^\phi} g_k g_k^T - \frac{1}{\alpha_k g_k^T p_k} y_k y_k^T - \left(\frac{\phi_k}{g_k^T p_k^\phi}\right) g_{k+1} g_{k+1}^T \\ + \left(\frac{\phi_k}{g_k^T p_k^\phi}\right) g_{k+1} g_{k+1}^T + \left(\frac{\phi_{k-1}}{g_{k-1}^T p_{k-1}^\phi} + \frac{1}{g_k^T p_k} - \frac{1}{g_k^T p_k^\phi}\right) g_k g_k^T. \end{aligned}$$

Applying the induction hypothesis $\alpha_k^\phi p_k^\phi = \alpha_k p_k$ and using (2.19), this relation becomes

$$H_{k+1} = H_{k+1}^\phi + \left(\frac{\phi_k}{g_k^T p_k^\phi}\right) g_{k+1} g_{k+1}^T + \left(\frac{\phi_{k-1}}{g_{k-1}^T p_{k-1}^\phi} + \frac{1}{g_k^T p_k} - \frac{1}{g_k^T p_k^\phi}\right) g_k g_k^T. \quad (2.20)$$

Post-multiplying this expression by $\alpha_k^\phi p_k^\phi$ ($= \alpha_k p_k$), and replacing both $H_{k+1} d_k$ and $H_{k+1}^\phi d_k^\phi$ by y_k gives

$$\frac{\phi_{k-1}}{g_{k-1}^T p_{k-1}^\phi} + \frac{1}{g_k^T p_k} - \frac{1}{g_k^T p_k^\phi} = 0,$$

so that (2.20) gives the required relationship (2.18) between H_{k+1} and H_{k+1}^ϕ.

To establish that $\alpha_{k+1}^\phi p_{k+1}^\phi = \alpha_{k+1} p_{k+1}$, we post-multiply both sides of (2.18) by p_{k+1}^ϕ and substitute $-g_{k+1}$ for $H_{k+1}^\phi p_{k+1}$. Then

$$H_{k+1} p_{k+1}^\phi = \phi_k \left(\frac{g_{k+1}^T p_{k+1}^\phi}{g_k^T p_k^\phi} - 1\right) g_{k+1} = \gamma g_{k+1},$$

where γ is a scalar. As $H_{k+1} p_{k+1} = -g_{k+1}$, and H_{k+1} is nonsingular, p_{k+1} must be a multiple of p_{k+1}^ϕ. This completes the induction. ∎

Theorem 2.1 shows that, when an *exact* line search is used, the iterates generated by different quasi-Newton methods formulas are identical. However, the number of *function evaluations* required for convergence varies significantly among updates, because the trial step length of unity (usually used to initiate the line search) is much closer to the univariate minimizer for some updates than for others.

When implemented with an *inexact* line search, the differences in efficiency among updates become even more striking. Quasi-Newton methods display significant differences in efficiency and reliability when implemented with typical "relaxed" step length criteria. Most researchers today agree that the BFGS method is "best" in practice, but the reasons for its superiority have not yet been fully explained.

2.9. Properties of quasi-Newton methods on a quadratic

One of the fascinating characteristics of certain quasi-Newton methods is their behavior when applied to the quadratic $f(x) = c^T x + \frac{1}{2} x^T H x$, with H symmetric and positive definite. In particular, updates of the Broyden class (2.16) are intricately related (and, in some situations, equivalent) to *conjugate-direction methods*, which arose originally in an entirely different context.

The special feature of quasi-Newton methods to be emphasized here is usually called the *finite termination* property. Suppose that we minimize the quadratic by applying a line-search method with the following features: each iterate is defined as $x_{k+1} = x_k + \alpha_k p_k$; p_k satisfies $H_k p_k = -\nabla f(x_k)$, where H_{k+1} is obtained from H_k using any update from the convex Broyden class (2.17); and α_k is the step from x_k to the (unique) minimizer of $f(x) = c^T x + \frac{1}{2} x^T H x$ along p_k. Then, for any x_0 and

any symmetric positive definite H_0, there exists $m \leq n$ such that $x_m = x^*$. This remarkable result states that a quasi-Newton method will locate the *exact* minimizer of the quadratic in no more than n iterations. Furthermore, if $x_m \neq x^*$ for $m < n$, then $H_n = H$, which means that the final quasi-Newton approximation will be *exact* if the iterations do not terminate "prematurely".

In the following, we prove the results discussed above. Consider the BFGS method with an *exact* line search applied to a quadratic function, i.e., the step length is given by $\alpha_k = -\nabla f(x_k)^T p_k / p_k^T H p_k$. If H_0 is any symmetric positive-definite matrix, the iterates of the BFGS method with an exact line search satisfy

$$\left.\begin{aligned} H_k p_k &= -\nabla f(x_k); \\ x_{k+1} &= x_k + \alpha_k p_k \equiv x_k + d_k; \\ \nabla f(x_{k+1})^T p_k &= 0; \\ H_{k+1} &= H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \frac{1}{y_k^T d_k} y_k y_k^T. \end{aligned}\right\} \tag{2.21}$$

Theorem 2.2. *Suppose that the BFGS algorithm with an exact line search is applied to the quadratic function $f(x) = c^T x + \frac{1}{2} x^T H x$. Then, for any k $(1 \leq k \leq n)$,*

$$\begin{aligned} H_k d_j &= H d_j, \qquad 0 \leq j < k; \\ d_j^T H d_i &= 0, \qquad 0 \leq j < k, \quad 0 \leq i < k, \quad i \neq j. \end{aligned}$$

Proof. Algorithm (2.21) terminates when $\nabla f(x_j) = 0$, and therefore we assume that $\nabla f(x_j) \neq 0$ for $j = 0, 1, \ldots, k$.

As f is quadratic and the quasi-Newton condition is satisfied at every iteration, we have

$$y_j = H d_j = H_{j+1} d_j. \tag{2.22}$$

An inductive argument is used to show that

$$H_k d_j = H d_j \quad 0 \leq j \leq k-1 \tag{2.23a}$$

$$d_i^T H d_j = 0, \qquad 0 \leq i \leq k, \qquad 0 \leq j \leq k, \quad i \neq j. \tag{2.23b}$$

The quasi-Newton condition implies that $H_1 d_0 = y_0$. Using (2.21) and (2.22) for $j = 0$, the result is

$$d_1^T H d_0 = d_1^T y_0 = d_1^T H_1 d_0 = \alpha_1 p_1^T H_1 d_0 = -\alpha_1 \nabla f(x_1)^T d_0 = 0.$$

It follows that (2.23) holds for $k = 1$.

Assume now that (2.23) is true for k $(1 < k \leq n)$. Then, given $j < k$, we substitute the expression $y_j = H d_j$ and use the induction hypothesis (2.23b) to

obtain

$$\begin{aligned} H_{k+1}d_j &= H_k d_j - H_k d_k \frac{d_k^T H_k d_j}{d_k^T H_k d_k} + y_k \frac{y_k^T d_j}{y_k^T d_k} \\ &= y_j - H_k d_k \frac{d_k^T H d_j}{d_k^T H_k d_k} + y_k \frac{d_k^T H d_j}{y_k^T d_k} \\ &= y_j = H d_j. \end{aligned} \tag{2.24}$$

This verifies that (2.23a) holds for $k+1$.

As f is quadratic, the definition of $\{x_k\}$ gives

$$\nabla f(x_{k+1}) = c + H x_{k+1} = c + H x_{j+1} + H(d_{j+1} + \cdots + d_k).$$

The inner product of this expression with d_j gives

$$\nabla f(x_{k+1})^T d_j = \nabla f(x_{j+1})^T d_j + d_j^T H(d_{j+1} + d_{j+2} + \cdots + d_k).$$

Substituting the identities $d_k = \alpha_k p_k$, $\nabla f(x_{k+1})^T p_k = 0$, and using the inductive hypothesis $d_j^T H d_k = 0$, yields $\nabla f(x_{k+1})^T d_j = 0$ for $j < k$. Using (2.24), (2.22) and the relation $y_j = H d_j$ in this identity gives

$$d_{k+1}^T H d_j = d_{k+1}^T y_j = d_{k+1}^T H_{k+1} d_j = -\alpha_{k+1} \nabla f(x_{k+1})^T d_j = 0. \tag{2.25}$$

This completes the induction. ∎

Given a symmetric, positive-definite matrix H, a set of non-zero vectors $\{d_j\}$ such that

$$d_i^T H d_j = 0, \qquad i \neq j,$$

are said to be *conjugate* with respect to the matrix H. Conjugate vectors are always linearly independent, which may be used to establish the following theorem.

Theorem 2.3. (Finite termination of the Broyden class)
If the BFGS method defined by (2.21) is applied to the quadratic function $f(x) = c^T x + \frac{1}{2} x^T H x$, there exists an integer m $(m \leq n)$ such that $\nabla f(x_m) = 0$. Moreover, if $\nabla f(x_k) \neq 0$ for $k = 0, 1, \ldots, n-1$, then $H_n = H$.

Proof. If $\nabla f(x_k) = 0$ for any $k < n$, the theorem holds. Therefore, we assume that $\nabla f(x_k) \neq 0$ for $k = 0, 1, \ldots, n-1$. As $\alpha_n > 0$, relation $d_{k+1}^T H d_j = 0$ (2.25) shows that

$$\nabla f(x_n)^T d_j = 0 \ \text{ for } \ j = 0, 1, \ldots, n-1.$$

As the vectors $\{ d_j \}$ are linearly independent, these relations imply that $\nabla f(x_n)$ is orthogonal to n linearly independent vectors, which can be the case only if $\nabla f(x_n) = 0$.

To establish the second part of the theorem, let S denote the matrix with j-th column d_j. Then (2.24) for $k = n-1$ can be written as $H_n D = HD$, because $y_i = H d_i$. Because D is non-singular, this last relation implies that $H_n = H$. ∎

This result is somewhat surprising, especially because intermediate quasi-Newton approximations may bear little resemblance to the exact Hessian, and may appear unrelated to other updates from the convex Broyden class. For example, consider the quadratic $c^T x + \frac{1}{2} x^T H x$ with

$$c = \begin{pmatrix} \frac{1}{2} \\ \frac{7}{22} \end{pmatrix}, \qquad H = \begin{pmatrix} 11 & -8 \\ -8 & 6 \end{pmatrix}.$$

If the initial iterate is

$$x_0 = \begin{pmatrix} \frac{8}{11} \\ 1 \end{pmatrix}, \qquad H_0 = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix},$$

then

$$\nabla f(x_0) = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \quad p_0 = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \quad \alpha_0 = 1,$$

and the first iteration gives

$$x_1 = \begin{pmatrix} -\frac{3}{11} \\ 0 \end{pmatrix}, \quad \nabla f(x_1) = \begin{pmatrix} -\frac{5}{2} \\ \frac{5}{2} \end{pmatrix}, \text{ and } y_0 = \begin{pmatrix} -3 \\ 2 \end{pmatrix}.$$

Using (2.12) and (2.14), the associated BFGS and DFP updates (rounded to four figures) are

$$H_1^{\text{BFGS}} = \begin{pmatrix} 9.083 & -6.083 \\ -6.083 & 4.083 \end{pmatrix}, \qquad H_1^{\text{DFP}} = \begin{pmatrix} 15.33 & -12.33 \\ -12.33 & 10.33 \end{pmatrix}. \tag{2.26}$$

Using H_1^{BFGS} to compute the search direction gives

$$p_1 = \begin{pmatrix} -60 \\ -90 \end{pmatrix}, \quad \alpha_1 = .04167, \text{ and } x_2 = \begin{pmatrix} -2.7727 \\ -3.75 \end{pmatrix},$$

which is the exact minimizer. At this point, both H_2^{BFGS} and H_2^{DFP} are equal to H. However, neither intermediate matrix in (2.26) is "close" to H, and that H_1^{BFGS} and H_1^{DFP} are quite different from one another.

In contrast, if we choose

$$x_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad H_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

then the first iteration yields

$$x_1 = \begin{pmatrix} 1.6820 \\ 0.4560 \end{pmatrix}, \qquad H_1^{\text{BFGS}} = \begin{pmatrix} 11.27 & -7.59 \\ -7.59 & 10.61 \end{pmatrix},$$

and H_1^{DFP} is equal to H_1^{BFGS} to within four figures. (In addition, the very first quasi-Newton approximation is similar to H.)

Despite this encouraging result for the quadratic case, a quasi-Newton approximation to the Hessian of a general nonlinear function need not resemble the Hessian at the solution. In particular, if $f(x)$ is *separable* and a certain variable is set initially to its optimal value, the corresponding diagonal element of the Hessian approximation will never change. However, the superlinear convergence properties of quasi-Newton methods mean that H_k must eventually resemble the true Hessian along certain critical directions.

3. Matrix Factorization

In this section we describe how to maintain a factorization of the approximate Hessian $H_k = R_k^T R_k$ where R_k is upper triangular. This turns out to be more expensive computationally than maintaining the inverse approximation $M_k \approx \nabla^2 f(x_k)^{-1}$ but substantially more reliable.

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 solving $p_k = -M_k \nabla f(x_k)$ or $H_k p_k = -\nabla f(x_k)$ 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 condition number or the eigenvalues 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 bound

$$\operatorname{cond}(R_k) \geq \frac{\max |r_{jj}|}{\min |r_{jj}|}.$$

Based on this bound, if $\operatorname{cond}(R_k)^2$ becomes large, then the (implicit) approximate Hessian H_k is reset.

3.1. Factored Hessian BFGS (`bfgsR`)

Next we derive a variant of the BFGS method that maintains the Cholesky factor described in the following result.

Result 3.1. (The Cholesky Factorization) *If H is positive definite then there exists a unique $n \times n$ upper-triangular matrix R such that $H = R^T R$.*

Given the Cholesky factorization $H_k = R_k^T R_k$, we can write the system $H_k p_k = -\nabla f(x_k)$ as

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

To obtain the solution p_k, first solve the lower-triangular system $R_k^T z_k = -\nabla f(x_k)$ followed by the upper-triangular system $R_k p_k = z_k$. In general, given

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

we need to compute R_{k+1} such that $H_{k+1} = R_{k+1}^T R_{k+1}$. In order to do this, write

$$H_{k+1} = (I - v_k d_k^T) H_k (I - d_k v_k^T) \quad \text{where} \quad v_k = \frac{1}{d_k^T H_k d_k} H_k d_k - \frac{1}{(y_k^T d_k)^{\frac{1}{2}} (d_k^T H_k d_k)^{\frac{1}{2}}} y_k.$$

Assuming that, at the kth iteration, we have the factorization $H_k = R_k^T R_k$, then

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

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

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

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 $\widehat{R}_k$ then becomes

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

or simply $\widehat{R}_k = R_k + u_k v_k^T$ with $u_k = R_k d_k / \|R_k d_k\|$ and $v_k = y_k/(y_k^T d_k)^{\frac{1}{2}} - R_k^T u_k$.

The matrix $\widehat{R}_k = R_k + u_k v_k^T$ has the desired property that $\widehat{R}_k^T \widehat{R}_k = H_{k+1}$, however it is *not* upper-triangular. This is corrected by premultiplying $\widehat{R}_k$ by a sequence of orthogonal plane rotations, whose product we denote by Q_k, so that $Q_k \widehat{R}_k$ *is* upper-triangular. If we let $R_{k+1} = Q_k \widehat{R}_k$ then R_{k+1} is upper triangular and

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

At each iteration we need only solve two triangular systems for p_k, compute $\widehat{R}_k$, and use orthogonal plane rotations to arrive at R_{k+1}.

Algorithm 1 `bfgsR`: 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 $R_k^T z_k = -\nabla f(x_k)$; Solve $R_k p_k = z_k$;
  $\alpha_k = \texttt{Wolfe}(x_k, f(x_k), \nabla f(x_k), p_k)$;
  $d_k = x_k + \alpha_k p_k$; $x_{k+1} = x_k + d_k$;
  $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
  if $y_k^T d_k > -\mu(1-\eta_W)\alpha_k \nabla f(x_k)^T p_k$ **then**
    $u_k = \dfrac{R_k d_k}{\|R_k d_k\|}$; $v_k = \dfrac{1}{(y_k^T d_k)^{\frac{1}{2}}} y_k - R_k^T u_k$;
    $Q_k = \texttt{planerot}(R_k, u_k, v_k)$;
    $R_{k+1} = Q_k R_k$;
  else
    $R_{k+1} = R_k$;
  end if
  $k = k + 1$;
end while

4. Self-Scaled Updates

An important factor in the numerical performance of quasi-Newton methods is the conditioning (i.e., the magnitude of the matrix condition number) of each approximate Hessian 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 [17]. Addition references include Oren [14–16], and Oren and Spedicato [18],

Scaling the BFGS Hessian for a quadratic. The influence of H_0 may be seen by considering the BFGS method applied to the strictly convex quadratic $f(x) = c^T x + \frac{1}{2} x^T H x$. If an exact line search is used, then Result 2.2 shows that $H_k d_j = H d_j$, for $0 \leq j < k$. As H is nonsingular, we get $H^{-1} H_k d_j = d_j$, which implies that $H^{-1} H_k$ has k unit eigenvalues corresponding to the k eigenvectors $\{ d_j \}$, $0 \leq j < k$. 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 spectrum of $H^{-1} H_{k+1}$ is 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

$$H^{-1}H_j = H^{-1/2}\big(H^{-1/2}H_jH^{-1/2}\big)H^{1/2}, \tag{4.1}$$

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

Result 4.1. *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*

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

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 (4.1) 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 (2.12) may be written as an update to the matrix Q_k, i.e.,

$$Q_{k+1} = Q_k - \frac{1}{q_k^TQ_kq_k}Q_kq_kq_k^TQ_k + \frac{1}{q_k^Tq_k}q_kq_k^T = P_k + \frac{1}{q_k^Tq_k}q_kq_k^T,$$

where

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

Direct multiplication gives $P_kq_k = 0$ and q_k is an eigenvector of P_k with zero eigenvalue. If the eigenvalues of P_k are denoted by $\mu_1, \mu_2, \ldots, \mu_n$, then

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

and the eigenvalue interlacing theorem gives

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

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 $\mu_{n-1}, \ldots, \mu_1$. But $1 \in [\lambda_n, \lambda_1]$, and $\lambda_n \leq \mu_{n-1} \leq \cdots \leq \mu_1 \leq \lambda_1$, which implies that $1, \mu_{n-1}, \ldots, \mu_1 \in [\lambda_n, \lambda_1]$, as required. ∎

The result is easily extended to the Broyden convex class.

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 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 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 $\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n$ 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 $0 < \lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$ and γ is any value such that $1/\lambda_1 \leq \gamma \leq 1/\lambda_n$, then the eigenvalues of $H^{-1}(\gamma H_0)$ are $0 < \gamma\lambda_n \leq \gamma\lambda_{n-1} \leq \cdots \leq \gamma\lambda_1$, with

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

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

4.1. Self-Scaled Explicit BFGS (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 (2.12) we obtain

$$H_{k+1} = \gamma_k\Big(H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k\Big) + \frac{1}{y_k^T d_k} y_k y_k^T. \tag{4.2}$$

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 $0 < \lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$, then any γ_k such that $1/\lambda_1 \leq \gamma \leq 1/\lambda_n$ will give

$$0 < \gamma_k\lambda_n \leq 1 \leq \gamma_k\lambda_1 \quad \text{and} \quad 1 \in [\gamma_k\lambda_n, \gamma_k\lambda_1]. \tag{4.3}$$

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

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

It follows that in the nonlinear case, if we choose $\gamma_k = y_k^T d_k / d_k^T H_k d_k$ in (4.2) then (4.3) 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.

These results suggest a two-parameter family of updates:

$$H_{k+1} = \gamma_k\Big(H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k\Big) + \frac{1}{y_k^T d_k} y_k y_k^T + \gamma_k \theta_k \, (d_k^T H_k d_k) w_k w_k^T,$$

with $w_k = (1/y_k^T d_k) y_k - 1/(d_k^T H_k d_k) H_k d_k$, and $0 \le \theta_k \le 1$.

The choice of γ_k is extended Oren in [16] to be a convex combination

$$\gamma_k = \varphi \frac{\nabla f(x_k)^T y_k}{\nabla f(x_k)^T H_k d_k} + (1-\varphi)\frac{y_k^T d_k}{d_k^T H_k d_k}, \tag{4.4}$$

where $\varphi \in [0,\ 1]$.

Algorithm 2 `bfgsHS`: Self-scaled Explicit BFGS

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 $y_k^T d_k > -\mu(1-\eta_W)\alpha_k \nabla f(x_k)^T p_k$ **then**
    Pick $\theta, \varphi \in [0,\ 1]$;
    $\gamma_k = \varphi \dfrac{\nabla f(x_k)^T y_k}{\nabla f(x_k)^T H_k d_k} + (1-\varphi)\dfrac{y_k^T d_k}{d_k^T H_k d_k}$;
    $w_k = \dfrac{1}{y_k^T d_k} y_k - \dfrac{1}{d_k^T H_k d_k} H_k d_k$;
    $H_{k+1} = \gamma_k \left(H_k - \dfrac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k\right) + \dfrac{1}{y_k^T d_k} y_k y_k^T + \gamma_k \theta_k \, (d_k^T H_k d_k) w_k w_k^T$;
  else
    $H_{k+1} = H_k$;
  end if
  $k = k + 1$;
end while

4.2. Self-Scaled Inverse BFGS (`bfgsMS`)

The equivalent update for the inverse is

$$M_{k+1} = \gamma_k \left(M_k - \frac{1}{y_k^T M_k y_k} M_k y_k y_k^T M_k\right) + \frac{1}{d_k^T y_k} d_k d_k^T + \gamma_k \theta_k \, (y_k^T M_k y_k) u_k u_k^T,$$

with $u_k = (1/d_k^T y_k) d_k - 1/(y_k^T M_k y_k) M_k y_k$, $0 \le \theta_k \le 1$, and γ_k chosen similarly to self-scaled explicit BFGS (4.4) but with the inverse equivalent

$$\gamma_k = \varphi \frac{\nabla f(x_k)^T d_k}{\nabla f(x_k)^T M_k y_k} + (1-\varphi)\frac{y_k^T d_k}{y_k^T M_k y_k}, \quad \text{for } \varphi \in [0,\ 1].$$

Algorithm 3 bfgsMS: Self-scaled Inverse BFGS

Choose $x_0 \in \mathbb{R}^n$; $k \leftarrow 0$;
while $\|\nabla f(x_k)\| > \varepsilon$ and $k \leq N$ **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 $y_k^T d_k > -\mu(1-\eta_W)\alpha_k \nabla f(x_k)^T p_k$ **then**
    Pick $\theta, \varphi \in [0,\ 1]$;
    $\gamma_k = \varphi \dfrac{\nabla f(x_k)^T d_k}{\nabla f(x_k)^T M_k y_k} + (1-\varphi)\dfrac{y_k^T d_k}{y_k^T M_k y_k}$;
    $u_k = \dfrac{1}{d_k^T y_k} d_k - \dfrac{1}{y_k^T M_k y_k} M_k y_k$;
    $M_{k+1} = \gamma_k \left(M_k - \dfrac{1}{y_k^T M_k y_k} M_k y_k y_k^T M_k \right) + \dfrac{1}{d_k^T y_k} d_k d_k^T + \gamma_k \theta_k \, (y_k^T M_k y_k) u_k u_k^T$;
  else
    $M_{k+1} = M_k$;
  end if
  $k = k + 1$;
end while

4.3. Self-Scaled Factored BFGS (bfgsRS)

All that is needed to apply the scaling described in Section 4 is to use the square root of the scaling factor. Since $H_k = R_k^T R_k$, 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 does not increase the operation count significantly.

5. Modified quasi-Newton 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 classic quasi-Newton condition (2.2), 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.

5.1. Function Interpolation (bfgsHY)

Consider the quadratic model defined by $q_k(p) = f(x_k) + p^T \nabla f(x_k) + \frac{1}{2} p^T H_k p$. The search direction obtained by solving $H_k p_k = -\nabla f(x_k)$ is also the solution of the

Algorithm 4 `bfgsRS`: Self-Scaled 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 $R_k^T z_k = -\nabla f(x_k)$; Solve $R_k p_k = z_k$;
  $\alpha_k = \texttt{Wolfe}(x_k, f(x_k), \nabla f(x_k), p_k)$;
  $d_k = \alpha_k p_k$; $x_{k+1} = x_k + d_k$;
  $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$;
  if $y_k^T d_k > -\mu(1-\eta_W)\alpha_k \nabla f(x_k)^T p_k$ **then**
    $\gamma_k = \dfrac{y_k^T d_k}{\|R_k d_k\|}$; $R_k = \gamma_k^{\frac{1}{2}} R_k$;
    $u_k = \dfrac{1}{\|R_k d_k\|} R_k d_k$; $v_k = \dfrac{1}{(y_k^T d_k)^{\frac{1}{2}}} y_k - R_k^T u_k$;
    $Q_k = \texttt{planerot}(R_k, u_k, v_k)$;
    $R_{k+1} = Q_k R_k$;
  else
    $R_{k+1} = R_k$;
  end if
  $k = k + 1$;
end while

following quadratic subproblem:

$$\min_{p \in \mathbb{R}^n} \; q_k(p).$$

For small values of p, $q_k(p)$ is approximately equal to $f(x_k + p)$, in particular, $q_k(p)$ satisfies the following standard interpolation conditions.

$$q_k(0) = f(x_k), \qquad \nabla q_k(0) = \nabla f(x_k), \quad \text{and} \quad \nabla^2 q_k(0) = H_k. \tag{5.1}$$

If the quasi-Newton condition (2.2) holds at x_{k-1} then the interpolation conditions (5.1) imply

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

These properties show 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

$$H_{k+1} = H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \gamma_k \frac{1}{y_k^T d_k} y_k y_k^T. \tag{5.2}$$

In order to determine γ_k we need to impose an extra condition. The quasi-Newton variant derived by Yuan in [22] is based on the condition

$$q_k(x_{k-1} - x_k) = f(x_{k-1}). \tag{5.3}$$

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 (5.1) 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 (5.3) it becomes $q_{k+1}(x_k - x_{k+1}) = f(x_k)$. Rewriting the quadratic model with k increased by one as well yields $q_{k+1}(p) = f(x_{k+1}) + p^T\nabla f(x_{k+1}) + \frac{1}{2}p^T H_{k+1}p$. Substituting $p = x_k - x_{k+1}$ into the model and setting the result equal to $f(x_k)$, i.e., applying (5.3), gives

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

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

$$d_k^T H_{k+1}d_k = 2\left(f(x_k) - f(x_{k+1}) + \nabla f(x_{k+1})^T d_k\right). \tag{5.4}$$

We want to update the approximate Hessian H_k using

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

so that the expression for $d_k^T H_{k+1}d_k$ (5.4) is true. To determine what value of γ_k will accomplish this, left- and right-multiply the scaled BFGS update (5.2) by d_k to get

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

Solving this expression for γ_k and substituting (5.4) into the result gives the scaling factor

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

By using this value of γ_k in the scaled update (5.2) we arrive at Algorithm `bfgsHY` below.

5.2. Adaptive Scaling (`bfgsMN`)

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

Recall the conjugacy condition derived in Section 2.9 for BFGS method applied to a quadratic function $f(x) = c^T x + \frac{1}{2}x^T H x$ given by

$$d_i^T H d_j = 0, \qquad i \neq j, \tag{5.5}$$

Algorithm 5 `bfgsHY`: BFGS with Function Interpolation

Choose $x_0 \in \mathbb{R}^n$; $k \leftarrow 0$;
while $\|\nabla f(x_k)\| > \varepsilon$ and $k \leq N$ **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 $y_k^T d_k > -\mu(1-\eta_W)\alpha_k \nabla f(x_k)^T p_k$ **then**
    $\gamma_k = \dfrac{2}{y_k^T d_k}\left(f(x_k) - f(x_{k+1}) + \nabla f(x_{k+1})^T d_k\right)$;
    $\widetilde{y}_k = \gamma_k y_k$;
    $H_{k+1} = H_k - \dfrac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \dfrac{1}{\widetilde{y}_k^T d_k}\widetilde{y}_k \widetilde{y}_k^T$;
  else
    $H_{k+1} = H_k$;
  end if
  $k = k + 1$;
end while

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

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

If this is to satisfy the conjugacy condition (5.5) then it may be reasonable to seek to achieve $d_{k+1}^T y_k = 0$. Since $\alpha_{k+1} \neq 0$ this amounts to

$$p_{k+1}^T y_k = 0. \tag{5.6}$$

The quasi-Newton condition for the inverse approximate Hessian update, given by $d_k = M_{k+1} y_k$, can be combined with (5.6) to write

$$\begin{aligned} p_{k+1}^T y_k &= (-M_{k+1} \nabla f(x_{k+1}))^T y_k \\ &= -\nabla f(x_{k+1})^T (M_{k+1} y_k) \\ &= -\nabla f(x_{k+1})^T d_k. \end{aligned}$$

Therefore one objective is to find a scaling factor γ_k so that when y_k is scaled to $\gamma_k y_k$ the magnitude of $-\gamma_k \nabla f(x_{k+1})^T d_k$ 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 $\gamma_k \leq 1$.

Most likely these two objectives cannot be satisfied simultaneously, so instead we seek the best compromise. Our strategy is to select γ_k as

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

which yields

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

With γ_k selected, the adaptive-scaled update to the Hessian approximation is

$$H_{k+1} = H_k - \frac{1}{d_k^T H_k d_k} H_k d_k d_k^T H_k + \gamma_k \frac{1}{y_k^T d_k} y_k y_k^T. \tag{5.7}$$

A critical property of the BFGS update is that if H_k is positive definite then so is H_{k+1}. In the next result we see that the proposed scaling preserves hereditary positive definiteness

Result 5.1. (bfgsMN: 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 (5.7) is also positive definite.*

Proof. If $x \neq 0$ then by the Cauchy-Schwarz inequality we have

$$(d_k^T H_k x)^2 \leq (d_k^T H_k d_k)(x^T H_k x).$$

Conjugating the scaled inverse update (5.7) by x gives

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

Therefore $x^T H_{k+1} x > 0$ for all $x \neq 0$, so H_{k+1} is positive definite. ∎

Using the Sherman-Morrison-Woodbury formula, we arrive at the corresponding rank-two update to the approximate inverse Hessian

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

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

Algorithm 6 `bfgsMN`: Adaptive, Scaled BFGS with a Wolfe Line Search

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 $y_k^T d_k > -\mu(1-\eta_W)\alpha_k \nabla f(x_k)^T p_k$ **then**
    $\gamma_k = \min\left(\dfrac{y_k^T d_k}{\|y\|^2 + |d_k^T \nabla f(x_{k+1})|}, 1\right)$;
    $\widetilde{y}_k = \gamma_k y_k$;
    $M_{k+1} = M_k - \dfrac{1}{\widetilde{y}_k^T M_k \widetilde{y}_k} M_k \widetilde{y}_k \widetilde{y}_k^T M_k + \dfrac{1}{\widetilde{y}_k^T d_k} d_k d_k^T + (\widetilde{y}_k^T M_k \widetilde{y}_k) w_k w_k^T$;
  else
    $M_{k+1} = M_k$;
  end if
  $k = k + 1$;
end while

5.3. Multistep quasi-Newton Equations (`bfgsMZ`)

The next method to be considered is the result of deriving modified quasi-Newton equations, rather than a new method satisfying the original quasi-Newton condition (2.2). At each iteration, the data

$$f(x_k), \quad f(x_{k+1}), \quad \nabla f(x_k), \quad \text{and} \quad \nabla f(x_{k+1}), \tag{5.8}$$

are all available, but are not fully utilized by other methods. The approach developed by Zhang in [23] is to arrive at a better approximation to $\nabla^2 f(x_k) d_k$ by using the data (5.8) to scale y_k to some $\widetilde{y}_k$, so that the quasi-Newton condition $H_{k+1} d_k = \widetilde{y}_k$ imparts more accuracy to the Hessian approximation.

The path from x_k to x_{k+1} can be parameterized with 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

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

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

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

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

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

which gives us a third condition

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

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

$$\left.\begin{aligned} h(0) &= \nabla f(x_k), \\ h(\|d_k\|) &= \nabla f(x_{k+1}), \\ \int_0^{\|d_k\|} h(t)^T x'(t)\,dt &= f(x_{k+1}) - f(x_k). \end{aligned}\right\} \tag{5.9}$$

Requirements (5.9) 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

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

which follows from

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

Now define

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

The simplest choice for a that satisfies $\|d_k\|^2 a^T d_k = \gamma_k$ 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 new quasi-Newton condition analogous to the conventional quasi-Newton condition (2.2)

$$\begin{aligned}
H_{k+1} d_k &= \widetilde{y}_k, \\
\widetilde{y}_k &= y_k + \frac{\gamma_k}{\|d_k\|^2} d_k, \\
\gamma_k &= 3(\nabla f(x_k) + \nabla f(x_{k+1}))^T d_k - 6(f(x_{k+1}) - f(x_k)).
\end{aligned} \tag{5.10}$$

It is clear that all the data (5.8) is being incorporated by the method if the modified quasi-Newton condition (5.10) is enforced at each iteration.

One important consideration is preserving the property $y_k^T d_k > 0$. If y_k is scaled by the new condition (5.10) then

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

For numerical stability, if $y_k^T d_k + \gamma_k < \tilde{\varepsilon} \|d_k\|^2$ then γ_k is set to zero and the unscaled y_k is used for the remainder of the iteration. A value of $\tilde{\varepsilon} = 10^{-18}$ was used by Zhang in [23] and in our implementation. (See Algorithm `bfgsMZ` below.)

6. 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 $n_{\mathcal{P}} = 234$. 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 6.2), 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.

Algorithm 7 `bfgsMZ`: Multistep, Scaled BFGS with a Wolfe Line Search

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 $y_k^T d_k > -\mu(1-\eta_w)\alpha_k \nabla f(x_k)^T p_k$ **then**
    $\gamma_k = 3(\nabla f(x_{k+1}) + \nabla f(x_k))^T - 6(f(x_{k+1}) + f(x_k))$;
    if $y_k^T d_k + \gamma_k < \tilde{\varepsilon}\|d_k\|^2$ **then**
      $\gamma_k = 0$;
    end if
    $\widetilde{y}_k = y_k + \gamma_k \dfrac{1}{\|d_k\|^2} d_k$; $w_k = \dfrac{1}{\widetilde{y}_k^T d_k} d_k - \dfrac{1}{\widetilde{y}_k^T M_k \widetilde{y}_k} M_k \widetilde{y}_k$;
    $M_{k+1} = M_k - \dfrac{1}{\widetilde{y}_k^T M_k \widetilde{y}_k} M_k \widetilde{y}_k \widetilde{y}_k^T M_k + \dfrac{1}{\widetilde{y}_k^T d_k} d_k d_k^T + (\widetilde{y}_k^T M_k \widetilde{y}_k) w_k w_k^T$;
  else
    $M_{k+1} = M_k$;
  end if
  $k = k + 1$;
end while

6.1. 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 paper we use *performance profiles*, described in [5].

For each $p \in \mathcal{P}$ and $s \in \mathcal{S}$, define $m_{p,s}$ 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 $\rho_s : \mathbb{R} \to [0, 1]$ 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, $\rho_s(\tau)$ is 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.

6.2. Hardware and Software

The code for each solver was written in MATLAB version R2019b by Jeb H. Runnoe. 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 2. 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`		
Update rejection tolerance μ	`1.00e-08`		
Unbounded objective	`-1.00e+09`		

Table 2: 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.

6.3. 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 illustrate this trade off.

The comparison of standard BFGS with the factored Hessian variation highlights the importance of numerical stability in the update formula. The fact that the relation $H_k = R_k^T R_k$ is an *equality* means that in infinite precision arithmetic `bfgsH` and `bfgsR` should perform identically. In practice they do not, and the difference in their performance seen in Figure 2 is owed entirely to the management of numerical factors like conditioning and indefiniteness, made possible by the use of matrix factorization.

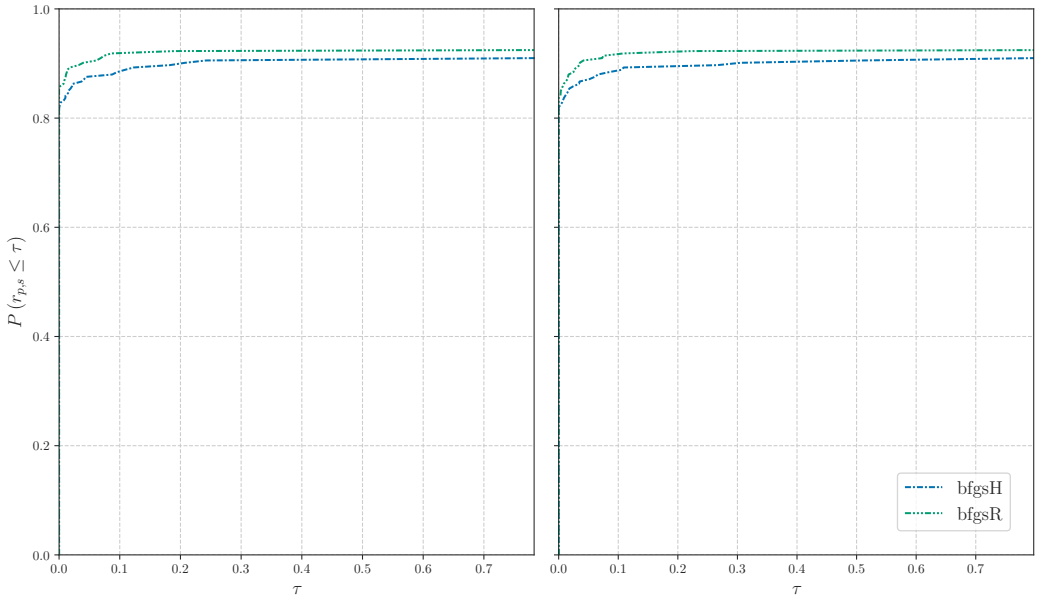

Figure 2: Performance profiles comparing function evaluations (left) and iterations (right) for `bfgsH` and `bfgsR`.

The self-scaled updates, whose purpose is to limit the negative effects of ill-conditioning, provide another example of the benefits of taking stability into consideration. Whether the explicit, inverse, or factored Hessian approximation is used as the base method, self-scaling improves overall performance. Notice that in Figure 3 the self-scaled method's performance in iterations is arguably worse than the standard. This is an example of the function-iteration trade off discussed at the beginning of this section. The improvement for the inverse method `bfgsMS` is more consistent than that of `bfgsHS`. Notice also that iteration performance `bfgsMS` does not suffer from scaling.

Out of all the variants considered in this thesis, the self-scaled factored Hessian `bfgsRS` is the highest performing method (see Figure 5). As `bfgsRS` makes use of both self-scaling to manage conditioning, and resetting in the event of indefiniteness, this supports the idea that numerical stability makes a real difference.

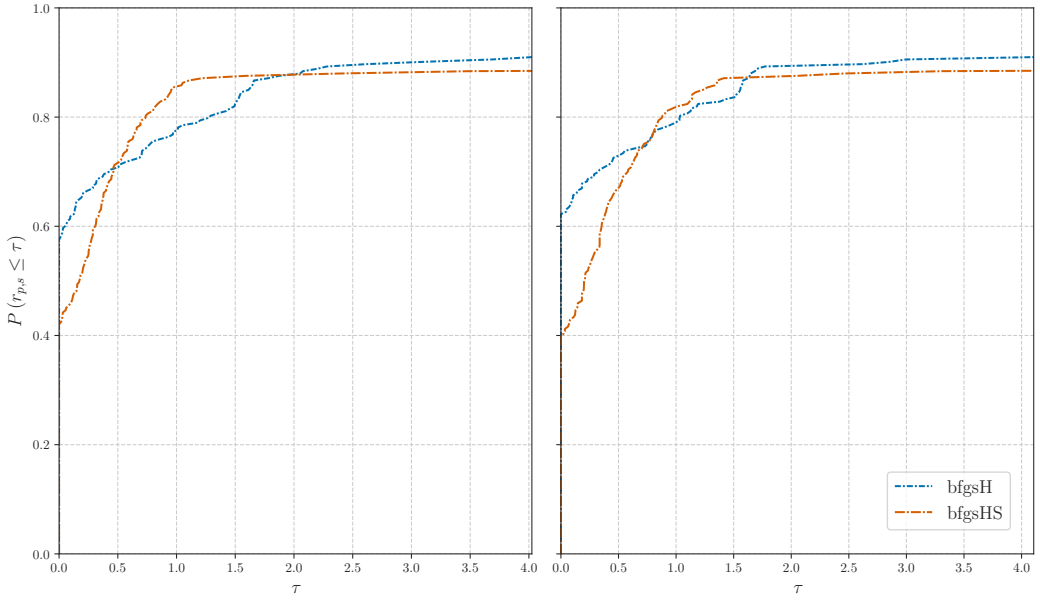

Figure 3: Performance profiles comparing function evaluations (left) and iterations (right) for `bfgsH` and `bfgsHS`.

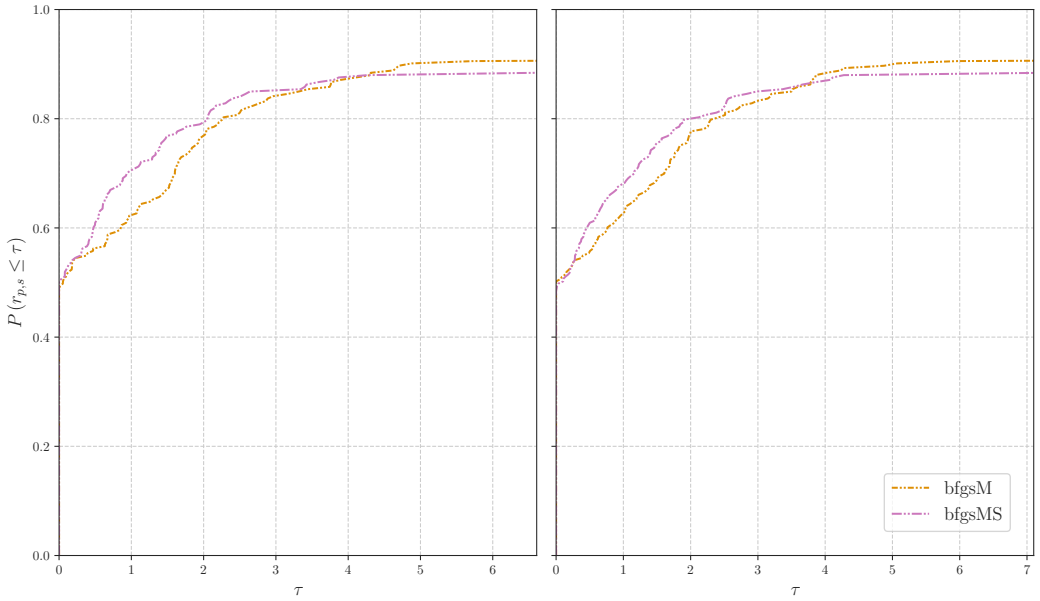

Figure 4: Performance profiles comparing function evaluations (left) and iterations (right) for `bfgsM` and `bfgsMS`.

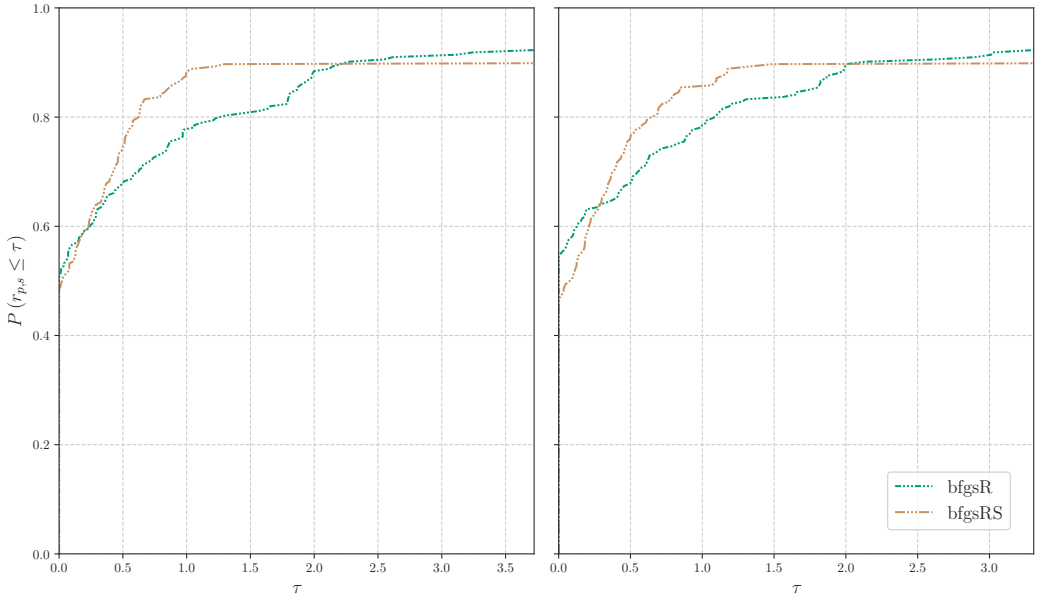

Figure 5: Performance profiles comparing function evaluations (left) and iterations (right) for `bfgsR` and `bfgsRS`.

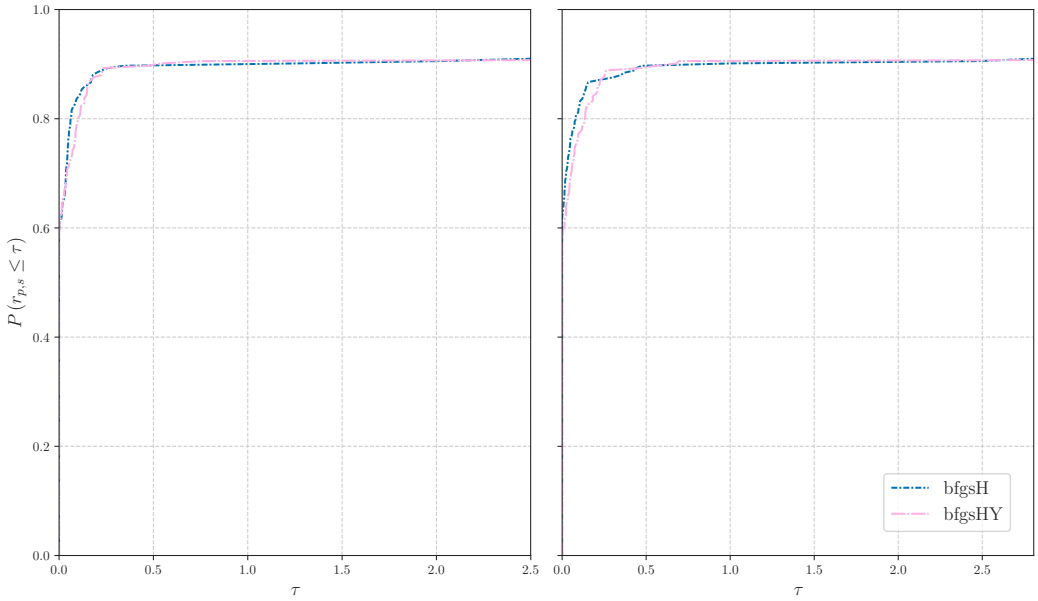

Figure 6: Performance profiles comparing function evaluations (left) and iterations (right) for `bfgsH` and `bfgsHY`.

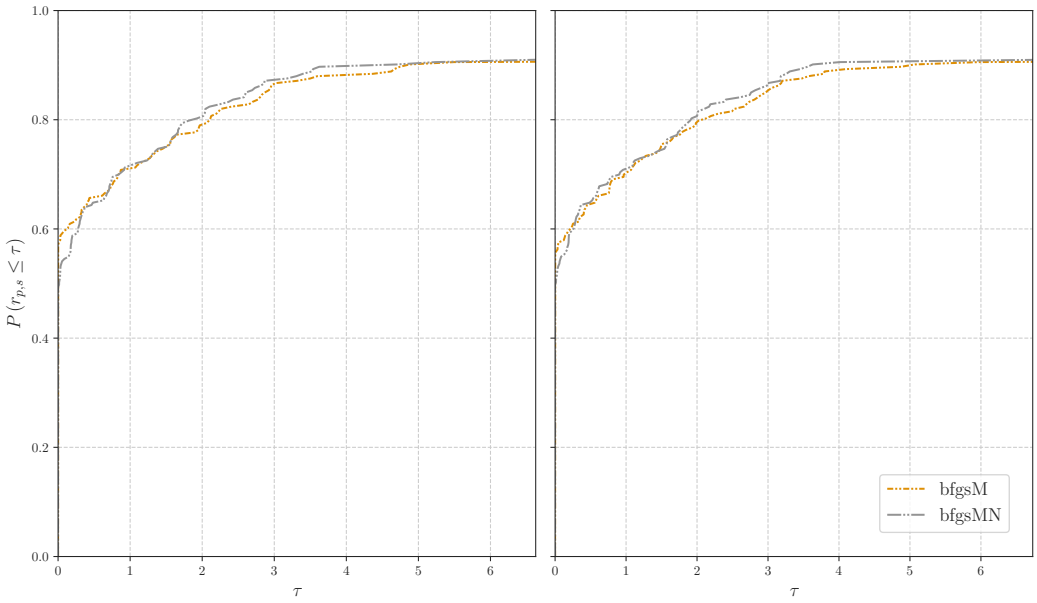

Figure 7: Performance profiles comparing function evaluations (left) and iterations (right) for `bfgsM` and `bfgsMN`.

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 `bfgsHY`, `bfgsMN`, and `bfgsMZ` are all implemented by scaling or shifting y_k. Looking at 6, it seems imposing an extra interpolation condition seems to have little measurable effect.

Similarly, in Figure 7 `bfgsMN` shows slight improvement in both function evaluations and iterations, but the improvement is negligible compared to other methods. 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, then `bfgsMN` appears to be far superior. However, if the test set is expanded to the full 234 problems then it is clear that there is actually no significant difference. Note that Figure 8 is different from the other figures in that both the left and right plots are CPU time profiles, but the left is on the full set while the right is just the common set.

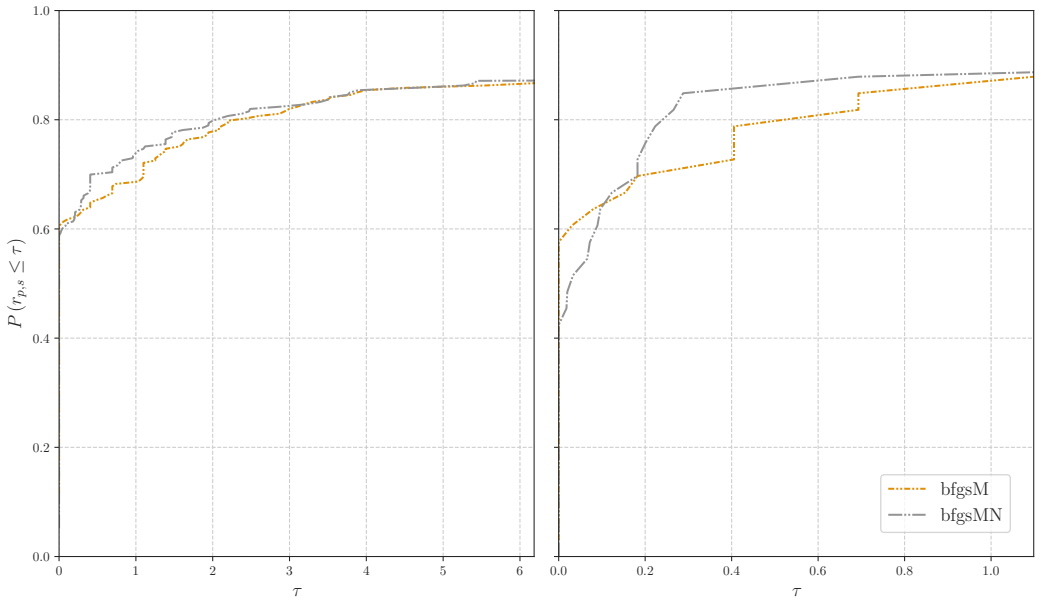

Figure 8: Performance profiles comparing CPU time for `bfgsM` and `bfgsMN` on the full problem set (left) and the common problem set (right).

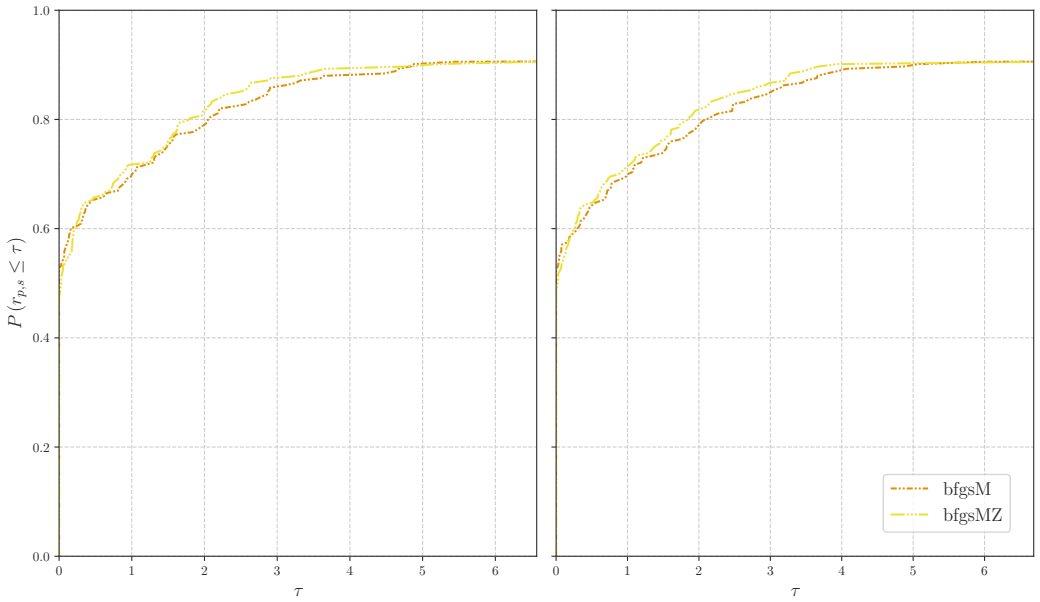

Figure 9: Performance profiles comparing function evaluations (left) and iterations (right) for `bfgsM` and `bfgsMZ`.

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 9).

7. Conclusion

In this thesis 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 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. From page 201 of Nocedal and Wright [13] (a standard graduate and undergraduate text on optimization):

> *"However, computational experience suggests no real advantages for this variant, and we prefer the simpler strategy of Algorithm 8.1."*

(In Algorithm 8.1 Nocedal and Wright update the inverse.)

Variations of quasi-Newton methods that focus on managing the condition number and preserving positive definiteness result in significant 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. This reminds us that stability must be taken into account in numerical optimization. A method might have wonderful theoretical properties, but if it fails to deal with numerical issues these properties simply cannot be realized.

7.1. Acknowledgments

I would like to thank my advisor Professor Philip E. Gill for all his help, patience, and guidance. Not only is he a brilliant mathematician, but he also has a tremendous capacity for sharing his experience and for developing students' skills. I have learned an incredible amount and I am fortunate to have had the opportunity to work with him. Thank you Professor Gill! Finally, I would like to acknowledge partial support from the National Science Foundation Research Training Grant (RTG) DMS-1345013.

References

[1] Neculai Andrei. An adaptive scaled BFGS method for unconstrained optimization. *Numerical Algorithms*, 77(2):413–432, 2018. 27, 38

[2] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966. 4

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

[4] 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. 7

[5] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2, Ser. A):201–213, 2002. 33

[6] Roger Fletcher and Michael J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168, 1963. 5

[7] Philip E. Gill and Walter Murray. Quasi-Newton methods for unconstrained optimization. *J. Inst. Maths Applics*, 9:91–108, 1972. 7

[8] 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. 4

[9] Philip E. Gill and Margaret H. Wright. *Computational Optimization: Nonlinear Programming*. Cambridge University Press, New York, NY, USA, 2020. To be published in 2020. 6

[10] 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

[11] Jorge J. Moré and Danny C. Sorensen. Newton's method. In G. H. Golub, editor, *Studies in Mathematics, Volume 24. MAA Studies in Numerical Analysis*, pages 29–82. Math. Assoc. America, Washington, DC, 1984. 5

[12] Jorge J. Moré and David J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Software*, 20(3):286–307, 1994. 4, 5

[13] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999. 5, 40

[14] Shmuel S. Oren. Self-scaling variable metric algorithms without line search for unconstrained optimization. *Math. Comp.*, 27(124):873–885, October 1973. 21

[15] Shmuel S. Oren. On the selection of parameters in self-scaling variable metric algorithms. *Math. Program.*, 7:351–367, 1974. 21

[16] Shmuel S. Oren. Self-scaling variable metric (SSVM) algorithms, Part ii: implementation and experiments. *Management Science*, 20:863–874, 1974. 21, 24

[17] 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. 21

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

[19] 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. 4

[20] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Rev.*, 11:226–235, 1969. 4

[21] Philip Wolfe. On the convergence of gradient methods under constraint. *IBM J. Res. Dev.*, 16:407–411, 1972. 5

[22] Ya-Xiang Yuan. A modified BFGS algorithm for unconstrained optimization. *IMA J. Numer. Anal.*, 11(3):325–332, 1991. 26

[23] 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. 30, 32