SIAM J. OPTIM.
Vol. 8, No. 4, pp. 1060–1083, November 1998
© 1998 Society for Industrial and Applied Mathematics
012

BFGS WITH UPDATE SKIPPING AND VARYING MEMORY*

TAMARA G. KOLDA†, DIANNE P. O'LEARY‡, AND LARRY NAZARETH§

Abstract. We give conditions under which limited-memory quasi-Newton methods with exact line searches will terminate in n steps when minimizing n-dimensional quadratic functions. We show that although all Broyden family methods terminate in n steps in their full-memory versions, only BFGS does so with limited-memory. Additionally, we show that full-memory Broyden family methods with exact line searches terminate in at most $n + p$ steps when p matrix updates are skipped. We introduce new limited-memory BFGS variants and test them on nonquadratic minimization problems.

Key words. minimization, quasi-Newton, BFGS, limited-memory, update skipping, Broyden family

AMS subject classifications. 65K10, 65H10

PII. S1052623496306450

1. Introduction.

The quasi-Newton family of algorithms remains a standard workhorse for minimization. Many of these methods share the properties of finite termination on strictly convex quadratic functions, a linear or superlinear rate of convergence on general convex functions, and no need to store or evaluate the second derivative matrix. In general, an approximation to the second derivative matrix is built by accumulating the results of earlier steps. Descriptions of many quasi-Newton algorithms can be found in books by Luenberger [17] and Dennis and Schnabel [8].

Although there are an infinite number of quasi-Newton methods, one method surpasses the others in popularity: the BFGS algorithm of Broyden, Fletcher, Goldfarb, and Shanno; see, e.g., Dennis and Schnabel [8]. This method exhibits more robust behavior than its relatives. Many attempts have been made to explain this robustness, but a complete understanding has yet to be obtained [24]. One result of the work in this paper is a small step toward this understanding, since we investigate the question of how much and which information can be dropped in BFGS and other quasi-Newton methods without destroying the property of quadratic termination.

We answer this question in the context of exact line search methods, those that find a minimizer on a one-dimensional subspace at every iteration. (In practice, inexact line searches that satisfy side conditions such as those proposed by Wolfe (see section 4.3) are substituted for exact line searches.) We focus on modifications of well-known quasi-Newton algorithms resulting from limiting the memory, either by discarding the results of early steps (section 2) or by skipping some updates to the second derivative approximation (section 3). We give conditions under which quasi-Newton methods will terminate in n steps when minimizing quadratic functions of n variables. Although all Broyden family methods (see section 2) terminate in n steps

*Received by the editors July 10, 1996; accepted for publication (in revised form) October 3, 1997; published electronically September 23, 1998.
http://www.siam.org/journals/siopt/8-4/30645.html

†Computational Methods, Building 6012, P.O. Box 2008, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6367 (kolda@msr.epm.ornl.gov). This work was supported in part by the National Physical Science Consortium, the National Security Agency, and the University of Maryland.

‡Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu). This work was supported by the National Science Foundation under grant NSF CCR-95-03126.

§Department of Pure and Applied Mathematics, Washington State University, Pullman, WA 99164 (nazareth@amath.washington.edu).

in their full-memory versions, we show that only BFGS has n-step termination under limited-memory. We also show that the methods from the Broyden family terminate in $n + p$ steps even if p updates are skipped, but termination is lost if we both skip updates and limit the memory.

In section 4, we report the results of experiments with new limited-memory BFGS(L-BFGS) variants on problems taken from the constrained and unconstrained testing environment (CUTE) [3] test set, showing that some savings in time can be achieved.

Notation. Matrices and vectors are denoted by boldface uppercase and lowercase letters, respectively. Scalars are denoted by Greek or Roman letters. The superscript "T" denotes transposition. Subscripts denote iteration number. Products are always taken from left to right:

$$\prod_{i=j}^{k} \mathbf{B}_j = \begin{cases} \mathbf{B}_j \cdot \mathbf{B}_{j+1} \cdots \mathbf{B}_k & \text{if } j \le k, \\ \mathbf{I} & \text{otherwise.} \end{cases}$$

The notation $\text{span}\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ denotes the subspace spanned by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. Whenever we refer to an n-dimensional strictly convex quadratic function, we assume it is of the form

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b},$$

where $\mathbf{A}$ is a positive definite $n \times n$ matrix and $\mathbf{b}$ is an n-vector.

2. Limited-memory variations of quasi-Newton algorithms. In this section we characterize full-memory and limited-memory methods that terminate in n iterations on n-dimensional strictly convex quadratic minimization problems using exact line searches. Most full-memory versions of the methods we will discuss are known to terminate in n iterations. Limited-memory methods store the quasi-Newton matrix implicitly and require less memory; furthermore, the computation of the search direction is often less expensive since it involves the implicitly stored matrix. L-BFGS was shown by Nocedal [23] to terminate in n steps. The preconditioned conjugate gradient method, which can be cast as a limited-memory quasi-Newton method, is also known to terminate in n iterations; see, e.g., Luenberger [17] or Golub and Van Loan [12]. Little else is known about termination of limited-memory methods.

Let $f(\mathbf{x})$ denote the strictly convex quadratic function to be minimized, and let $\mathbf{g}(\mathbf{x})$ denote the gradient of f. We define $\mathbf{g}_k \equiv \mathbf{g}(\mathbf{x}_k)$, where $\mathbf{x}_k$ is the kth iterate and denote the change in iterate and gradient by

$$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k,$$
$$\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k.$$

We present a general result that characterizes quasi-Newton methods (see Figure 2.1) that terminate in n iterations. We restrict ourselves to methods with an update of the form

$$\mathbf{H}_{k+1} = \gamma_k \mathbf{P}_k^T \mathbf{H}_0 \mathbf{Q}_k + \sum_{i=1}^{m_k} \mathbf{w}_{ik}\mathbf{z}_{ik}^T. \tag{2.1}$$

Here,

1. $\mathbf{H}_0$ is an $n \times n$ symmetric positive definite matrix that remains constant for all k, and γ_k is a nonzero scalar that can be thought of as an iterative rescaling of $\mathbf{H}_0$;

Let $\mathbf{x}_0$ be the starting point, and let $\mathbf{H}_0$ be the initial inverse Hessian approximation. For $k = 0, 1, \ldots$

1. Compute $\mathbf{d}_k = -\mathbf{H}_k \mathbf{g}_k$.
2. Choose $\alpha_k > 0$ such that $f(\mathbf{x}_k + \alpha \mathbf{d}_k) \geq f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)$ for all $\alpha > 0$.
3. Set $\mathbf{s}_k = \alpha_k \mathbf{d}_k$.
4. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$.
5. Compute $\mathbf{g}_{k+1}$.
6. Set $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$.
7. Choose $\mathbf{H}_{k+1}$.

FIG. 2.1. *General quasi-Newton method.*

2. $\mathbf{P}_k$ is an $n \times n$ matrix that is the product of projection matrices of the form

$$\mathbf{I} - \frac{\mathbf{u}\mathbf{v}^T}{\mathbf{u}^T\mathbf{v}}, \tag{2.2}$$

where $\mathbf{u} \in \text{span}\{\mathbf{y}_0, \ldots, \mathbf{y}_k\}$ and $\mathbf{v} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{k+1}\}$[1], and $\mathbf{Q}_k$ is an $n \times n$ matrix that is the product of projection matrices of the same form where $\mathbf{u}$ is any n-vector and $\mathbf{v} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\}$;

3. m_k is a nonnegative integer, $\mathbf{w}_{ik}$ $(i = 1, 2, \ldots, m_k)$ is any n-vector, and $\mathbf{z}_{ik}$ $(i = 1, 2, \ldots, m_k)$ is any vector in $\text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\}$.

We refer to this form as the *general form.* The general form fits many known quasi-Newton methods, including the Broyden family and the L-BFGS method. We do not assume that these quasi-Newton methods satisfy the secant condition

$$\mathbf{H}_{k+1}\mathbf{y}_k = \mathbf{s}_k,$$

nor that $\mathbf{H}_{k+1}$ is positive definite and symmetric. Symmetric positive definite updates are desirable since this guarantees that the quasi-Newton method produces descent directions. Note that if the update is not positive definite, we may produce a $\mathbf{d}_k$ such that $\mathbf{d}_k^T \mathbf{g}_k > 0$, in which case we choose α_k over all *negative* α rather than all positive α.

Example 1. The method of steepest descent [17] fits the general form (2.1). For each k we define

$$\gamma_k = 1, \quad m_k = 0, \text{ and } \mathbf{P}_k = \mathbf{Q}_k = \mathbf{H}_0 = \mathbf{I}. \tag{2.3}$$

Note that neither $\mathbf{w}$ nor $\mathbf{z}$ vectors are specified since $m_k = 0$.

Example 2. The $(k+1)$st update for the conjugate gradient method with preconditioner $\mathbf{H}_0$ fits the general form (2.1) with

$$\gamma_k = 1, \quad m_k = 0, \quad \mathbf{P}_k = \mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{s}_k^T \mathbf{y}_k}, \text{ and } \mathbf{Q}_k = \mathbf{I}. \tag{2.4}$$

Example 3. A full-memory quasi-Newton method can be converted into a limited-memory method in the following way. Define $\mathbf{H}_{k+1}$ to be the result of applying the

[1] The vector s_{k+1} has not yet been explicitly calculated but is needed here only for the theoretical framework, not for the computational algorithms. In fact, it may also be available computationally in algorithms such as the limited memory DFP; see the proof of Proposition 2.1.

update formula to $\mathbf{H}_0$ m times using the m most recent $(\mathbf{s}, \mathbf{y})$ pairs. The L-BFGS update (see Nocedal [23]) with limited-memory constant m can be written as

$$\mathbf{H}_{k+1} = \mathbf{V}_{k-m_k+1,k}^T \mathbf{H}_0 \mathbf{V}_{k-m_k+1,k} + \sum_{i=k-m_k+1}^{k} \mathbf{V}_{i+1,k}^T \frac{\mathbf{s}_i \mathbf{s}_i^T}{\mathbf{s}_i^T \mathbf{y}_i} \mathbf{V}_{i+1,k}, \tag{2.5}$$

where $m_k = \min\{k+1, m\}$ and

$$\mathbf{V}_{ik} = \prod_{j=i}^{k} \left(\mathbf{I} - \frac{\mathbf{y}_i \mathbf{s}_i^T}{\mathbf{s}_i^T \mathbf{y}_i} \right).$$

L-BFGS fits the general form (2.1) if at iteration k we choose

$$\begin{aligned} &\gamma_k = 1, \quad m_k = \min\{k+1, m\}, \\ &\mathbf{P}_k = \mathbf{Q}_k = \mathbf{V}_{k-m_k+1,k}, \text{ and} \\ &\mathbf{w}_{ik} = \mathbf{z}_{ik} = \frac{(\mathbf{V}_{k-m_k+i+1,k})^T (\mathbf{s}_{k-m_k+i})}{\sqrt{(\mathbf{s}_{k-m_k+i})^T (\mathbf{y}_{k-m_k+i})}}. \end{aligned} \tag{2.6}$$

Observe that $\mathbf{P}_k, \mathbf{Q}_k$, and $\mathbf{z}_{ik}$ all obey the constraints imposed on their construction.

Example 4. We define limited-memory DFP (L-DFP) in a similar way: $\mathbf{H}_{k+1} = \hat{\mathbf{H}}_{k+1}^{(m_k)}$, where, for $i = 0, \ldots, m_k$,

$$\hat{\mathbf{H}}_{k+1}^{(0)} = \mathbf{H}_0$$

and

$$\hat{\mathbf{H}}_{k+1}^{(i)} = \hat{\mathbf{H}}_{k+1}^{(i-1)} + \mathbf{U}_{\text{DFP}}(\hat{\mathbf{H}}_{k+1}^{(i-1)}, \mathbf{s}_{k-m_k+i}, \mathbf{y}_{k-m_k+i}),$$

with

$$\mathbf{U}_{\text{DFP}}(\mathbf{H}, \mathbf{s}, \mathbf{y}) = -\frac{\mathbf{H}\mathbf{y}\mathbf{y}^T\mathbf{H}}{\mathbf{y}^T\mathbf{H}\mathbf{y}} + \frac{\mathbf{s}\mathbf{s}^T}{\mathbf{s}^T\mathbf{y}}.$$

To simplify our description, note that $\hat{\mathbf{H}}_{k+1}^{(i)}$ can be rewritten as

$$\begin{aligned} \hat{\mathbf{H}}_{k+1}^{(i)} &= \left(\mathbf{I} - \frac{\hat{\mathbf{H}}_{k+1}^{(i-1)} \mathbf{y}_{k-m_k+i} \mathbf{y}_{k-m_k+i}^T}{\mathbf{y}_{k-m_k+i}^T \hat{\mathbf{H}}_{k+1}^{(i-1)} \mathbf{y}_{k-m_k+i}} \right) \hat{\mathbf{H}}_{k+1}^{(i-1)} + \frac{\mathbf{s}_{k-m_k+i} \mathbf{s}_{k-m_k+i}^T}{\mathbf{s}_{k-m_k+i}^T \mathbf{y}_{k-m_k+i}} \\ &= \left(\hat{\mathbf{V}}_{0k}^{(i)} \right)^T \mathbf{H}_0 + \sum_{j=1}^{i} \left(\hat{\mathbf{V}}_{jk}^{(i)} \right)^T \frac{\mathbf{s}_{k-m_k+j} \mathbf{s}_{k-m_k+j}^T}{\mathbf{s}_{k-m_k+j}^T \mathbf{y}_{k-m_k+j}}, \end{aligned}$$

for $i \geq 1$, where

$$\hat{\mathbf{V}}_{jk}^{(i)} = \prod_{l=j+1}^{i} \left[\mathbf{I} - \frac{\mathbf{y}_{k-m_k+l} \left(\mathbf{H}_{k+1}^{(l-1)} \mathbf{y}_{k-m_k+l} \right)^T}{\mathbf{y}_{k-m_k+l}^T \mathbf{H}_{k+1}^{(l-1)} \mathbf{y}_{k-m_k+l}} \right].$$

Thus $\mathbf{H}_{k+1}$ can be written as

$$\mathbf{H}_{k+1} = \mathbf{V}_{0k}^T \mathbf{H}_0 + \sum_{i=1}^{m_k} \left(\mathbf{V}_{ik}^T \frac{\mathbf{s}_{k-m_k+i} \mathbf{s}_{k-m_k+i}^T}{\mathbf{s}_{k-m_k+i}^T \mathbf{y}_{k-m_k+i}} \right), \tag{2.7}$$

where

$$\mathbf{V}_{ik} = \prod_{j=i+1}^{m_k} \left[\mathbf{I} - \frac{\mathbf{y}_{k-m_k+j} \left(\hat{\mathbf{H}}_{k+1}^{(j-1)} \mathbf{y}_{k-m_k+j} \right)^T}{\mathbf{y}_{k-m_k+j}^T \hat{\mathbf{H}}_{k+1}^{(j-1)} \mathbf{y}_{k-m_k+j}} \right].$$

Equation (2.7) looks very much like the general form given in (2.1). L-DFP fits the general form with the following choices:

$$\begin{aligned} &\gamma_k = 1, \quad \mathbf{P}_k = \mathbf{V}_{0k}, \quad \mathbf{Q}_k = \mathbf{I}, \\ &\mathbf{w}_{ik} = \mathbf{V}_{ik}^T \mathbf{s}_{k-m_k+i} / (\mathbf{s}_{k-m_k+i}^T \mathbf{y}_{k-m_k+i}), \text{ and } \mathbf{z}_{ik} = \mathbf{s}_{k-m_k+i}. \end{aligned} \tag{2.8}$$

Except for the choice of $\mathbf{P}_k$, it is trivial to verify that the choices satisfy the general form. To prove that $\mathbf{P}_k$ satisfies the requirements, we need to show

$$\hat{\mathbf{H}}_{k+1}^{(i-1)} \mathbf{y}_{k-m_k+i} \in \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{k+1}\}, \text{ for } i = 1, \ldots, m_k \text{ and all } k. \tag{2.9}$$

PROPOSITION 2.1. *For L-DFP, the following three conditions hold for each value of k:*

$$\hat{\mathbf{H}}_{k+1}^{(i-1)} \mathbf{y}_{k-m_k+i} \in \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\} \text{ for } i = 1, \ldots, m_k - 1, \tag{2.10}$$

$$\hat{\mathbf{H}}_{k+1}^{(i-1)} \mathbf{y}_{k-m_k+i} \in \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k, \mathbf{H}_0 \mathbf{g}_{k+1}\} \text{ for } i = m_k, \text{ and} \tag{2.11}$$

$$\operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_{k+1}\} \subseteq \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{k+1}\}. \tag{2.12}$$

Proof. We will prove this via induction. Suppose $k = 0$. Then $m_0 = 1$. We have

$$\hat{\mathbf{H}}_{k+1}^{(0)} \mathbf{y}_k = \mathbf{H}_0 \mathbf{y}_0 = \mathbf{H}_0 \mathbf{g}_1 - \mathbf{H}_0 \mathbf{g}_0 \in \operatorname{span}\{\mathbf{s}_0, \mathbf{H}_0 \mathbf{g}_1\}.$$

(Recall that $\operatorname{span}\{\mathbf{s}_0\}$ is trivially equal to $\operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0\}$.) Furthermore,

$$\begin{aligned} \mathbf{s}_1 &= -\alpha_1 \mathbf{H}_1 \mathbf{g}_1 \\ &= -\alpha_1 \left[\mathbf{H}_0 \mathbf{g}_1 - \frac{\mathbf{y}_0^T \mathbf{H}_0 \mathbf{g}_1}{\mathbf{y}_0^T \mathbf{H}_0 \mathbf{y}_0} (\mathbf{H}_0 \mathbf{g}_1 - \mathbf{H}_0 \mathbf{g}_0) + \frac{\mathbf{s}_0^T \mathbf{g}_1}{\mathbf{y}_0^T \mathbf{s}_0} \mathbf{s}_0 \right]. \end{aligned}$$

So we can conclude

$$\left(1 - \frac{\mathbf{y}_0^T \mathbf{H}_0 \mathbf{g}_1}{\mathbf{y}_0^T \mathbf{H}_0 \mathbf{y}_0} \right) \mathbf{H}_0 \mathbf{g}_1 = - \left[\frac{1}{\alpha_1} \mathbf{s}_1 + \frac{\mathbf{y}_0^T \mathbf{H}_0 \mathbf{g}_1}{\mathbf{y}_0^T \mathbf{H}_0 \mathbf{y}_0} \mathbf{H}_0 \mathbf{g}_0 + \frac{\mathbf{s}_0^T \mathbf{g}_1}{\mathbf{y}_0^T \mathbf{s}_0} \mathbf{s}_0 \right].$$

Hence, $\mathbf{H}_0 \mathbf{g}_1 \in \operatorname{span}\{\mathbf{s}_0, \mathbf{s}_1\}$, and so the base case holds.

Assume that

$$\begin{aligned} &\hat{\mathbf{H}}_k^{(i-1)} \mathbf{y}_{k-1-m_{k-1}+i} \in \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{k-1}\} \text{ for } i = 1, \ldots, m_{k-1} - 1, \\ &\hat{\mathbf{H}}_k^{(i-1)} \mathbf{y}_{k-1-m_{k-1}+i} \in \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{k-1}, \mathbf{H}_0 \mathbf{g}_k\} \text{ for } i = m_{k-1}, \text{ and} \\ &\operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_k\} \subseteq \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\}. \end{aligned}$$

Using the induction assumption, we will show that (2.10)–(2.12) holds for $(k+1)$. We show (2.10) for $i = 1, \ldots, m_k - 1$. For $i = 1$ (assume $m_k > i$),

$$\hat{\mathbf{H}}_{k+1}^{(0)} \mathbf{y}_{k-m_k+1} = \mathbf{H}_0 \mathbf{y}_{k-m_k+1} = \mathbf{H}_0 \mathbf{g}_{k-m_k+2} - \mathbf{H}_0 \mathbf{g}_{k-m_k+1}.$$

Using the induction hypothesis, we get that

$$\hat{\mathbf{H}}_{k+1}^{(0)}\mathbf{y}_{k-m_k+1} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\}.$$

Assume that

$$\hat{\mathbf{H}}_{k+1}^{(j)}\mathbf{y}_{k-m_k+j+1} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\} \tag{2.13}$$

for j between 1 and $i-2$, $i \leq m_k - 2$. Then

$$\hat{\mathbf{H}}_{k+1}^{(i-1)}\mathbf{y}_{k-m_k+i} = \left(\hat{\mathbf{V}}_{0k}^{(i-1)}\right)^T \mathbf{H}_0\mathbf{y}_{k-m_k+i} + \sum_{j=1}^{i-1} \frac{\mathbf{s}_{k-m_k+j-1}^T\mathbf{y}_{k-m_k+i}}{\mathbf{s}_{k-m_k+j-1}^T\mathbf{y}_{k-m_k+j-1}} \left(\hat{\mathbf{V}}_{jk}^{(i-1)}\right)^T \mathbf{s}_{k-m_k+j-1}.$$

For values of $i \leq m_k - 1$, $\left(\hat{\mathbf{V}}_{jk}^{(i-1)}\right)^T$ maps any vector $\mathbf{v}$ into

$$\text{span}\{\mathbf{v}, \hat{\mathbf{H}}_{k+1}^{(0)}\mathbf{y}_{k-m_k+1}, \ldots, \hat{\mathbf{H}}_{k+1}^{(i-2)}\mathbf{y}_{k-2}\},$$

and so $\hat{\mathbf{H}}_{k+1}^{(i-1)}\mathbf{y}_{k-m_k+i}$ is in

$$\text{span}\{\mathbf{H}_0\mathbf{y}_{k-m_k+i}, \hat{\mathbf{H}}_{k+1}^{(0)}\mathbf{y}_{k-m_k+1}, \ldots, \hat{\mathbf{H}}_{k+1}^{(i-2)}\mathbf{y}_{k-2}, \mathbf{s}_{k-m_k+1}, \ldots, \mathbf{s}_{k-2}\}.$$

Using the induction hypothesis and (2.13), we get

$$\hat{\mathbf{H}}_{k+1}^{(i-1)}\mathbf{y}_{k-m_k+i} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\},$$

and we can conclude that (2.10) is true for $i = 1, \ldots, m_k - 1$ in the $(k+1)$st case. If $i = m_k$, then

$$\hat{\mathbf{H}}_{k+1}^{(m_k-1)}\mathbf{y}_k \in \text{span}\{\mathbf{H}_0\mathbf{y}_k, \hat{\mathbf{H}}_{k+1}^{(0)}\mathbf{y}_{k-m_k+1}, \ldots, \hat{\mathbf{H}}_{k+1}^{(m_k-2)}\mathbf{y}_{k-1}, \mathbf{s}_{k-m_k+1}, \ldots, \mathbf{s}_{k-1}\},$$

so

$$\hat{\mathbf{H}}_{k+1}^{(m_k-1)}\mathbf{y}_k \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k, \mathbf{H}_0\mathbf{g}_{k+1}\}.$$

Hence (2.11) is true for $(k+1)$.

Now, consider

$$\begin{aligned}\mathbf{s}_{k+1} &= -\alpha_{k+1}\mathbf{H}_{k+1}\mathbf{g}_{k+1} \\ &= \mathbf{V}_{0k}^T\mathbf{H}_0\mathbf{g}_{k+1} + \sum_{i=1}^{m_k} \frac{\mathbf{s}_{k-m_k+i}^T\mathbf{g}_{k+1}}{\mathbf{s}_{k-m_k+i}^T\mathbf{y}_{k-m_k+i}}\mathbf{V}_{ik}^T\mathbf{s}_{k-m_k+i}.\end{aligned}$$

Using the structure of $\mathbf{V}_{jk}$ and (2.10) we see that $\mathbf{H}_0\mathbf{g}_{k+1} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{k+1}\}$. Hence, (2.12) also holds for $(k+1)$. □

Example 5. The Broyden family is the class of quasi-Newton methods whose matrices are linear combinations of the DFP and BFGS matrices:

$$\mathbf{H}_{k+1} = \phi\,\mathbf{H}_{k+1}^{BFGS} + (1-\phi)\,\mathbf{H}_{k+1}^{DFP}, \qquad \phi \in \mathbf{R};$$

see, e.g., Luenberger [17, Chap. 9]. The parameter ϕ is usually restricted to values that are guaranteed to produce a positive definite update, although recent work with SR1, a Broyden family method, by Khalfan, Byrd, and Schnabel [15] may change this practice. No restriction on ϕ is necessary for the development of our theory. The Broyden family update can be expressed as

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_k} - \frac{\mathbf{H}_k\mathbf{y}_k\mathbf{y}_k^T\mathbf{H}_k}{\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k} + \phi\,(\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k)\left(\frac{\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{y}_k} - \frac{\mathbf{H}_k\mathbf{y}_k}{\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k}\right)\left(\frac{\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{y}_k} - \frac{\mathbf{H}_k\mathbf{y}_k}{\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k}\right)^T.$$

We sketch the explanation of how the full-memory version fits the general form given in (2.1). The limited-memory case is similar. We can rewrite the Broyden family update as follows:

$$\begin{aligned}\mathbf{H}_{k+1} &= \mathbf{H}_k + (\phi-1)\frac{\mathbf{H}_k\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k}\mathbf{H}_k - \phi\,\frac{\mathbf{s}_k\mathbf{y}_k^T}{\mathbf{s}_k^T\mathbf{y}_k}\mathbf{H}_k + \frac{\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_k} \\ &\quad + \phi\,\frac{\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k\cdot\mathbf{s}_k\mathbf{s}_k^T}{(\mathbf{s}_k^T\mathbf{y}_k)^2} - \phi\,\frac{\mathbf{H}_k\mathbf{y}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_k} \\ &= \left[\mathbf{I} - \frac{\left((1-\phi)\mathbf{s}_k^T\mathbf{y}_k\cdot\mathbf{H}_k\mathbf{y}_k + \phi\,\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k\cdot\mathbf{s}_k\right)\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k\cdot\mathbf{s}_k^T\mathbf{y}_k}\right]\mathbf{H}_k \\ &\quad + \left[\left(1+\phi\,\frac{\mathbf{y}_k^T\mathbf{H}_k\mathbf{y}_k}{\mathbf{s}_k^T\mathbf{y}_k}\right)\mathbf{s}_k - \phi\,\mathbf{H}_k\mathbf{y}_k\right]\frac{\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_k}.\end{aligned}$$

Hence,

$$\mathbf{H}_{k+1} = \mathbf{V}_{0k}\mathbf{H}_0 + \sum_{i=1}^{k+1}\mathbf{w}_{ik}\mathbf{z}_{ik}^T,$$

where

$$\mathbf{V}_{ik} = \prod_{j=i}^{k}\left[\mathbf{I} - \frac{\left((1-\phi)\mathbf{s}_j^T\mathbf{y}_j\cdot\mathbf{H}_j\mathbf{y}_j + \phi\,\mathbf{y}_j^T\mathbf{H}_j\mathbf{y}_j\cdot\mathbf{s}_j\right)\mathbf{y}_j^T}{\mathbf{y}_j^T\mathbf{H}_j\mathbf{y}_j\cdot\mathbf{s}_j^T\mathbf{y}_j}\right],$$

$$\mathbf{w}_{ik} = \mathbf{V}_{ik}\left[\left(1+\phi\,\frac{\mathbf{y}_{i-1}^T\mathbf{H}_{i-1}\mathbf{y}_{i-1}}{\mathbf{s}_{i-1}^T\mathbf{y}_{i-1}}\mathbf{s}_{i-1}\right) - \phi\,\mathbf{H}_{i-1}\mathbf{y}_{i-1}\right], \text{ and } \mathbf{z}_{ik} = \frac{\mathbf{s}_{i-1}^T}{\mathbf{s}_{i-1}^T\mathbf{y}_{i-1}}.$$

It is left to the reader to show that $\mathbf{H}_k\mathbf{y}_k$ is in $\text{span}\{\mathbf{s}_0,\ldots,\mathbf{s}_{k+1}\}$, and thus the Broyden family updates fit the form in (2.1).

2.1. Termination of limited-memory methods. In this section we show that methods fitting the general form (2.1) produce conjugate search directions (see Theorem 2.2) and terminate in n iterations (see Corollary 2.3) *if and only if* $\mathbf{P}_k$ maps the vectors $\mathbf{y}_0$ through $\mathbf{y}_k$ into $\text{span}\{\mathbf{y}_0,\ldots,\mathbf{y}_{k-1}\}$ for each $k = 1, 2, \ldots, n$. Furthermore, this condition on $\mathbf{P}_k$ is satisfied only if $\mathbf{y}_k$ is used in its formation (see Corollary 2.4).

THEOREM 2.2. *Suppose that we apply a quasi-Newton method* (*Figure* 2.1) *with an update of the form* (2.1) *to minimize an* n*-dimensional strictly convex quadratic function. Then for each* k *before termination* (*i.e.*, $\mathbf{g}_{k+1} \neq 0$),

$$\mathbf{g}_{k+1}^T\mathbf{s}_j = 0, \text{ for all } j = 0, 1, \ldots, k, \tag{2.14}$$

$$\mathbf{s}_{k+1}^T\mathbf{A}\mathbf{s}_j = 0, \text{ for all } j = 0, 1, \ldots, k, \text{ and} \tag{2.15}$$

$$\text{span}\{\mathbf{s}_0,\ldots,\mathbf{s}_{k+1}\} = \text{span}\{\mathbf{H}_0\mathbf{g}_0,\ldots,\mathbf{H}_0\mathbf{g}_{k+1}\}, \tag{2.16}$$

if and only if

$$\mathbf{P}_j \mathbf{y}_i \in \text{span}\{\mathbf{y}_0, \ldots, \mathbf{y}_{j-1}\} \text{ for all } i = 0, 1, \ldots, j, \quad j = 0, 1, \ldots, k. \tag{2.17}$$

Proof. ($\Leftarrow$) Assume that (2.17) holds. We will prove (2.14)–(2.16) by induction. Since the line searches are exact, $\mathbf{g}_1$ is orthogonal to $\mathbf{s}_0$. Using the fact that $\mathbf{P}_0 \mathbf{y}_0 = 0$ from (2.17) and the fact that $\mathbf{z}_{i0} \in \text{span}\{\mathbf{s}_0\}$ implies $\mathbf{g}_1^T \mathbf{z}_{i0} = 0$, $i = 1, \ldots, m_k$, we see that $\mathbf{s}_1$ is conjugate to $\mathbf{s}_0$ since

$$\begin{aligned}
\mathbf{s}_1^T \mathbf{A} \mathbf{s}_0 &= \alpha_1 \mathbf{d}_1^T \mathbf{y}_0 \\
&= -\alpha_1 \mathbf{g}_1^T \mathbf{H}_1^T \mathbf{y}_0 \\
&= -\alpha_1 \mathbf{g}_1^T \left(\gamma_0 \mathbf{Q}_0^T \mathbf{H}_0 \mathbf{P}_0 + \sum_{i=0}^{m_0} \mathbf{z}_{i0} \mathbf{w}_{i0}^T \right) \mathbf{y}_0 \\
&= 0.
\end{aligned}$$

Finally, $\text{span}\{\mathbf{s}_0\} = \text{span}\{\mathbf{H}_0 \mathbf{g}_0\}$, and so the base case is established.

We will assume that claims (2.14)–(2.16) hold for $k = 0, 1, \ldots, \hat{k} - 1$ and prove that they also hold for $k = \hat{k}$.

The vector $\mathbf{g}_{\hat{k}+1}$ is orthogonal to $\mathbf{s}_{\hat{k}}$ since the line search is exact. Using the induction hypotheses that $\mathbf{g}_{\hat{k}}$ is orthogonal to $\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}-1}\}$ and $\mathbf{s}_{\hat{k}}$ is conjugate to $\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}-1}\}$, we see that, for $j < \hat{k}$,

$$\mathbf{g}_{\hat{k}+1}^T \mathbf{s}_j = (\mathbf{g}_{\hat{k}} + \mathbf{y}_{\hat{k}})^T \mathbf{s}_j = (\mathbf{g}_{\hat{k}} + \mathbf{A}\mathbf{s}_{\hat{k}})^T \mathbf{s}_j = 0.$$

Hence, (2.14) holds for $k = \hat{k}$.

To prove (2.15), we note that

$$\mathbf{s}_{\hat{k}+1}^T \mathbf{A} \mathbf{s}_j = -\alpha_{\hat{k}+1} \mathbf{g}_{\hat{k}+1}^T \mathbf{H}_{\hat{k}+1}^T \mathbf{y}_j,$$

so it is sufficient to prove that $\mathbf{g}_{\hat{k}+1}^T \mathbf{H}_{\hat{k}+1}^T \mathbf{y}_j = 0$ for $j = 0, 1, \ldots, \hat{k}$. We will use the following facts:

(i) $\mathbf{g}_{\hat{k}+1}^T \mathbf{Q}_{\hat{k}}^T = \mathbf{g}_{\hat{k}+1}^T$ since the $\mathbf{v}$ in each of the projections used to form $\mathbf{Q}_{\hat{k}}$ is in $\text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}}\}$, and $\mathbf{g}_{\hat{k}+1}$ is orthogonal to that span.

(ii) $\mathbf{g}_{\hat{k}+1}^T \mathbf{z}_{i\hat{k}} = 0$ for $i = 1, \ldots, m_{\hat{k}}$ since each $\mathbf{z}_{i\hat{k}}$ is in $\text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}}\}$, and $\mathbf{g}_{\hat{k}+1}$ is orthogonal to that span.

(iii) Since we are assuming that (2.17) holds true, for each $j = 0, 1, \ldots, \hat{k}$ there exist $\mu_0, \ldots, \mu_{\hat{k}-1}$ such that $\mathbf{P}_{\hat{k}} \mathbf{y}_j$ can be expressed as $\sum_{i=0}^{\hat{k}-1} \mu_i \mathbf{y}_i$.

(iv) For $i = 0, 1, \ldots, \hat{k}-1$, $\mathbf{g}_{\hat{k}+1}$ is orthogonal to $\mathbf{H}_0 \mathbf{y}_i$ because $\mathbf{g}_{\hat{k}+1}$ is orthogonal to $\text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}}\}$ and $\mathbf{H}_0 \mathbf{y}_i \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}}\}$ from (2.16).

Thus,

$$\begin{aligned}
\mathbf{g}_{\hat{k}+1}^T \mathbf{H}_{\hat{k}+1}^T \mathbf{y}_j &= \mathbf{g}_{\hat{k}+1}^T \left(\gamma_{\hat{k}} \mathbf{Q}_{\hat{k}}^T \mathbf{H}_0 \mathbf{P}_{\hat{k}} + \sum_{i=1}^{m_{\hat{k}}} \mathbf{z}_{i\hat{k}} \mathbf{w}_{i\hat{k}}^T \right) \mathbf{y}_j \\
&= \gamma_{\hat{k}} \mathbf{g}_{\hat{k}+1}^T \mathbf{Q}_{\hat{k}}^T \mathbf{H}_0 \mathbf{P}_{\hat{k}} \mathbf{y}_j + \sum_{i=1}^{m_{\hat{k}}} \mathbf{g}_{\hat{k}+1}^T \mathbf{z}_{i\hat{k}} \mathbf{w}_{i\hat{k}}^T \mathbf{y}_j \\
&= \gamma_{\hat{k}} \mathbf{g}_{\hat{k}+1}^T \mathbf{H}_0 \mathbf{P}_{\hat{k}} \mathbf{y}_j
\end{aligned}$$

$$
\begin{aligned}
&= \gamma_{\hat{k}} \mathbf{g}_{\hat{k}+1}^T \mathbf{H}_0 \left(\sum_{i=1}^{\hat{k}-1} \mu_i \mathbf{y}_i \right) \\
&= \gamma_{\hat{k}} \sum_{i=1}^{\hat{k}-1} \mu_i \mathbf{g}_{\hat{k}+1}^T \mathbf{H}_0 \mathbf{y}_i \\
&= 0.
\end{aligned}
$$

Thus, (2.15) holds for $k = \hat{k}$.

Finally, using (i) and (ii) from above,

$$
\begin{aligned}
\mathbf{s}_{\hat{k}+1} &= -\alpha_{\hat{k}+1} \mathbf{H}_{\hat{k}+1} \mathbf{g}_{\hat{k}+1} \\
&= -\alpha_{\hat{k}+1} \left(\gamma_{\hat{k}} \mathbf{P}_{\hat{k}}^T \mathbf{H}_0 \mathbf{Q}_{\hat{k}} \mathbf{g}_{\hat{k}+1} + \sum_{i=1}^{m_{\hat{k}}} \mathbf{w}_{i\hat{k}} \mathbf{z}_{i\hat{k}}^T \mathbf{g}_{\hat{k}+1} \right) \\
&= -\alpha_{\hat{k}+1} \gamma_{\hat{k}} \mathbf{P}_{\hat{k}}^T \mathbf{H}_0 \mathbf{g}_{\hat{k}+1}.
\end{aligned}
$$

Since $\mathbf{P}_{\hat{k}}^T$ maps any vector $\mathbf{v}$ into $\operatorname{span}\{\mathbf{v}, \mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}+1}\}$ by construction, there exist $\sigma_0, \ldots, \sigma_{\hat{k}+1}$ such that

$$
\mathbf{s}_{\hat{k}+1} = -\alpha_{\hat{k}+1} \gamma_{\hat{k}} \left(\mathbf{H}_0 \mathbf{g}_{\hat{k}+1} + \sum_{i=0}^{\hat{k}+1} \sigma_i \mathbf{s}_i \right).
$$

Hence,

$$
\mathbf{H}_0 \mathbf{g}_{\hat{k}+1} \in \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}+1}\},
$$

so

$$
\operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_{\hat{k}+1}\} \subseteq \operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}+1}\}.
$$

To show equality of the sets, we will show that $\mathbf{H}_0 \mathbf{g}_{\hat{k}+1}$ is linearly independent of $\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_{\hat{k}}\}$. (We already know that the vectors $\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_{\hat{k}}$ are linearly independent since they span the same space as the linearly independent set $\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}}\}$.) Suppose that $\mathbf{H}_0 \mathbf{g}_{\hat{k}+1}$ is not linearly independent. Then there exist $\phi_0, \ldots, \phi_{\hat{k}}$, not all zero, such that

$$
\mathbf{H}_0 \mathbf{g}_{\hat{k}+1} = \sum_{i=0}^{\hat{k}} \phi_i \mathbf{H}_0 \mathbf{g}_i.
$$

Recall that $\mathbf{g}_{\hat{k}+1}$ is orthogonal to $\{\mathbf{s}_0, \ldots, \mathbf{s}_{\hat{k}}\}$. By our induction assumption, this implies that $\mathbf{g}_{\hat{k}+1}$ is also orthogonal to $\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_{\hat{k}}\}$. Thus, for any j between 0 and $\hat{k}$,

$$
0 = \mathbf{g}_{\hat{k}+1}^T \mathbf{H}_0 \mathbf{g}_j = \left(\sum_{i=0}^{\hat{k}} \phi_i \mathbf{H}_0 \mathbf{g}_i \right)^T \mathbf{g}_j = \sum_{i=0}^{\hat{k}} \phi_i \mathbf{g}_i^T \mathbf{H}_0 \mathbf{g}_j = \phi_j \mathbf{g}_j^T \mathbf{H}_0 \mathbf{g}_j.
$$

Since $\mathbf{H}_0$ is positive definite and $\mathbf{g}_j$ is nonzero, we conclude that ϕ_j must be zero. Since this is true for every j between zero and $\hat{k}$, we have a contradiction. Thus, the set $\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_{\hat{k}+1}\}$ is linearly independent. Hence, (2.16) holds for $k = \hat{k}$.

($\Rightarrow$) Assume that (2.14)–(2.16) hold for all k such that $\mathbf{g}_{k+1} \neq 0$ but that (2.17) does not hold; i.e., there exist j and k such that $\mathbf{g}_{k+1} \neq 0$, j is between 0 and k, and

$$\mathbf{P}_k \mathbf{y}_j \notin \operatorname{span}\{\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}\}. \tag{2.18}$$

This will lead to a contradiction. By construction of $\mathbf{P}_k$, there exist $\mu_0, \ldots, \mu_k$ such that

$$\mathbf{P}_k \mathbf{y}_j = \sum_{i=0}^{k} \mu_i \mathbf{y}_i. \tag{2.19}$$

By assumption (2.18), μ_k must be nonzero. From (2.15), it follows that $\mathbf{g}_{k+1}^T \mathbf{H}_{k+1}^T \mathbf{y}_j = 0$. Using facts (i), (ii), and (iv) from before, (2.16), and (2.19), we get

$$\begin{aligned}
0 = \mathbf{g}_{k+1}^T \mathbf{H}_{k+1}^T \mathbf{y}_j &= \mathbf{g}_{k+1}^T \left(\gamma_k \mathbf{Q}_k^T \mathbf{H}_0 \mathbf{P}_k + \sum_{i=1}^{m_k} \mathbf{z}_{ik} \mathbf{w}_{ik}^T \right) \mathbf{y}_j \\
&= \gamma_k \mathbf{g}_{k+1}^T \mathbf{Q}_k^T \mathbf{H}_0 \mathbf{P}_k \mathbf{y}_j + \sum_{i=1}^{m_k} \mathbf{g}_{k+1}^T \mathbf{z}_{ik} \mathbf{w}_{ik}^T \mathbf{y}_j \\
&= \gamma_k \mathbf{g}_{k+1}^T \mathbf{H}_0 \mathbf{P}_k \mathbf{y}_j \\
&= \gamma_k \mathbf{g}_{k+1}^T \mathbf{H}_0 \left(\sum_{i=0}^{k} \mu_i \mathbf{y}_i \right) \\
&= \gamma_k \mu_k \mathbf{g}_{k+1}^T \mathbf{H}_0 \mathbf{y}_k \\
&= \gamma_k \mu_k \left(\mathbf{g}_{k+1}^T \mathbf{H}_0 \mathbf{g}_{k+1} - \mathbf{g}_{k+1}^T \mathbf{H}_0 \mathbf{g}_k \right) \\
&= \gamma_k \mu_k \mathbf{g}_{k+1}^T \mathbf{H}_0 \mathbf{g}_{k+1}.
\end{aligned}$$

Thus, since neither γ_k nor μ_k is zero, we must have

$$\mathbf{g}_{k+1}^T \mathbf{H}_0 \mathbf{g}_{k+1} = 0,$$

but this is a contradiction since $\mathbf{H}_0$ is positive definite and $\mathbf{g}_{k+1}$ was assumed to be nonzero. □

When a method produces conjugate search directions, we can say something about termination.

COROLLARY 2.3. *Suppose we have a method of the type described in Theorem* 2.2 *satisfying* (2.17). *Suppose further that* $\mathbf{H}_k \mathbf{g}_k \neq 0$ *whenever* $\mathbf{g}_k \neq 0$. *Then the scheme reproduces the iterates from the conjugate gradient method with preconditioner* $\mathbf{H}_0$ *and terminates in no more than* n *iterations.*

Proof. Let k be such that $\mathbf{g}_0, \ldots, \mathbf{g}_k$ are all nonzero and such that $\mathbf{H}_i \mathbf{g}_i \neq 0$ for $i = 0, \ldots, k$. Since we have a method of the type described in Theorem 2.2 satisfying (2.17), conditions (2.14)–(2.16) hold. We claim that the $(k+1)$st subspace of search directions, $\operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\}$, is equal to the $(k+1)$st Krylov subspace, $\operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, (\mathbf{H}_0 \mathbf{A})^k \mathbf{H}_0 \mathbf{g}_0\}$.

From (2.16), we know that $\operatorname{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_k\} = \operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_k\}$. We will show via induction that $\operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_k\} = \operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, (\mathbf{H}_0 \mathbf{A})^k \mathbf{H}_0 \mathbf{g}_0\}$. This base case is trivial, so assume that

$$\operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, \mathbf{H}_0 \mathbf{g}_i\} = \operatorname{span}\{\mathbf{H}_0 \mathbf{g}_0, \ldots, (\mathbf{H}_0 \mathbf{A})^i \mathbf{H}_0 \mathbf{g}_0\}$$

for some $i < k$. Now,

$$\mathbf{g}_{i+1} = \mathbf{A}\mathbf{x}_{i+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_i + \mathbf{s}_i) - \mathbf{b} = \mathbf{A}\mathbf{s}_i + \mathbf{g}_i,$$

and from (2.16) and the induction hypothesis,

$$\mathbf{s}_i \in \text{span}\{\mathbf{H}_0\mathbf{g}_0, \ldots, \mathbf{H}_0\mathbf{g}_i\} = \text{span}\{\mathbf{H}_0\mathbf{g}_0, \ldots, (\mathbf{H}_0\mathbf{A})^i\mathbf{H}_0\mathbf{g}_0\},$$

which implies that $\mathbf{H}_0\mathbf{A}\mathbf{s}_i \in \text{span}\{(\mathbf{H}_0\mathbf{A})\mathbf{H}_0\mathbf{g}_0, \ldots, (\mathbf{H}_0\mathbf{A})^{i+1}\mathbf{H}_0\mathbf{g}_0\}$. So,

$$\mathbf{H}_0\mathbf{g}_{i+1} \in \text{span}\{\mathbf{H}_0\mathbf{g}_0, \ldots, (\mathbf{H}_0\mathbf{A})^{i+1}\mathbf{H}_0\mathbf{g}_0\}.$$

Hence, the search directions span the Krylov subspace. Since the search directions are conjugate (2.15) and span the Krylov subspace, the iterates are the same as those produced by conjugate gradients with preconditioner $\mathbf{H}_0$.

Since we produce the same iterates as the conjugate gradient method and the conjugate gradient method is well known to terminate within n iterations, we can conclude that this scheme terminates in at most n iterations. □

Note that we require that $\mathbf{H}_k\mathbf{g}_k$ be nonzero whenever $\mathbf{g}_k$ is nonzero; this requirement is necessary since not all the methods produce positive definite updates and it is possible to construct an update that maps $\mathbf{g}_k$ to zero. If this were to happen, we would have a breakdown in the method.

The next corollary defines the role that the latest information ($\mathbf{s}_k$ and $\mathbf{y}_k$) plays in the formation of the kth $\mathbf{H}$-update.

COROLLARY 2.4. *Suppose we have a method of the type described in Theorem* 2.2 *satisfying* (2.17). *Suppose further that at the kth iteration* $\mathbf{P}_k$ *is composed of p projections of the form in* (2.2). *Then at least one of the projections must have* $\mathbf{u} = \sum_{i=0}^{k} \sigma_i \mathbf{y}_i$ *with* $\sigma_k \neq 0$. *Furthermore, if* $\mathbf{P}_k$ *is a single projection* ($p = 1$), *then* $\mathbf{v}$ *must be of the form* $\mathbf{v} = \rho_k\mathbf{s}_k + \rho_{k+1}\mathbf{s}_{k+1}$ *with* $\rho_k \neq 0$.

Proof. Consider the case of $p = 1$. We have

$$\mathbf{P}_k = \mathbf{I} - \frac{\mathbf{u}\mathbf{v}^T}{\mathbf{v}^T\mathbf{u}},$$

where $\mathbf{u} \in \text{span}\{\mathbf{y}_0, \ldots, \mathbf{y}_k\}$ and $\mathbf{v} \in \text{span}\{\mathbf{s}_0, \ldots, \mathbf{s}_{k+1}\}$. We will assume that

$$\mathbf{u} = \sum_{i=0}^{k} \sigma_i \mathbf{y}_i \quad \text{and} \quad \mathbf{v} = \sum_{i=0}^{k+1} \rho_i \mathbf{s}_i$$

for some scalars σ_i and ρ_i. By (2.17), there exist $\mu_0, \ldots, \mu_{k-1}$ such that

$$\mathbf{P}_k\mathbf{y}_k = \sum_{i=0}^{k-1} \mu_i \mathbf{y}_i.$$

Then

$$\mathbf{y}_k - \frac{\mathbf{v}^T\mathbf{y}_k}{\mathbf{v}^T\mathbf{u}}\mathbf{u} = \sum_{i=0}^{k-1} \mu_i \mathbf{y}_i,$$

and so

$$\frac{\mathbf{v}^T\mathbf{y}_k}{\mathbf{v}^T\mathbf{u}}\mathbf{u} = \mathbf{y}_k - \sum_{i=0}^{k-1} \mu_i \mathbf{y}_i. \tag{2.20}$$

From (2.15), the set $\{\mathbf{s}_0, \dots \mathbf{s}_k\}$ is conjugate and thus linearly independent. Since we are working with a quadratic, $\mathbf{y}_i = \mathbf{A}\mathbf{s}_i$ for all i, and since $\mathbf{A}$ is symmetric positive definite, the set $\{\mathbf{y}_0, \dots, \mathbf{y}_k\}$ is also linearly independent. So the coefficient of the $\mathbf{y}_k$ on the left-hand side of (2.20) must match that on the right-hand side, thus

$$\frac{\mathbf{v}^T\mathbf{y}_k}{\mathbf{v}^T\mathbf{u}}\sigma_k = 1.$$

Hence, $\sigma_k \neq 0$ and $\mathbf{y}_k$ must make a nontrivial contribution to $\mathbf{P}_k$.

Next we will show that $\rho_0 = \rho_1 = \cdots = \rho_{k-1} = 0$. Assume that j is between 0 and $k-1$. Then

$$\begin{aligned}
\mathbf{P}_k\mathbf{y}_j &= \mathbf{y}_j - \frac{\mathbf{v}^T\mathbf{y}_j}{\mathbf{v}^T\mathbf{u}}\mathbf{u} \\
&= \mathbf{y}_j - \frac{\left(\sum_{i=1}^{k+1}\rho_i\mathbf{s}_i\right)^T\mathbf{y}_j}{\mathbf{v}^T\mathbf{u}}\mathbf{u} \\
&= \mathbf{y}_j - \frac{\sum_{i=1}^{k+1}\rho_i\mathbf{s}_i^T\mathbf{A}\mathbf{s}_j}{\mathbf{v}^T\mathbf{u}}\mathbf{u} \\
&= \mathbf{y}_j - \frac{\rho_j\mathbf{s}_j^T\mathbf{A}\mathbf{s}_j}{\mathbf{v}^T\mathbf{u}}\mathbf{u}.
\end{aligned}$$

Now $\mathbf{s}_j\mathbf{A}\mathbf{s}_j$ is nonzero because $\mathbf{A}$ is positive definite. If ρ_j is nonzero, then the coefficient of $\mathbf{u}$ is nonzero, and so $\mathbf{y}_k$ must make a nontrivial contribution to $\mathbf{P}_k\mathbf{y}_j$, implying that $\mathbf{P}_k\mathbf{y}_j \notin \text{span}\{\mathbf{y}_0, \dots, \mathbf{y}_{k-1}\}$. This is a contradiction. Hence, $\rho_j = 0$.

To show that $\rho_k \neq 0$, consider $\mathbf{P}_k\mathbf{y}_k$. Suppose that $\rho_k = 0$. Then

$$\begin{aligned}
\mathbf{v}^T\mathbf{y}_k &= \rho_{k+1}\mathbf{y}_k^T\mathbf{s}_{k+1} + \rho_k\mathbf{y}_k^T\mathbf{s}_k \\
&= \rho_{k+1}\mathbf{s}_k^T\mathbf{A}\mathbf{s}_{k+1} \\
&= 0,
\end{aligned}$$

and so

$$\mathbf{P}_k\mathbf{y}_k = \mathbf{y}_k - \frac{\mathbf{v}^T\mathbf{y}_k}{\mathbf{v}^T\mathbf{u}}\mathbf{u} = \mathbf{y}_k.$$

This contradicts $\mathbf{P}_k\mathbf{y}_k \in \text{span}\{\mathbf{y}_0, \dots, \mathbf{y}_{k-1}\}$, so ρ_k must be nonzero.

Now we will discuss the $p > 1$ case. Label the $\mathbf{u}$-components of the p projections as $\mathbf{u}_1$ through $\mathbf{u}_p$. Then

$$\mathbf{P}_k\mathbf{y}_k = \mathbf{y}_k + \sum_{i=1}^{p}\gamma_i\mathbf{u}_i$$

for some scalars γ_1 through γ_p. Furthermore, each u_i can be written as a linear combination of $\{\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_k\}$, so

$$\mathbf{P}_k\mathbf{y}_k = \mathbf{y}_k + \sum_{i=1}^{p}\sum_{j=0}^{k}\gamma_i\sigma_{ij}\mathbf{y}_j$$

for some scalars σ_{10} through σ_{pk}. Since $\mathbf{P}_k\mathbf{y}_k \in \text{span}\{\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}\}$ and $\mathbf{y}_k \notin \text{span}\{\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}\}$, we must have

$$1 + \sum_{i=1}^{p} \gamma_i \sigma_{ik} = 0.$$

Thus σ_{ik} must be nonzero for some i, and we can conclude that at least one $\mathbf{u}_i$ must have a nontrivial contribution from $\mathbf{y}_k$. □

2.2. Examples of methods that reproduce the conjugate gradient iterates. Here are some specific examples of methods that fit the general form, satisfy condition (2.17) of Theorem 2.2, and thus terminate in at most n iterations. The conjugate gradient, L-BFGS, and DFP examples are well-known results, but Corollary 2.5 is original.

Example 6. The conjugate gradient method with preconditioner $\mathbf{H}_0$ (see (2.4)) satisfies condition (2.17) of Theorem 2.2 since

$$\mathbf{P}_k\mathbf{y}_j = \left(\mathbf{I} - \frac{\mathbf{y}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_k}\right)\mathbf{y}_j = \mathbf{0} \text{ for all } j = 0, \ldots, k.$$

Example 7. L-BFGS (see (2.6)) satisfies condition (2.17) of Theorem 2.2 since

$$\mathbf{P}_k\mathbf{y}_j = \begin{cases} \mathbf{0} & \text{for } j = k - m_k + 1, \ldots, k, \\ \mathbf{y}_j & \text{for } j = 0, \ldots, k - m_k. \end{cases}$$

Example 8. DFP (with full memory) (see (2.8)) satisfies condition (2.17) of Theorem 2.2. Consider $\mathbf{P}_k$ in the full-memory case. We have

$$\mathbf{P}_k = \prod_{i=0}^{k} \left(\mathbf{I} - \frac{\mathbf{y}_i\mathbf{y}_i\mathbf{H}_i^T}{\mathbf{y}_i^T\mathbf{H}_i\mathbf{y}_i}\right).$$

For full-memory DFP, $\mathbf{H}_i\mathbf{y}_j = \mathbf{s}_j$ for $j = 0, \ldots, i-1$. Using this fact, one can easily verify that $\mathbf{P}_k\mathbf{y}_j = \mathbf{0}$ for $j = 0, \ldots, k$. Therefore, full-memory DFP satisfies condition (2.17) of Theorem 2.2. The same reasoning does not apply to the limited-memory case, as we shall show in section 2.3.

The next corollary gives some ideas for other methods that are related to L-BFGS and terminate in at most n iterations on strictly convex quadratics.

COROLLARY 2.5. *The L-BFGS* (2.5) *method with exact line search will terminate in n iterations on an n-dimensional strictly convex quadratic function even if any combination of the following modifications is made to the update*:

1. *Vary the limited-memory constant, keeping* $m_k \geq 1$.
2. *Form the projections used in* $\mathbf{V}_k$ *from the most recent* $(\mathbf{s}_k, \mathbf{y}_k)$ *pair along with any set of* $m-1$ *other pairs from* $\{(\mathbf{s}_0, \mathbf{y}_0), \ldots, (\mathbf{s}_{k-1}, \mathbf{y}_{k-1})\}$.
3. *Form the projections used in* $\mathbf{V}_k$ *from the most recent* $(\mathbf{s}_k, \mathbf{y}_k)$ *pair along with any* $m-1$ *other linear combinations of pairs from* $\{(\mathbf{s}_0, \mathbf{y}_0), \ldots, (\mathbf{s}_{k-1}, \mathbf{y}_{k-1})\}$.
4. *Iteratively rescale* $\mathbf{H}_0$.

Proof. For each variant, we show that the method fits the general form in (2.1), satisfies condition (2.17) of Theorem 2.2, and hence terminates by Corollary 2.3:

1. Let $m > 0$ be any value which may change from iteration to iteration, and define

$$\mathbf{V}_{ik} = \prod_{j=i}^{k} \left(\mathbf{I} - \frac{\mathbf{y}_j\mathbf{s}_j^T}{\mathbf{s}_j^T\mathbf{y}_j}\right).$$

Choose

$$\gamma_k = 1, \quad m_k = \min\{k+1, m\},$$
$$\mathbf{P}_k = \mathbf{Q}_k = \mathbf{V}_{k-m_k+1,k}, \text{ and}$$
$$\mathbf{w}_{ik} = \mathbf{z}_{ik} = \frac{(\mathbf{V}_{k-m_k+i+1,k})^T(\mathbf{s}_{k-m_k+i})}{\sqrt{(\mathbf{s}_{k-m_k+i})^T(\mathbf{y}_{k-m_k+i})}}.$$

These choices fit the general form. Furthermore,

$$\mathbf{P}_k\mathbf{y}_j = \begin{cases} \mathbf{0} & \text{if } j = k-m_k, k-m_k+1, \ldots, k, \\ \mathbf{y}_j & \text{if } j = 0, 1, \ldots, k-m_k-1, \end{cases}$$

so this variation satisfies condition (2.17) of Theorem 2.2.

2. This is a special case of the next variant.

3. At iteration k, let $(\hat{\mathbf{s}}_k^{(i)}, \hat{\mathbf{y}}_k^{(i)})$ denote the ith ($i = 1, \ldots, m-1$) choice of any linear combination from the span of the set

$$\{(\mathbf{s}_0, \mathbf{y}_0), \ldots, (\mathbf{s}_{k-1}, \mathbf{y}_{k-1})\},$$

and let $(\hat{\mathbf{s}}_k^{(m)}, \hat{\mathbf{y}}_k^{(m)}) = (\mathbf{s}_k, \mathbf{y}_k)$. Define

$$\mathbf{V}_{ik} = \prod_{j=i}^{m} \left(\mathbf{I} - \frac{(\hat{\mathbf{y}}_k^{(i)})(\hat{\mathbf{s}}_k^{(i)})^T}{(\hat{\mathbf{s}}_k^{(i)})^T(\hat{\mathbf{y}}_k^{(i)})} \right).$$

Choose

$$\gamma_k = 1, \quad m_k = \min\{k+1, m\},$$
$$\mathbf{P}_k = \mathbf{Q}_k = \mathbf{V}_{1,k}, \text{ and}$$
$$\mathbf{w}_{ik} = \mathbf{z}_{ik} = \frac{(\mathbf{V}_{i+1,k})^T(\hat{\mathbf{s}}_k^{(i)})}{\sqrt{(\hat{\mathbf{s}}_k^{(i)})^T(\hat{\mathbf{y}}_k^{(i)})}}.$$

These choices satisfy the general form (2.1). Furthermore,

$$\mathbf{P}_k\mathbf{y}_j = \begin{cases} \mathbf{0} & \text{if } \mathbf{y}_j = \mathbf{y}_k^{(i)} \text{ for some } i, \\ \mathbf{y}_j & \text{otherwise.} \end{cases}$$

Hence, this variation satisfies condition (2.17) of Theorem 2.2.

4. Let γ_k in (2.1) be the scaling constant and choose the other vectors and matrices as in L-BFGS (2.6).

Combinations of variants are left to the reader. □

Remark 1. Part 3 of the previous corollary shows that the "accumulated step" method of Gill and Murray [11] terminates on quadratics.

Remark 2. Part 4 of the previous corollary shows that scaling does not affect termination in L-BFGS. In fact, for any method that fits the general form, it is easy to see that scaling will not affect termination on quadratics.

2.3. Examples of methods that do not reproduce the conjugate gradient iterates. We will discuss several methods that fit the general form given in (2.1) but do not satisfy the conditions of Theorem 2.2.

Example 9. Steepest descent (see (2.3)) does not satisfy condition (2.17) of Theorem 2.2 and thus does not produce conjugate search directions. This fact is well known; see, e.g., Luenberger [17].

Example 10. L-DFP (see (2.8)) with $m < n$ does not satisfy the condition on $\mathbf{P}_k$ (2.17) for all k, and so the method will not produce conjugate directions. This fact was previously unknown.

For example, suppose that we have a convex quadratic with

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Using a limited-memory constant of $m = 1$ and exact arithmetic, it can be seen that the iteration does not terminate within the first 20 iterations of L-DFP with $\mathbf{H}_0 = \mathbf{I}$. The MAPLE notebook file used to compute this example is available on the World Wide Web [10].

Remark 3. Using the above example, we can easily see that no limited-memory Broyden family method except L-BFGS terminates within the first n iterations.

3. Update-skipping variations for Broyden family quasi-Newton algorithms. The previous section discussed limited-memory methods that behave like conjugate gradients on n-dimensional strictly convex quadratic functions. In this section, we are concerned with methods that skip some updates. The average computation cost per iteration is reduced and memory can be saved if the quasi-Newton matrix is stored implicitly. We establish conditions under which finite termination is preserved but delayed for the Broyden family.

3.1. Termination when updates are skipped. It was shown by Powell [27] that if we skip every other update and take *direct prediction steps* (i.e., steps of length one) in a Broyden family method, then the procedure will terminate in no more than $2n+1$ iterations on an n-dimensional strictly convex quadratic function. An alternate proof of this result is given by Nazareth [22].

We will prove a related result. Suppose that we are doing *exact line searches* using a Broyden family quasi-Newton method on a strictly convex quadratic function and decide to "skip" p updates to $\mathbf{H}$ (i.e., choose $\mathbf{H}_{k+1} = \mathbf{H}_k$ on p occasions). Then the algorithm terminates in no more than $n + p$ iterations. The algorithm is somewhat more robust than using direct prediction steps; it does not matter which updates are skipped or if multiple updates are skipped in a row.

THEOREM 3.1. *Suppose that a Broyden family method using exact line searches is applied to an n-dimensional strictly convex quadratic function and p updates are skipped. Let*

$$J(k) = \{j \le k : \text{ the update at iteration } j \text{ is not skipped}\}.$$

Then for all $k = 0, 1, \ldots$

$$\mathbf{g}_{k+1}^T \mathbf{s}_j = 0 \quad \text{for all } j \in J(k) \tag{3.1}$$

and

$$\mathbf{s}_{k+1}^T \mathbf{A} \mathbf{s}_j = 0 \quad \text{for all } j \in J(k). \tag{3.2}$$

Furthermore, the method terminates in at most $n + p$ iterations at the exact minimizer.

Proof. We will use induction on k to show (3.1) and

$$\mathbf{H}_{k+1}\mathbf{y}_j = \mathbf{s}_j \text{ for all } j \in J(k). \tag{3.3}$$

Then (3.2) follows easily since, for all $j \in J(k)$,

$$\begin{aligned}\mathbf{s}_{k+1}^T \mathbf{A}\mathbf{s}_j &= -\alpha_{k+1}\mathbf{g}_{k+1}^T \mathbf{H}_{k+1}\mathbf{y}_j \\ &= -\alpha_{k+1}\mathbf{g}_{k+1}^T \mathbf{s}_j \\ &= 0.\end{aligned}$$

Let k_0 be the least value of k such that $J(k)$ is nonempty; i.e., $J(k_0) = \{k_0\}$. Then $\mathbf{g}_{k_0+1}$ is orthogonal to $\mathbf{s}_{k_0}$ since line searches are exact, and $\mathbf{H}_{k_0+1}\mathbf{y}_{k_0} = \mathbf{s}_{k_0}$ since all members of the Broyden family satisfy the secant condition. Hence, the base case is true. Now assume that (3.1) and (3.3) hold for all values of $k = 0, 1, \ldots, \hat{k}-1$. We will show that they also hold for $k = \hat{k}$.

Case 1. Suppose that $\hat{k} \notin J(\hat{k})$. Then $\mathbf{H}_{\hat{k}+1} = \mathbf{H}_{\hat{k}}$ and $J(\hat{k}-1) = J(\hat{k})$, so, for any $j \in J(\hat{k})$,

$$\mathbf{g}_{\hat{k}+1}^T \mathbf{s}_j = 0 \tag{3.4}$$

and

$$\mathbf{H}_{\hat{k}+1}\mathbf{y}_j = \mathbf{H}_{\hat{k}}\mathbf{y}_j = \mathbf{s}_j.$$

Case 2. Suppose that $\hat{k} \in J(\hat{k})$. Then $\mathbf{H}_{\hat{k}+1}$ satisfies the secant condition and $J(\hat{k}) = J(\hat{k}-1) \cup \{\hat{k}\}$. Now $\mathbf{g}_{\hat{k}+1}$ is orthogonal to $\mathbf{s}_k$ since the line searches are exact, and it is orthogonal to the older $\mathbf{s}_j$ by the argument in (3.4). The secant condition guarantees that $\mathbf{H}_{\hat{k}+1}\mathbf{y}_{\hat{k}} = \mathbf{s}_{\hat{k}}$, and, for $j \in J(\hat{k})$ but $j \neq \hat{k}$, we have

$$\begin{aligned}\mathbf{H}_{\hat{k}+1}\mathbf{y}_j &= \mathbf{H}_{\hat{k}}\mathbf{y}_j + \frac{\mathbf{s}_{\hat{k}}\mathbf{s}_{\hat{k}}^T}{\mathbf{s}_{\hat{k}}^T\mathbf{y}_{\hat{k}}}\mathbf{y}_j - \frac{\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}}{\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}\mathbf{y}_j \\ &\quad + \phi\,(\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}})\left(\frac{\mathbf{s}_{\hat{k}}}{\mathbf{s}_{\hat{k}}^T\mathbf{y}_{\hat{k}}} - \frac{\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}{\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}\right)\left(\frac{\mathbf{s}_{\hat{k}}}{\mathbf{s}_{\hat{k}}^T\mathbf{y}_{\hat{k}}} - \frac{\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}{\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}\right)^T\mathbf{y}_j \\ &= \mathbf{s}_j + \frac{\mathbf{s}_{\hat{k}}^T\mathbf{A}\mathbf{s}_j}{\mathbf{s}_{\hat{k}}^T\mathbf{y}_{\hat{k}}}\mathbf{s}_{\hat{k}} - \frac{\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}\mathbf{y}_{\hat{k}}^T\mathbf{s}_j}{\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}} \\ &\quad + \phi\,(\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}})\left(\frac{\mathbf{s}_{\hat{k}}}{\mathbf{s}_{\hat{k}}^T\mathbf{y}_{\hat{k}}} - \frac{\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}{\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}\right)\left(\frac{\mathbf{s}_{\hat{k}}^T\mathbf{A}\mathbf{s}_j}{\mathbf{s}_{\hat{k}}^T\mathbf{y}_{\hat{k}}} - \frac{\mathbf{y}_{\hat{k}}^T\mathbf{s}_j}{\mathbf{y}_{\hat{k}}^T\mathbf{H}_{\hat{k}}\mathbf{y}_{\hat{k}}}\right) \\ &= \mathbf{s}_j.\end{aligned}$$

In either case, the induction result follows.

Suppose that we skip p updates. Then the set $J(n-1+p)$ has cardinality n. Without loss of generality, assume that the set $\{\mathbf{s}_i\}_{i \in J(n-1+p)}$ has no zero elements. From (3.2), the vectors are linearly independent. By (3.1),

$$\mathbf{g}_{n+p}^T \mathbf{s}_j = 0 \quad \text{for all } j \in J(n-1+p),$$

and so $\mathbf{g}_{n+p}$ must be zero. This implies that $\mathbf{x}_{n+p}$ is the exact minimizer of f. ∎

3.2. Loss of termination for update skipping with limited-memory. Unfortunately, updates that use both limited-memory and repeated update-skipping do not produce conjugate search directions for n-dimensional strictly convex quadratics, and the termination property is lost. We will show a simple example.

Example 11. Suppose that we have a convex quadratic with

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

We apply L-BFGS with limited-memory constant $m = 1$ and $\mathbf{H}_0 = \mathbf{I}$ and skip every other update to $\mathbf{H}$. Using exact arithmetic in MAPLE, we observe that the process does not terminate even after 100 iterations [10]. Note that, according to Corollary 2.4, we would still be guaranteed termination if we used the most recent information in each update.

4. Experimental results. Thus far we have only given results for convex quadratic functions. While termination on quadratics is beautiful in theory, it does not necessarily yield insight into how these methods will do in practice. In this section, we develop and compare some of the new methods. We describe the collection of test problems in section 4.2. Complete numerical results, many graphs of the numerical results, and the original FORTRAN code are available [10].

4.1. Motivation. We will not present any new results relating to convergence of these algorithms on general functions; however, many of these can be shown to converge using the convergence analysis presented in section 7 of [16]. In [16], Liu and Nocedal show that an L-BFGS method implemented with a line search that satisfies the strong Wolfe conditions (see section 4.3 for a definition) is R-linearly convergent on a convex function that satisfies a few modest conditions.

4.2. Test problems. For our test problems, we used the CUTE by Bongartz, Conn, Gould, and Toint. The package is documented in [3] and can be obtained via the World Wide Web [2] or via ftp [1]. The package contains a large collection of test problems as well as the interfaces necessary for using the problems. We chose a collection of 22 unconstrained problems. The problems ranged in size from 10 to 10,000 variables, but each took L-BFGS with limited-memory constant $m = 5$ at least 60 iterations to solve. Table 4.1 enumerates the problems, giving the SIF file name, the dimension (n), and a description for each problem. The CUTE package also provides a starting point (x_0) for each problem.

4.3. Test environment. We used FORTRAN77 code on an SGI Indigo^2 to run the algorithms, with FORTRAN BLAS routines from NETLIB. We used the compiler's default optimization level.

Figure 2.1 outlines the general quasi-Newton implementation that we followed. For the line search, we use the routines `cvsrch` and `cstep` written by Jorge J. Moré and David Thuente from a 1983 version of `MINPACK`. This line search routine finds an α that meets the strong Wolfe conditions

$$f(\mathbf{x} + \alpha\mathbf{d}) \leq f(\mathbf{x}) + \omega_1 \alpha \mathbf{g}(\mathbf{x})^T \mathbf{d}, \tag{4.1}$$

$$|\mathbf{g}(\mathbf{x} + \alpha\mathbf{d})^T \mathbf{d}| \leq \omega_2 |\mathbf{g}(\mathbf{x})^T \mathbf{d}|; \tag{4.2}$$

see, e.g., Nocedal [24]. We used $\omega_1 = 1.0 \times 10^{-4}$ and $\omega_2 = 0.9$. Except for the first iteration, we always attempt a step length of 1.0 first and only use an alternate value

TABLE 4.1
Test problem collection. Each problem was chosen from the CUTE package.

No.	SIF name	n	Description and reference
1	EXTROSNB	10	Extended Rosenbrock function (nonseparable version) [30, Problem 10].
2	WATSONS	31	Watson problem [18, Problem 20].
3	TOINTGOR	50	Toint's operations research problem [29].
4	TOINTPSP	50	Toint's PSP operations research problem [29].
5	CHNROSNB	50	Chained Rosenbrock function [29].
6	ERRINROS	50	Nonlinear problem similar to CHNROSNB [3].
7	FLETCHBV	100	Fletcher's boundary value problem [9, Problem 1].
8	FLETCHCR	100	Fletcher's chained Rosenbrock function [9, Problem 2].
9	PENALTY2	100	Second penalty problem [18, Problem 24].
10	GENROSE	500	Generalized Rosenbrock function [19, Problem 5].
11	BDQRTIC	1000	Quartic with a banded Hessian with bandwidth = 9 [6, Problem 61].
12	BROYDN7D	1000	Seven diagonal variant of the Broyden tridiagonal system with a band away from diagonal [29].
13	PENALTY1	1000	First penalty problem [18, Problem 23].
14	POWER	1000	Power problem by Oren [26].
15	MSQRTALS	1024	The dense matrix square root problem by Nocedal and Liu (case 0) seen as a nonlinear equation problem [4, Problem 204].
16	MSQRTBLS	1025	The dense matrix square root problem by Nocedal and Liu (case 1) seen as a nonlinear equation problem [4, Problem 201].
17	CRAGGLVY	5000	Extended Cragg and Levy problem [30, Problem 32].
18	NONDQUAR	10000	Nondiagonal quartic test problem [6, Problem 57].
19	POWELLSG	10000	Extended Powell singular function [18, Problem 13].
20	SINQUAD	10000	Another function with nontrivial groups and repetitious elements [13].
21	SPMSRTLS	10000	Liu and Nocedal tridiagonal matrix square root problem [4, Problem 151].
22	TRIDIA	10000	Shanno's TRIDIA quadratic tridiagonal problem [30, Problem 8].

if 1.0 does not satisfy the Wolfe conditions. In the first iteration, we initially try a step length equal to $\|\mathbf{g}_0\|^{-1}$. The remaining line search parameters are detailed in Table 4.2.

We generate the matrix $\mathbf{H}_k$ by either the limited-memory update or one of the variations described in section 4.4, storing the matrix implicitly in order to save both memory and computation time.

TABLE 4.2
Line search parameters.

Variable	Value	Description
STP	1.0	Step length to try first.
XTOL	1.0×10^{-15}	Relative width of interval of uncertainty.
STPMIN	1.0×10^{-15}	Minimum step length.
STPMAX	1.0×10^{15}	Maximum step length.
MAXFEV	20	Maximum number of function evaluations.

We terminate at iteration k if any of the following conditions is met:

1. the iterate satisfies

$$\frac{\|\mathbf{g}_k\|}{\|\mathbf{x}_k\|} < 1.0 \times 10^{-5};$$

2. the line search fails to satisfy both (4.1) and (4.2); or
3. the number of iterations exceeds 3000.

We say that the iterates have converged if the first condition is satisfied. Otherwise, the method has failed.

4.4. L-BFGS and its variations. We tried a number of variations to the standard L-BFGS algorithm. L-BFGS and these variations are described in this subsection and summarized in Table 4.3.

TABLE 4.3
Description of numerical algorithms.

No.	Reference	Brief Description
0	Section 4.4.1	L-BFGS with no options.
1	Section 4.4.2, Variation 4	Allow m to vary iteratively, basing the choice of m on $\|\mathbf{g}\|/\|\mathbf{x}\|$ and allowing m to decrease.
2	Section 4.4.3	Dispose of old information if the step length is greater than one.
3	Section 4.4.4, Variation 1	Back up if the current iteration is odd.
4	Section 4.4.4, Variation 3	Back up if a step length of 1.0 was used in the last iteration.
5	Section 4.4.5, Variation 2	Merge the 2nd and 3rd most recent $(\mathbf{s}, \mathbf{y})$ pairs if the corresponding step lengths were 1 and neither pair is the result of a previous merge.
6	Section 4.4.6, Variation 1	Skip update on odd iterations.
7	Algorithm 2 and Algorithm 4	Dispose of old information and back up on the next iteration if the step length is greater than one.
8	Algorithm 6 and Algorithm 1	Merge if we did not do a merge in the last iteration and there are at least two old $\mathbf{s}$ vectors to merge, and allow m to vary iteratively, basing the choice of m on $\|\mathbf{g}\|/\|\mathbf{x}\|$ and allowing m to decrease.

4.4.1. L-BFGS: Algorithm 0. The L-BFGS update is given in (2.5) and described fully by Byrd, Nocedal, and Schnabel [5].

The storage costs are $O(mn)$, rather than n^2 required by BFGS. The computation of $\mathbf{Hg}$ takes at most $O(mn)$ operations rather than $O(n^2)$.

We are using L-BFGS as our basis for comparison. For information on the performance of L-BFGS, see Liu and Nocedal [16] and Nash and Nocedal [20].

4.4.2. Varying m iteratively: Algorithm 1. In typical implementations of L-BFGS, m is fixed throughout the iterations; once m updates have accumulated, m updates are always used. We considered the possibility of varying m iteratively, preserving finite termination on convex quadratics. Using an argument similar to that presented in [16], we can also prove that this algorithm has a linear rate of convergence on a convex function that satisfies a few modest conditions.

We scaled m in relation to the size of $\|\mathbf{g}\|/\max\{1, \|\mathbf{x}\|\}$. Let m_k be the number of iterates saved at the kth iteration, with $m_0 = 1$. Here, think of m as the maximum allowable value of m_k. Let the convergence test be given by $\|\mathbf{g}_k\|/\max\{1, \|\mathbf{x}_k\|\} < \epsilon$.

TABLE 4.4

The number of failures of the algorithms on the 22 test problems. An algorithm is said to have "failed" on a particular problem if a line search fails or the maximum allowable number of iterations (3000 in our case) is exceeded.

Alg. No.	$m = 5$	$m = 10$	$m = 15$	$m = 50$
0	1	0	0	1
1	1	0	0	1
2	0	0	0	0
3	1	0	0	1
4	0	0	0	0
5	1	0	0	1
6	12	12	12	12
7	0	0	0	0
8	3	1	0	1

TABLE 4.5

Function evaluations comparison. The first number in each entry is the number of times the algorithm did as well as or better than normal L-BFGS in terms of function evaluations. The second number is the total number of problems solved by at least one of the two methods (the algorithm and/or L-BFGS).

Alg. No.	$m = 5$	$m = 10$	$m = 15$	$m = 50$
1	12/21	17/22	15/22	16/21
2	19/22	20/22	20/22	21/22
3	21/21	22/22	22/22	21/21
4	12/22	14/22	12/22	15/22
5	3/22	4/22	4/22	4/22
6	1/21	1/22	1/22	1/21
7	12/22	13/22	12/22	14/22
8	1/22	2/22	4/22	4/22

Then the formula for m_k at iteration k is

$$m_k = \min\left\{ m_{k-1} + 1, \left\lfloor (m-1) \frac{\log \frac{\|\mathbf{g}_k\|}{\max\{1, \|\mathbf{x}_k\|\}} - \log \frac{\|\mathbf{g}_0\|}{\max\{1, \|\mathbf{x}_0\|\}}}{\log 100\epsilon - \log \frac{\|\mathbf{g}_0\|}{\max\{1, \|\mathbf{x}_0\|\}}} \right\rfloor + 1 \right\}.$$

We used four values of m: 5, 10, 15, and 50. The results are summarized in Tables 4.4–4.8. More extensive results are given in [10].

Table 4.4 shows that this algorithm had the same number of failures as L-BFGS.

Table 4.5 compares the algorithm to L-BFGS in terms of function evaluations. The number of times that the algorithm used *as few or fewer* function evaluations than L-BFGS is listed relative to the total number of admissible problems. Problems are admissible if either of the two methods solved it. This algorithm used as few or fewer function evaluations than L-BFGS for over half the test problems.

Table 4.6 compares this algorithm to L-BFGS in terms of time. The entries are similar to those in Table 4.5. Observe that Algorithm 1 did very well in terms of time, doing as well or better than L-BFGS on approximately 80% of the problems.

For each problem, we computed the ratio of the number of function evaluations for the algorithm to the number of function evaluations for L-BFGS. Table 4.7 lists the means of these ratios. A mean below 1.0 implies that the algorithm does better than L-BFGS on average. The average is better for the first algorithm in 2 out of 4 cases. Observe, however, that all the means are close to one.

TABLE 4.6
Time comparison. The first number in each entry is the number of times the algorithm did as well as or better than normal L-BFGS in terms of time. The second number is the total number of problems solved by at least one of the two methods (the algorithm and/or L-BFGS).

Alg. No.	$m = 5$	$m = 10$	$m = 15$	$m = 50$
1	17/21	18/22	20/22	18/21
2	15/22	13/22	14/22	15/22
3	16/21	19/22	15/22	15/21
4	11/22	7/22	6/22	5/22
5	5/22	10/22	13/22	17/22
6	1/21	1/22	1/22	2/21
7	11/22	8/22	5/22	4/22
8	9/22	16/22	16/22	18/22

TABLE 4.7
Mean function evaluations ratios for each algorithm compared to L-BFGS. Problems for which either method failed are not used in this mean.

Alg. No.	$m = 5$	$m = 10$	$m = 15$	$m = 50$
1	0.998	1.297	0.970	1.000
2	1.021	0.971	1.005	1.010
3	1.000	1.000	1.000	1.000
4	0.991	1.677	1.507	0.891
5	1.137	1.178	1.244	1.373
6	8.227	8.666	9.073	9.308
7	0.981	1.023	0.924	0.918
8	1.406	1.161	1.178	1.394

This algorithm tends to save fewer vectors than L-BFGS since m_k is typically less than m and so less work is done computing $\mathbf{H}_k\mathbf{g}_k$. Table 4.8 gives the mean of the ratios of time to solve for each value of m in each algorithm. Note that most of the ratios are far below one.

4.4.3. Disposing of old information: Algorithm 2. We may decide that we should stop using outdated information. For example, we may choose to keep only the most recent information whenever we take a big step, since the old information may not be relevant to the new neighborhood. We use the following test: If the last step length was bigger than 1, discard all but the most recent $\mathbf{s}$ and $\mathbf{y}$ pair.

The algorithm performed nearly the same as L-BFGS. There was substantial deviation on only one or two problems for each value of m, and this seemed evenly divided in terms of better and worse. From Table 4.4, we see that this algorithm successfully converged on every problem. Table 4.5 shows that it almost always did as well or better than L-BFGS in terms of function evaluations. However, Table 4.7 shows that the differences were minor. The algorithm generally took less time than L-BFGS (Table 4.6), but again, considering the mean ratios of time (Table 4.8), the differences were minor.

4.4.4. Backing up in the update to H: Algorithms 3-4. As discussed in section 2.2, if we always use the most recent $\mathbf{s}$ and $\mathbf{y}$ in the update, we preserve quadratic termination regardless of which older values of $\mathbf{s}$ and $\mathbf{y}$ we use.

Using this idea, we created some algorithms. Under certain conditions, we discard the next most recent values of $\mathbf{s}$ and $\mathbf{y}$ although we still use the *most* recent $\mathbf{s}$ and $\mathbf{y}$ vectors and any other vectors that have been saved from previous iterations. We call

TABLE 4.8
Mean time ratios for each algorithm compared to L-BFGS. Problems for which either method failed are not used in this mean.

Alg. No.	$m = 5$	$m = 10$	$m = 15$	$m = 50$
1	0.907	1.119	0.823	0.856
2	1.041	0.969	0.993	1.004
3	1.007	0.983	0.977	0.995
4	1.057	1.421	1.426	1.425
5	1.083	1.082	0.983	0.960
6	5.008	4.046	3.527	2.646
7	1.053	1.166	1.089	1.399
8	1.258	0.927	0.859	0.974

this "backing up" because it is as if we back up over the next most recent update. These algorithms used the following tests to trigger backing up:

1. The current iteration is odd.
2. A step length of 1.0 was used in the last iteration.

Algorithm 3 did not fail at all. See Table 4.4 for more information.

Backing up on odd iterations (Algorithm 3) seemed to have almost no effect on the number of function evaluations (Table 4.7) and little effect on the time (Table 4.8).

In Algorithm 4, we back up if the previous step length was one. This wipes out the data from the previous iteration after it has been used in one update. It is an improvement over L-BFGS in terms of function evaluations; in fact, this algorithm has the best function evaluation ratio for the $m = 50$ case (Table 4.7). Unfortunately, this algorithm did not compete with L-BFGS in terms of time (Table 4.8).

4.4.5. Merging s and y information in the update: Algorithm 5. Yet another idea is to "merge" $\mathbf{s}$ data to use less storage and computation time. By merging, we mean forming some linear combination of various $\mathbf{s}$ vectors. The $\mathbf{y}$ vectors would be merged correspondingly. Corollary 2.5 shows that as long as the most recent $\mathbf{s}$ and $\mathbf{y}$ are used without merge, old $\mathbf{s}$ vectors may be replaced by any linear combination of the old $\mathbf{s}$ vectors in L-BFGS.

We used this idea in the following way: we merged the 2nd and 3rd most recent $(\mathbf{s}, \mathbf{y})$ pairs if the corresponding step lengths were 1 and neither pair was the result of a previous merge. A merge is accomplished by adding the two pairs together and replacing the two pairs with the single "sum" pair.

Algorithm 5 is better than L-BFGS in terms of time, especially for the larger values of m (Tables 4.6 and 4.8). Unfortunately, this reflects only a saving in the amount of linear algebra required. The number of function evaluations generally is larger for this algorithm than L-BFGS (Tables 4.5 and 4.7).

4.4.6. Skipping updates to H: Algorithm 6. If every other update to $\mathbf{H}$ is skipped and a step length of one is always chosen, BFGS will terminate in $2n$ iterations on a strictly convex quadratic function. The same holds true when doing an exact line search. (See section 3.) Unfortunately, neither property holds in the limited-memory case. We will, however, try an algorithm motivated by this idea.

Skipping on odd updates (Algorithm 6) did extremely well for every value of m only on problem 1. Otherwise, it did very badly.

4.4.7. Combined methods: Algorithms 7–8. We did some experimentation with combinations of methods described in the previous sections.

In Algorithm 7, we combined Algorithms 2 and 4; we dispose of old information and back up on the next iterations if the step length is greater than one. Essentially we are assuming that we have stepped out of the region being modeled by the quasi-Newton matrix if we take a long step, and we should thus rid the quasi-Newton matrix of that information. This algorithm did well in terms of function evaluations, having mean ratios of less than one for three values of m (Table 4.7), but it did not do as well in terms of time.

In Algorithm 8, we combined merging (Algorithm 5) and varying m (Algorithm 1). This algorithm did well in terms of time for larger m (Table 4.8) but not in terms of function evaluations (Table 4.7).

5. Conclusions. There is a spectrum of quasi-Newton methods, ranging from those that require the storage of an $n \times n$ approximate Hessian (e.g., the Broyden family) to those that require only the storage of a few vectors (e.g., conjugate gradients). Limited-memory quasi-Newton methods fall in between these extremes in terms of performance and storage. There are other methods that fall into the middle ground; for example, conjugate gradient methods such as those proposed by Shanno [28] and Nazareth [21], the truncated-Newton method [25, 7], and the partitioned quasi-Newton method [14].

We have characterized which limited-memory quasi-Newton methods fitting a general form (2.1) have the property of producing conjugate search directions on convex quadratics. We have shown that L-BFGS is the only Broyden family member that has a limited-memory analog with this property. We also considered update-skipping, something that may seem attractive in a parallel environment. We show that update skipping on quadratic problems is acceptable for full-memory Broyden family members in that it only delays termination, but that we lose the property of finite termination if we both limit memory and skip updates.

We have also introduced some simple-to-implement modifications of the standard L-BFGS algorithm that seem to behave well on some practical problems.

REFERENCES

[1] I. Bongartz, A. R. Conn, N. Gould, and P. L. Toint, CUTE ftp site, ftp://thales.math.fundp.ac.be/pub/cute.

[2] I. Bongartz, A. R. Conn, N. Gould, and P. L. Toint, CUTE home page, http://www.dci.clrc.ac.uk/Activity.asp?CUTE.

[3] I. Bongartz, A. R. Conn, N. Gould, and P. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[4] A. Buckley, *Test functions for unconstrained minimization*, Tech. Report TR 1989CS-3, Mathematics, Statistics and Computing Centre, Dalhousie University, Halifax (CDN), 1989.

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

[6] A. Conn, N. Gould, M. Lescrenier, and P. Toint, *Performance of a multifrontal scheme for partially separable optimization*, Tech. Report 88/4, Department of Mathematics, FUNDP, Namur, Belgium, 1988. Cited in [1, 2].

[7] R. S. Dembo and T. Steihaug, *Truncated-Newton algorithms for large-scale unconstrained optimization*, Math. Programming, 26 (1983), pp. 190–212.

[8] J. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Series in Computational Mathematics, Prentice-Hall, Englewood Cliffs, NJ, 1983. Reprinted by Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.

[9] R. Fletcher, *An optimal positive definite update for sparse Hessian matrices*, Numerical Analysis NA/145, Technical report, University of Dundee, 1992. Cited in [1, 2].

[10] T. Gibson, D. O'Leary, and L. Nazareth, L-BFGS with Variations home page, http://www.cs.umd.edu/users/oleary/LBFGS/index.html (1996).
[11] P. E. Gill and W. Murray, *Conjugate-gradient methods for large-scale nonlinear optimization*, Tech. Report SOL 79-15, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA, 1979.
[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, 1989.
[13] N. Gould, Private communication to authors of [3], 1989. Cited in [1, 2].
[14] A. Griewank and P. L. Toint, *Partitioned variable metric updates for large structured optimization problems*, Numer. Math., 39 (1982), pp. 119–137.
[15] H. Khalfan, R. Byrd, and R. Schnabel, *A theoretical and experimental study of the symmetric rank one update*, Tech. Report CU-CS-489-90, Department of Computer Science, University of Colorado at Boulder, 1990.
[16] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Programming, 45 (1989), pp. 503–528.
[17] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed., Addison-Wesley, Reading, MA, 1984.
[18] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.
[19] S. Nash, *Newton-type minimization via the Lanczos process*, SIAM J. Numer. Anal., 21 (1984), pp. 770–788.
[20] S. G. Nash and J. Nocedal, *A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization*, SIAM J. Optim., 1 (1991), pp. 358–372.
[21] L. Nazareth, *A relationship between BFGS and conjugate gradient algorithms and its implications for new algorithms*, SIAM J. Numer. Anal., 16 (1979), pp. 794–800.
[22] L. Nazareth, *On the BFGS Method*, unpublished manuscript, University of California at Berkeley, Berkeley, CA, 1981.
[23] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comp., 35 (1980), pp. 773–782.
[24] J. Nocedal, *Theory of algorithms for unconstrained optimization*, in Acta Numerica (1991), Cambridge University Press, London, 1992, pp. 199–242.
[25] D. P. O'Leary, *A discrete Newton algorithm for minimizing a function of many variables*, Math. Programming, 23 (1982), pp. 20–33.
[26] S. Oren, *Self-scaling variable metric algorithms, Part* II*: Implementation and experiments*, Management Science, 20 (1974), pp. 863–874. Cited in [1, 2].
[27] M. J. D. Powell, *Quadratic termination properties of minimization algorithms* I: *Statement and discussion of results*, J. Inst. Math. Appl., 10 (1972), pp. 333–342.
[28] D. F. Shanno, *Conjugate gradient methods with inexact line searches*, Math. Oper. Res., 3 (1978), pp. 244–256.
[29] P. Toint, *Some numerical results using a sparse matrix updating formula in unconstrained optimization*, Math. Comput., 32 (1978), pp. 839–852.
[30] P. Toint, *Test problems for partially separable optimization and results for the routine PSPMIN*, Tech. Report 83/4, Department of Mathematics, FUNDP, Namur, Belgium, 1983. Cited in [1, 2].