ON EFFICIENTLY COMPUTING THE EIGENVALUES OF LIMITED-MEMORY QUASI-NEWTON MATRICES

JENNIFER B. ERWAY AND ROUMMEL F. MARCIA

ABSTRACT. In this paper, we consider the problem of efficiently computing the eigenvalues of limited-memory quasi-Newton matrices that exhibit a compact formulation, e.g., BFGS, DFP and SR1 matrices. Further, we provide a compact formulation for the entire Broyden convex class of updates, making it possible to compute the eigenvalues of any limited-memory quasi-Newton matrix generated by these updates. The proposed method makes use of efficient updates to the QR factorization.

1. INTRODUCTION

Limited-memory quasi-Newton matrices are widely-used in both constrained and unconstrained optimization. In particular, they may be used as preconditioners for iterative methods or in place of exact Hessians when the Hessian is too computationally expensive to compute or otherwise unavailable. Some of the more conventional limited-memory quasi-Newton matrices are generated by BFGS, DFP, SR1, or the Broyden convex class of updates. In this paper we demonstrate how to efficiently compute the eigenvalues of these limited-memory quasi-Newton matrices.

Our proposed method relies on compact representations of these matrices. It is well-known that compact representations of BFGS, DFP, and SR1 matrices are available [5, 6, 8]; in particular, if \( B \) is generated using any of these updates with an initial Hessian approximation \( B_0 = \gamma I, \gamma \in \mathbb{R} \), then \( B \) can be written in the form

\[
B = \gamma I + \Psi M \Psi^T,
\]

where \( M \) is symmetric. With this compact representation in hand, the eigenvalue computation makes use of the QR factorization of \( \Psi \). This method was first proposed by Burdakov et al. [4]. The bulk of the computational effort involves computing the QR factorization of \( \Psi \).

This paper has two main contributions: (1) The compact representation of the Broyden convex class of updates, and (2) the efficient updating of the QR factorization used for the eigenvalue decomposition. We note that while compact representations of BFGS and DFP matrices are known [5, 6, 8], to our knowledge there has been no work done on generalizing the compact representation to the entire Broyden convex class of updates. This new compact
representation allows us to extend the eigenvalue computation to any member of the Broyden convex class of updates.

Prior work on explicitly computing eigenvalues of quasi-Newton matrices has been restricted to at most two updates. A theorem by Wilkinson [18, pp. 94–97] can be used to compute the eigenvalues of a quasi-Newton matrix after one rank-one update. For more than one rank-one update, Wilkinson’s theorem may only provide bounds on eigenvalues. The eigenvalues of rank-one modifications are considered by Golub [10] and several methods are proposed, including Newton’s method on the characteristic equation, linear interpolation on a related tridiagonal generalized eigenvalue problem, and finding zeros of the secular equation. Bunch et al. [3] extends the work of Golub to the case of eigenvalue algebraic multiplicity of more than one. Eigenvalue computations for more than one rank-one update are not proposed and, as is, these methods cannot be used to compute the eigenvalues for the general Broyden convex class of updates. Apostolopoulou et al. [2] and Apostolopoulou et al. [1] compute the eigenvalues of minimal-memory BFGS matrices, where the number of BFGS updates is limited to at most two. In these papers, formulas for the characteristic polynomials are derived that may be solved analytically. Due to the complexity involved in formulating characteristic polynomials and root finding, these approaches cannot be generalized to handle more than two updates.

This paper is organized in five sections. In Section 2, we outline the compact formulations for the BFGS, DFP, and SR1 matrices. In Section 3, we present the compact formulation for the Broyden convex class of updates. The method to compute the eigenvalues of any limited-memory quasi-Newton matrix with the compact formulation (1) is given in Section 4. An efficient method to update the QR factorization of $\Psi$ is also given in this section. Finally, in Section 5, there are some concluding remarks.

2. COMPACT FORMULATIONS OF QUASI-NEWTON MATRICES

In this section, we review compact formulations of some of the most widely-used quasi-Newton matrices; in particular, we consider the BFGS, DFP, and SR1 matrices. First, we introduce notation and assumptions used throughout this paper.

Given a continuously differentiable function $f(x) \in \mathbb{R}^{n \times 1}$ and iterates $\{x_k\}$, the quasi-Newton pairs $\{s_i, y_i\}$ are defined as follows:

\[
 s_i \triangleq x_{i+1} - x_i \quad \text{and} \quad y_i \triangleq \nabla f(x_{i+1}) - \nabla f(x_i),
\]

where $\nabla f$ denotes the gradient of $f$.

The goal of this section is to express a quasi-Newton matrix obtained from these updates in the form

\[
 B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T,
\]
where $\Psi_k \in \mathbb{R}^{n \times l}, M_k \in \mathbb{R}^{l \times l}$, and $B_0$ is a diagonal matrix (i.e., $B_0 = \gamma$, $\gamma \in \mathbb{R}$). We will obtain factorizations of the form \ref{e2} where $l = k + 1$ or $l = 2(k + 1)$; in either case, we assume $l \ll n$.

Throughout this section, we make use of the following matrices:

\begin{align*}
S_k &\triangleq \begin{bmatrix} s_0 & s_1 & s_2 & \cdots & s_k \end{bmatrix} \in \mathbb{R}^{n \times (k+1)}, \\
Y_k &\triangleq \begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_k \end{bmatrix} \in \mathbb{R}^{n \times (k+1)}.
\end{align*}

Furthermore, we make use of the following decomposition of $S_k^T Y_k \in \mathbb{R}^{(k+1) \times (k+1)}$:

$$S_k^T Y_k = L_k + D_k + R_k,$$

where $L_k$ is strictly lower triangular, $D_k$ is diagonal, and $R_k$ is strictly upper triangular. We assume all updates are well-defined; for example, for the BFGS and DFP updates, we assume that $s_i^T y_i > 0$ for $i = 0, 1, \ldots, k$.

2.1. The BFGS update. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) update is given by

$$B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k y_k^T B_k + \frac{1}{y_k^T s_k} y_k y_k^T.$$

Byrd et al. \cite[Theorem 2.3]{6} showed that $B_{k+1}$ can be written in the form

$$B_{k+1} = B_0 - \Psi_k \Gamma_k^{-1} \Psi_k^T,$$  \hspace{0.5cm} (3)

where

$$\Psi_k = (B_0 S_k \ Y_k) \quad \text{and} \quad \Gamma_k = \begin{pmatrix} S_k^T B_0 S_k & L_k \\ L_k & -D_k \end{pmatrix}. \hspace{0.5cm} (4)$$

Defining $M_k \triangleq -\Gamma_k^{-1}$ gives us the desired form \ref{e2} with $l = 2(k + 1)$.

2.2. The DFP update. The Davidon-Fletcher-Powell (DFP) update gives the following formula for $B_{k+1}$:

$$B_{k+1} = \left( I - \frac{y_k s_k^T}{y_k^T s_k} \right) B_k \left( I - \frac{s_k y_k^T}{y_k^T s_k} \right) + \frac{y_k y_k^T}{y_k^T s_k}.$$

Erway et al. \cite[Theorem 1]{8} showed that $B_{k+1}$ can be written in the form \ref{e2} where

$$\Psi_k = (B_0 S_k \ Y_k) \quad \text{and} \quad M_k = \begin{pmatrix} 0 & -\bar{L}_k^{-T} \\ -\bar{L}_k^{-1} & (D_k + S_k^T B_0 S_k) \bar{L}_k^{-T} \end{pmatrix},$$  \hspace{0.5cm} (5)

where $\bar{L}_k \triangleq L_k + D_k$. In this case, $l = 2(k + 1)$. We note that in \cite{8}, $\Psi_k M_k \Psi_k^T$ is expressed as the equivalent product

$$\Psi_k M_k \Psi_k^T = (Y_k \ B_0 S_k) \begin{pmatrix} \bar{L}_k^{-1} (D_k + S_k^T B_0 S_k) \bar{L}_k^{-T} & -\bar{L}_k^{-1} \\ -\bar{L}_k^{-T} & 0 \end{pmatrix} \begin{pmatrix} Y_k^T \\ (B_0 S_k)^T \end{pmatrix}.$$
2.3. The SR1 update. The symmetric rank-one (SR1) update formula is given by

\[ B_{k+1} = B_k + \frac{1}{s_k^T(y_k - B_k s_k)}(y_k - B_k s_k)(y_k - B_k s_k)^T. \]  

(6)

Byrd et al. [6, Theorem 5.1] showed that can be written in the form (2) where

\[ \Psi_k = Y_k - B_0 S_k \quad \text{and} \quad M_k = (D_k + L_k + L_k^T - S_k^T B_0 S_k)^{-1}. \]

Note that in the SR1 case, \( l = k + 1 \).

3. The Broyden convex class of updates

In this section, we present a compact formulation for the Broyden convex class of updates. The Broyden convex class of updates is given by

\[ B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k y_k^T + \frac{1}{y_k^T s_k} y_k^T + \phi(s_k^T B_k s_k)w_k w_k^T, \]

(7)

where \( \phi \in [0, 1] \) and

\[ w_k = \frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}, \]

(see, e.g., [15, 12]). Expanding (7) yields,

\[ B_{k+1} = B_k - \frac{1 - \phi}{s_k^T B_k s_k} B_k s_k y_k^T + \frac{\phi}{y_k^T s_k} B_k s_k y_k^T - \frac{\phi}{y_k^T s_k} y_k s_k y_k^T B_k + \left( 1 + \phi \frac{s_k^T B_k s_k}{y_k^T s_k} \right) \frac{1}{y_k^T s_k} y_k y_k^T, \]

and thus, \( B_{k+1} \) can be written compactly as

\[ B_{k+1} = B_k + (B_k s_k y_k) \left( \begin{array}{c} (1 - \phi) \\ \phi \end{array} \right) \left( \begin{array}{c} y_k^T s_k \\ y_k^T \end{array} \right) \left( \begin{array}{c} (B_k s_k)^T \\ y_k^T \end{array} \right). \]

(8)

Recall that our goal is to write \( B_{k+1} \) in the form

\[ B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T, \]

where \( \Psi_k \in \mathbb{R}^{n \times 2(k+1)} \) and \( M_k \in \mathbb{R}^{2(k+1) \times 2(k+1)} \). Letting \( \Psi_k \) be defined as

\[ \Psi_k \triangleq \begin{pmatrix} B_0 s_0 & \cdots & B_0 s_k & y_0 & y_1 & \cdots & y_k \end{pmatrix} = \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix}, \]

(9)

we now derive an expression for \( M_k \).
3.1. General $M_k$. In this section we state and prove a theorem that gives an expression for $M_k$. The eigenvalue computation in Section 5 requires the ability to form $M_k$. For this reason, we also provide a practical method for computing $M_k$ that makes use of recursion.

**Theorem 1.** Let $\Lambda_k \in \mathbb{R}^{(k+1) \times (k+1)}$ be a diagonal matrix such that
\[
\Lambda_k = \text{diag}(\lambda_i), \quad \text{where} \quad \lambda_i = \frac{1}{1 - \phi \frac{s_i^T B_i s_i}{s_i^T y_i}} \quad \text{for} \quad 0 \leq i \leq k. \tag{10}
\]
If $B_{k+1}$ is updated using the Broyden convex class of updates (7), where $\phi \in [0, 1]$, then $B_{k+1}$ can be written as
\[
B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T, \quad \text{where} \quad \Psi_k \text{ is defined as in (9)} \quad \text{and}
\]
\[
M_k = \begin{pmatrix}
-S^T_k B_0 S_k + \phi \Lambda_k & -L_k + \phi \Lambda_k \\
-L^T_k + \phi \Lambda_k & D_k + \phi \Lambda_k
\end{pmatrix}^{-1}.	ag{11}
\]

**Proof.** This proof is broken into two parts. First, we consider the special cases when $\phi = 0$ and $\phi = 1$. Then, we prove by induction the case when $\phi \in (0, 1)$.

When $\phi = 0$, (7) becomes the BFGS update and $M_k$ in (11) simplifies to
\[
M_k = \begin{pmatrix}
-S^T_k B_0 S_k - L_k \\
-L^T_k & D_k
\end{pmatrix}^{-1},
\]
which is consistent with (3) and (4). When $\phi = 1$, then $\Lambda_k = -D_k$ and so (7) is the DFP update and with
\[
M_k = \begin{pmatrix}
-S^T_k B_0 S_k - D_k & -\bar{L}_k \\
-\bar{L}^T_k & 0
\end{pmatrix}^{-1},
\]
where $\bar{L} = L_k + D_k$. After some algebra, it can be shown that this is exactly $M_k$ given in (5). Thus, $M_k$ in (11) is correct for $\phi = 0$ and $\phi = 1$.

The proof for $\phi \in (0, 1)$ is by induction on $k$. We begin by considering the base case $k = 0$. For $k = 0$, $B_1$ is given by (8), and thus, $B_1 = B_0 + \Psi_0 \tilde{M}_0 \Psi_0^T$ where $\Psi_0 = (B_0 s_0, y_0)$ and
\[
\tilde{M}_0 \triangleq \begin{pmatrix}
-\frac{(1 - \phi)}{s_0^T B_0 s_0} & -\frac{\phi}{y_0^T s_0} \\
-\frac{\phi}{y_0^T s_0} & \left(1 + \frac{s_0^T B_0 s_0}{y_0^T s_0}\right) \frac{1}{y_0^T s_0}
\end{pmatrix}.	ag{12}
\]
To complete the base case, we now show that $\tilde{M}_0$ in (12) is equivalent to $M_0$ in (11). For simplicity, $\tilde{M}_0$ can be written as
\[
\tilde{M}_0 = \begin{pmatrix}
\alpha_0 & \beta_0 \\
\beta_0 & \delta_0
\end{pmatrix},
\]
where
\[
\alpha_0 = -\frac{(1 - \phi)}{s_0^T B_0 s_0}, \quad \beta_0 = -\frac{\phi}{y_0^T s_0}, \quad \text{and} \quad \delta_0 = \left(1 + \frac{s_0^T B_0 s_0}{y_0^T s_0}\right) \frac{1}{y_0^T s_0}.	ag{14}
\]
We note that $\alpha_0$ and $\beta_0$ are nonzero since $0 < \phi < 1$. Consequently, $\delta_0$ can be written as

$$
\delta_0 = \left(1 + \phi \frac{s_0^T B_0 s_0}{y_0^T s_0}\right) \frac{1}{y_0^T s_0} = -\left(1 + (1 - \phi)\frac{\beta_0}{\alpha_0}\right) \frac{\beta_0}{\phi} = -\frac{\beta_0}{\phi} - \frac{\beta_0^2}{\phi \alpha_0} + \frac{\beta_0^2}{\alpha_0}. \quad (15)
$$

The determinant, $\eta_0$, of $\hat{M}_0$ can be written as

$$
\eta_0 = \alpha_0 \delta_0 - \beta_0^2 = -\frac{\alpha_0 \beta_0}{\phi} - \frac{\beta_0^2}{\phi} = -\frac{\beta_0}{\phi} (\alpha_0 + \beta_0). \quad (16)
$$

Since all members of the convex class are positive definite, both $\alpha_0$ and $\beta_0$ are negative, and thus, $\alpha_0 + \beta_0 \neq 0$ and $\eta_0 \neq 0$ in (16). It follows that $\hat{M}_0$ is invertible, and in particular,

$$
\hat{M}_0^{-1} = \begin{pmatrix} \delta_0/\eta_0 & -\beta_0/\eta_0 \\ -\beta_0/\eta_0 & \alpha_0/\eta_0 \end{pmatrix}.
$$

Together with (15), the (1,1) entry of $\hat{M}_0^{-1}$ simplifies to

$$
\frac{\delta_0}{\eta_0} = \frac{-\left(\frac{\alpha_0 + \beta_0}{\alpha_0} - \frac{\phi \beta_0}{\alpha_0}\right) \beta_0}{\phi} = \frac{1}{\alpha_0} - \frac{\phi \beta_0}{\alpha_0 (\alpha_0 + \beta_0)} = \frac{(\alpha_0 + \beta_0)(1 - \phi) + \phi \alpha_0}{\alpha_0 (\alpha_0 + \beta_0)} = \frac{1 - \phi}{\alpha_0} + \frac{\phi}{\alpha_0 + \beta_0}. \quad (17)
$$

Finally, the (2,2) entry of $\hat{M}_0^{-1}$ can be written as

$$
\frac{\alpha_0}{\eta_0} = -\frac{\phi \alpha_0}{\beta_0 (\alpha_0 + \beta_0)} = \frac{-\phi}{\beta_0} + \frac{\phi}{\alpha_0 + \beta_0}. \quad (18)
$$

Thus, combining (16), (17), and (18), we obtain the following equivalent expression for $\hat{M}_0^{-1}$:

$$
\hat{M}_0^{-1} = \begin{pmatrix} \frac{1 - \phi}{\alpha_0} + \frac{\phi}{\alpha_0 + \beta_0} & \frac{\phi}{\alpha_0 + \beta_0} \\ \frac{\phi}{\alpha_0 + \beta_0} & -\frac{\phi}{\beta_0} + \frac{\phi}{\alpha_0 + \beta_0} \end{pmatrix}. \quad (19)
$$
For the case $k = 0$, $L_0 = R_0 = 0$; moreover, $\lambda_0 = 1/(\alpha_0 + \beta_0)$. Substituting back in for $\alpha_0$, $\beta_0$ and $\delta_0$ using (14), we obtain

$$\vec{M}_0 = \left( s_0^T B_0 s_0 + \phi \lambda_0 \begin{array}{cc} \phi \lambda_0 \\ s_0^T y_0 + \phi \lambda_0 \end{array} \right)^{-1},$$

$$= \left( -S_k^T B_0 s_k + \phi \Lambda_k \begin{array}{cc} -L_k + \phi \Lambda_k \\ -L_k + \phi \Lambda_k \end{array} \right)^{-1},$$

$$= M_0,$$

proving the base case.

For the induction step, assume

$$B_m = B_0 + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T, \quad (20)$$

where $\Psi_{m-1}$ is defined as in (9) and

$$M_{m-1} = \left( \begin{array}{cc} -S_{m-1}^T B_0 s_{m-1} + \phi \Lambda_{m-1} & -L_{m-1} + \phi \Lambda_{m-1} \\ -L_{m-1} + \phi \Lambda_{m-1} & D_{m-1} + \phi \Lambda_{m-1} \end{array} \right)^{-1}. \quad (21)$$

From (8), we have

$$B_{m+1} = B_0 + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T + (B_m s_m \ y_m) \begin{pmatrix} \alpha_m & \beta_m \\ \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} (B_m s_m)^T \\ y_m^T \end{pmatrix}, \quad (22)$$

where

$$\alpha_m = -\frac{1 - \phi}{s_m^T B_m s_m}, \quad \beta_m = -\frac{\phi}{y_m^T s_m},$$

and

$$\delta_m = \left( 1 + \phi \frac{s_m^T B_m s_m}{y_m^T s_m} \right) \frac{1}{y_m^T s_m} = - \left( 1 + (1 - \phi) \frac{\beta_m}{\alpha_m} \right) \frac{\beta_m}{\phi}.$$

As in the base case, $k = 0$, we note that $\alpha_m$ and $\beta_m$ are nonzero since $0 < \phi < 1$, and that the determinant $\alpha_m \delta_m - \beta_m^2$ is also nonzero.

Multiplying (20) by $s_m$ on the right, we obtain

$$B_m s_m = B_0 s_m + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T s_m. \quad (23)$$

Then, substituting this into (22) yields

$$B_{m+1} = B_0 + \Psi_{m-1} M_{m-1} \Psi_{m-1}^T + (B_0 s_m + \Psi_{m-1} p_m \ y_m) \begin{pmatrix} \alpha_m & \beta_m \\ \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} (B_0 s_m + \Psi_{m-1} p_m)^T \\ y_m^T \end{pmatrix}, \quad (24)$$

where $p_m \triangleq M_{m-1} \Psi_{m-1}^T s_m$. Equivalently,

$$B_{m+1} = B_0 + (\Psi_{m-1} B_0 s_m \ y_m) \begin{pmatrix} M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\ \alpha_m p_m & \alpha_m & \beta_m \\ \beta_m p_m & \beta_m & \delta_m \end{pmatrix} \begin{pmatrix} \Psi_{m-1}^T (B_0 s_m)^T \\ y_m^T \end{pmatrix}.$$
The $3 \times 3$ block matrix in (25) has the following decomposition:

$$
\begin{pmatrix}
I & p_m & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
I & -p_m & 0 \\
p_m & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
M_{m-1} & 0 & 0 \\
0 & \alpha_m & \beta_m \\
0 & \beta_m & \delta_m \\
\end{pmatrix}
\begin{pmatrix}
I & -p_m & 0 \\
p_m & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
$$

allowing us to compute its inverse as follows:

$$
\begin{pmatrix}
M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\
\alpha_m p_m^T & \alpha_m & \beta_m \\
\beta_m p_m^T & \beta_m & \delta_m \\
\end{pmatrix}
^{-1} =
\begin{pmatrix}
I & 0 & 0 \\
-\alpha_m & 0 & -\beta_m \\
\beta_m & 0 & \delta_m \\
\end{pmatrix}
\begin{pmatrix}
M_{m-1} & 0 & 0 \\
0 & \alpha_m & \beta_m \\
0 & \beta_m & \delta_m \\
\end{pmatrix}
\begin{pmatrix}
I & -p_m & 0 \\
p_m & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
$$

where

$$
\begin{align*}
\tilde{\alpha}_m &= \frac{\delta_m}{\alpha_m \delta_m - \beta_m^2} = \frac{1 - \phi}{\alpha_m} + \frac{\phi}{\alpha_m + \beta_m} \\
\tilde{\beta}_m &= \frac{1}{\alpha_m \delta_m - \beta_m^2} = \frac{\phi}{\alpha_m + \beta_m} \\
\tilde{\delta}_m &= \frac{1}{\alpha_m \delta_m - \beta_m^2} = \frac{-\phi}{\alpha_0} + \frac{\phi}{\alpha_0 + \beta_0}.
\end{align*}
$$

We now simplify the entries of (27). Since $p_m = M_{m-1} \Psi_{m-1} s_m$, then $M_{m-1}^{-1} p_m = \Psi_{m-1}^T s_m$, giving us an expression for the $(1,2)$ and $(2,1)$ entries. The $(2,2)$ block entry is simplified by first multiplying (23) by $s_m^T$ on the left to obtain $s_m^T B_m s_m = s_m^T B_0 s_m + s_m^T B_m s_m + \tilde{\alpha}_m$. Then,

$$
p_m^T M_{m-1}^{-1} p_m + \tilde{\alpha}_m = -s_m^T B_0 s_m + s_m^T B_m s_m + \tilde{\alpha}_m
$$

Thus, (27) can be written as

$$
\begin{pmatrix}
M_{m-1} & -\Psi_{m-1}^T \Psi_{m-1} s_m & 0 \\
-\Psi_{m-1}^T \Psi_{m-1} s_m & \Phi & 0 \\
0 & 0 & \Psi_{m-1}^T \Psi_{m-1} s_m + \Phi
\end{pmatrix}
$$

(28)
We now show \((11)\) holds using \((28)\). Define the permutation matrix \(\Pi_m \in \mathbb{R}^{2(m+1) \times 2(m+1)}\)
as follows:

\[
\Pi_m = \begin{pmatrix}
I_m & 0 \\
0 & I_m \\
1 & 0 \\
\end{pmatrix}.
\]

Then,

\[
[\Psi_{m-1} \ B_0 s_m \ y_m] \Pi_m = \Psi_m;
\]
in other words, \([\Psi_{m-1} \ B_0 s_m \ y_m] = \Psi_m \Pi_m^T\). Therefore, \((25)\) can be written as

\[
B_{m+1} = B_0 + \Psi_m \Pi_m^T \hat{M}_m \Pi_m \Psi_m^T,
\]

where

\[
\hat{M}_m = \begin{pmatrix}
M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\
\alpha_m p_m^T & \alpha_m & \beta_m \\
\beta_m p_m^T & \beta_m & \delta_m
\end{pmatrix}.
\]

It remains to show \((11)\) holds for \(k = m\) with \(M_m = \Pi_m^T \hat{M}_m \Pi_m\).

We now consider \(M_m^{-1}\) given by

\[
M_m^{-1} = \left(\Pi_m^T \begin{pmatrix}
M_{m-1} + \alpha_m p_m p_m^T & \alpha_m p_m & \beta_m p_m \\
\alpha_m p_m^T & \alpha_m & \beta_m \\
\beta_m p_m^T & \beta_m & \delta_m
\end{pmatrix} \Pi_m\right)^{-1}, \tag{29}
\]

which can be simplified using \((28)\):

\[
M_m^{-1} = \Pi_m^T \begin{pmatrix}
M_{m-1}^{-1} & -\Psi_{m-1} s_m & 0 \\
-\Psi_{m-1} s_m & \frac{-\Psi_{m-1} s_m + \phi}{\alpha_m + \beta_m} & \frac{\phi}{\alpha_m + \beta_m} \\
0 & \frac{\phi}{\alpha_m + \beta_m} & \frac{y_m s_m + \phi}{\alpha_m + \beta_m}
\end{pmatrix} \Pi_m. \tag{30}
\]

Now partition \(M_m^{-1}\) as follows:

\[
M_m^{-1} = \begin{pmatrix}
(M_{m-1}^{-1})_{11} & (M_{m-1}^{-1})_{12} \\
(M_{m-1}^{-1})_{21} & (M_{m-1}^{-1})_{22}
\end{pmatrix}.
\]

Applying the permutation matrices together with \(\Psi_{m-1}^T s_m = (S_{m-1}^T B_0 s_m, \ Y_{m-1}^T s_m)\), we have that

\[
M_m^{-1} = \begin{pmatrix}
(M_{m-1}^{-1})_{11} & -S_{m-1}^T B_0 s_m & 0 \\
-S_{m-1}^T B_0 s_m & \frac{-S_{m-1}^T B_0 s_m + \phi}{\alpha_m + \beta_m} & \frac{\phi}{\alpha_m + \beta_m} \\
0 & \frac{\phi}{\alpha_m + \beta_m} & \frac{y_m s_m + \phi}{\alpha_m + \beta_m}
\end{pmatrix}.
\]
Simplifying using the induction hypothesis \[21\] yields
\[
M_{m}^{-1} = \begin{pmatrix}
-S_{m-1}^TB_0S_{m-1} + \phi \Lambda_{m-1} & -S_{m-1}^TB_0s_m & -L_{m-1} + \phi \Lambda_{m-1} & 0 \\
-s_m^TB_0S_{m-1} & -s_m^TB_0s_m + \phi & -s_m^TY_m & \phi \\
-\phi & \alpha_m + \beta_m & 0 & \alpha_m + \beta_m \\
0 & -Y_m^T s_m & \phi & y_m^T s_m + \phi \\
\end{pmatrix}
\]
\[
= \begin{pmatrix}
-S_{m}^TB_0S_m + \phi \Lambda_m & -L_m + \phi \Lambda_m \\
-\phi & \alpha_m + \beta_m \\
-D_m + \phi \Lambda_m & \phi \\
\end{pmatrix},
\]
i.e., \[11\] holds for \(k = m\). □

Although we have found an expression for \(M_k\), computing \(M_k\) is not straightforward. In particular, the diagonal matrix \(\Lambda_k\) in Eq. \[10\] involves \(s_i^TB_is_i\), which requires \(B_i\) for \(0 \leq i \leq k\). In the following section we propose a different way of computing \(M_k\) that does not necessitate storing the quasi-Newton matrices \(B_i\) for \(0 \leq i \leq k\).

3.2. Computing \(M_k\). In this section we propose a recursive method for computing \(M_k\) from \(M_{k-1}\). From \[29\],
\[
M_k = \Pi_k^T \begin{pmatrix}
M_{k-1} + \alpha_k p_k p_k^T & \alpha_k p_k & \beta_k p_k \\
\alpha_k p_k^T & \alpha_k & \beta_k \\
\beta_k p_k^T & \beta_k & \delta_k \\
\end{pmatrix} \Pi_k.
\]
The vector \(p_k\) can be computed as follows:
\[
p_k = M_{k-1} \Psi_{k-1}^s s_k = (M_{k-1} \begin{pmatrix}
(B_0S_{k-1})^T \\
Y_{k-1}^T \\
\end{pmatrix}) s_k = M_{k-1} \begin{pmatrix}
S_{k-1}^TB_0s_k \\
Y_{k-1}^s \\
\end{pmatrix}.
\]
Note that \((S_{k-1}^TB_0s_k)^T\) is the last row (save the diagonal entry) of \(S_{k-1}^TB_0S_k\) and \((Y_{k-1}^s)^T\) is the last row (save the diagonal entry) of \(S_{k-1}^TY_k\). The entry \(\alpha_k\), which is given by \(\alpha_k = -(1 - \phi)/s_k^TB_k s_k\) can be computed from the following:
\[
s_k^T B_k s_k = s_k^T (B_0 + \Psi_{k-1} M_{k-1} \Psi_{k-1}^T) s_k = s_k^T B_0 s_k + s_k^T \Psi_{k-1} p_k.
\]
The quantity \(s_k^T B_0 s_k\) is the last diagonal entry in \(S_k^TB_0S_k\), and \(s_k^T \Psi_{k-1} p_k\) is the inner product of \(\Psi_{k-1}^s s_k\) (which was formed when computing \(p_k\)) and \(p_k\). The entry \(\beta_k\) is given by \(\beta_k = -\phi/y_k^T s_k\), where \(y_k^T s_k\) is the last diagonal entry in \(S_k^TY_k\). Finally, \(\delta_k = (1 + \phi s_k^T B_k s_k/y_k^T s_k)/y_k^T s_k\), which uses the previously computed quantities \(s_k^T B_k s_k\) and \(y_k^T s_k\).

We summarize this recursive method in the algorithm below:

Algortihm 1. This algorithm computes \(M_k\) in \[11\].

Define \(\phi\) and \(B_0\);
Define $M_0$ using (12):

for $j = 1 : k$

\[
p_j \leftarrow M_{j-1}(\Psi_{j-1}s_j);
\]

\[
s_j^T B_j s_j \leftarrow s_j^T B_0 s_j + (s_j^T \Psi_{j-1}) p_j;
\]

\[
\alpha_j \leftarrow -(1 - \phi)/(s_j^T B_j s_j);
\]

\[
\beta_j \leftarrow -\phi/(y_j^T s_j);
\]

\[
\delta_j \leftarrow (1 + \phi(s_j^T B_j s_j)/(y_j^T s_j))/(y_j^T s_j);
\]

\[
M_j \leftarrow \Pi_j^T \begin{pmatrix} M_{j-1} + \alpha_j p_j p_j^T & \alpha_j p_j & \beta_j p_j \\ \alpha_j p_j & \alpha_j & \beta_j \\ \beta_j p_j & \beta_j & \delta_j \end{pmatrix} \Pi_j
\]

end

The matrices \{\Pi_j\} are not formed explicitly since they are permutation matrices; thus, no matrix-matrix products are required by the recursion algorithm.

4. Computing the eigenvalues of $B_{k+1}$

In this section, we demonstrate how to compute the eigenvalues of a limited-memory matrix $B_{k+1}$ when the following decomposition is available:

\[
B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T,
\]

where $B_0 = \gamma I$, $\gamma \in \mathbb{R}$. We assume that $B_{k+1} \in \mathbb{R}^{n \times n}$ but only $m$ limited-memory updates are stored, where $m \ll n$ (see, e.g., [13, 14, 16, 17]). In large-scale optimization, typically $m < 10$ (e.g., Byrd et al. [6] recommend $m \in [2, 6]$).

4.1. Eigenvalues via the QR decomposition. We begin by finding the eigenvalues of $B_{k+1}$ when $B_{k+1}$ is obtained using the Broyden convex class of updates; at the end of this section, we describe the modifications needed to find the eigenvalues for the SR1 case. We assume $k + 1 \leq m$.

For the Broyden convex class of updates, $\Psi_k = (B_0 S \ Y)$, i.e.,

\[
\Psi_k = \begin{pmatrix} B_0 s_0 & B_0 s_1 & \cdots & B_0 s_k & y_0 & y_1 & \cdots & y_k \end{pmatrix}.
\]

To facilitate updating $\Psi_k$ after computing a new limited-memory pair (see Section 4.2), we permute the columns of $\Psi_k$ using a permutation matrix $P$ so that

\[
\hat{\Psi}_k \triangleq \Psi_k P = \begin{pmatrix} B_0 s_0 & y_0 & B_0 s_1 & y_1 & \cdots & B_0 s_k & y_k \end{pmatrix}.
\]

Let

\[
\hat{\Psi}_k = QR \in \mathbb{R}^{n \times l}
\]

be the QR decomposition of $\hat{\Psi}_k$, where $Q \in \mathbb{R}^{n \times n}$ has orthonormal columns and $R \in \mathbb{R}^{n \times l}$ is upper triangular (see, e.g., [11]).
Then,
\[
B_{k+1} = B_0 + \Psi_k M_k \Psi_k^T \\
= B_0 + \hat{\Psi}_k P^T M_k P \hat{\Psi}_k^T \\
= B_0 + Q R P^T M_k P R^T Q^T
\]

The matrix \( R P^T M_k P R^T \) is a real symmetric \( n \times n \) matrix. Since \( \hat{\Psi}_k \in \mathbb{R}^{n \times l} \), \( R \) has at most rank \( l \); moreover, \( R \) can be written in the form
\[
R = \begin{pmatrix} R_1 & 0 \\ 0 & I \end{pmatrix}
\]
where \( R_1 \in \mathbb{R}^{l \times l} \). Then,
\[
R P^T M_k P R^T = \begin{pmatrix} R_1 & 0 \\ 0 & I \end{pmatrix} P^T M_k P \begin{pmatrix} R_1^T & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} R_1 P^T M_k P R_1^T & 0 \\ 0 & 0 \end{pmatrix}.
\]
The eigenvalues of \( R P^T M_k P R^T \) can be explicitly computed by forming the spectral decomposition of \( R_1 P^T M_k P R_1^T \in \mathbb{R}^{l \times l} \). That is, suppose \( V_1 D_1 V_1^T \) is the spectral decomposition of \( R_1 P^T M_k P R_1^T \). Then,
\[
R P^T M_k P R^T = \begin{pmatrix} R_1 P^T M_k P R_1^T & 0 \\ 0 & I \end{pmatrix} = V D V^T
\]
where
\[
V \triangleq \begin{pmatrix} V_1 & 0 \\ 0 & I \end{pmatrix} \in \mathbb{R}^{n \times n} \quad \text{and} \quad D \triangleq \begin{pmatrix} D_1 & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n}.
\]
This gives that
\[
B_{k+1} = B_0 + Q V D V^T Q^T \\
= \gamma I + Q V D V^T Q^T \\
= Q V (\gamma I + D) V^T Q^T,
\]
yielding the spectral decomposition of \( B_{k+1} \). The matrix \( B_{k+1} \) has an eigenvalue of \( \gamma \) with multiplicity \( n - l \) and \( l \) eigenvalues given by \( \gamma + d_i \), where \( 1 \leq i \leq l \). In practice, the matrices \( Q \) and \( V \) in \((32)\) are not stored.

In the case of the SR1 updates, \( \Psi_k = Y_k - B_0 S_k \) and no permutation matrix is required (i.e., \( P = I \)).

Computing the eigenvalues of \( B_{k+1} \) directly is an \( O(n^3) \) process. In contrast, the above decomposition requires the QR factorization of \( \Psi_k \) and the eigendecomposition of \( R_1 P^T M_k P R_1^T \), requiring \( (O(n^2) \) flops) and \( O(l^3) \) flops, respectively. Since \( l \ll n \), the proposed method results in a substantial computational savings.
4.2. **Handling updates to $\hat{\Psi}$**. In this section we detail handling updates to the QR decomposition of $\hat{\Psi}_k$ when additional limited-memory pairs are added to $S$ and $Y$. We consider two cases: Adding a limited-memory pair $(s_k, y_k)$ when $k < m$ and when $k \geq m$, where $m$ is the maximum number of limited-memory updates allowed to be stored. The case $k < m$ requires adding a row and column to the $R$ factor; whereas the case $k \geq m$ requires first deleting a column (or two) of $\hat{\Psi}_k$ before adding the newest limited-memory pair. In both cases, the columns of $Q$ need not be formed nor stored. However, when $\hat{\Psi}_k$ is not full rank, the QR decomposition must be computed from scratch.

We begin by discussing the process to compute $\hat{\Psi}_{k+1}$ from $\hat{\Psi}_k$ when a new limited-memory pair is added to $S$ and $Y$. The discussion considers the Broyden convex class of updates; however, comments are included at the end of each subsection regarding the SR1 case.

4.2.1. **Adding a column to $S$ and $Y$**. Suppose $\hat{\Psi}_k = QR \in \mathbb{R}^{n \times l}$ is full rank and we have stored $k + 1$ limited-memory Broyden convex class updates such that $k + 1 < m$, where $m$ is the maximum number of limited-memory updates allowed to be stored by the limited-memory quasi-Newton method. Further, suppose we have computed a $(k + 2)$nd pair $(s_{k+1}, y_{k+1})$. To update the QR decomposition, we augment $\hat{\Psi}_k$ with the two columns $(B_0 s_{k+1}, y_{k+1})$. This can be accomplished by using the procedure proposed by Gill et al. [9] for updating the QR factorization after a column is added. This method relies upon $\hat{\Psi}_k$ having full column rank. For completeness, this procedure is presented below in the context of adding two columns to $\hat{\Psi}_k$. As in the previous section, we assume that $B_k$ is updated using the Broyden convex set of updates.

We begin by first adding the column $B_0 s_{k+1}$ to $\hat{\Psi}_k$; the same process may be followed to add the new last column, $y_{k+1}$. Suppose

$$
\hat{\Psi}_k = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix},
$$

where $R \in \mathbb{R}^{n \times l}$. Moreover, suppose we insert $B_0 s_{k+1}$ into the final column of $\hat{\Psi}_k$ to obtain $\hat{\Psi}_k$. Setting

$$
\hat{\Psi}_k = Q \begin{pmatrix} R_1 \\ 0 \\ u_1 \\ u_2 \end{pmatrix}
$$

yields that

$$
B_0 s_{k+1} = Qu \quad \text{with} \quad u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},
$$

where $u_1 \in \mathbb{R}^l$ and $u_2 \in \mathbb{R}^{n-l}$. We now construct an orthogonal matrix $H_1$ such that

$$
H_1 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ \eta \\ 0 \end{pmatrix},
$$
where \( \eta = \pm \| u_2 \| \). Choosing \( H_1 \) to be a Householder matrix preserves the structure of \( R_1 \), i.e.,

\[
H_1 \begin{pmatrix} R_1 & u_1 \\ 0 & u_2 \end{pmatrix} = \begin{pmatrix} R_1 & u_1 \\ 0 & \eta \\ 0 & 0 \end{pmatrix}.
\]

Then, \( \hat{\Psi}_k = Q \hat{R} \) is the QR decomposition of \( \hat{\Psi}_k \), where

\[
\hat{R} = H_1^T \begin{pmatrix} R_1 & u_1 \\ 0 & \eta \\ 0 & 0 \end{pmatrix}.
\]

In this procedure, the matrices \( Q \), \( \hat{Q} \), and \( H_1 \) are not stored; moreover, the unknowns \( u_1 \) and \( \eta \) can be computed without explicitly using these matrices. Specifically, the relation in (34) implies \( \hat{\Psi}_k^T B_0 s_{k+1} = (R_1^T 0) Q^T Qu \), i.e.,

\[
\hat{\Psi}_k^T B_0 s_{k+1} = R_1 u_1.
\] (36)

Equation (36) is a square \( l \times l \) system that can be solved for \( u_1 \) provided \( \hat{\Psi}_k \) is full rank. Finally, the scalar \( \eta \) can be computed from the following relation obtained from combining equations (34) and (35): \( \| B_0 s_{k+1} \|^2 = \| (u_1 \quad \eta) \|^2 \). This yields that \( \eta^2 = \| B_0 s_{k+1} \|^2 - \| u_1 \|^2 \).

This procedure can be repeated to add \( y_{k+1} \) to the new last column of \( \hat{\Psi}_k \), thereby updating the QR factorization of \( \hat{\Psi}_k \) to \( \hat{\Psi}_{k+1} \) with a new pair of updates \( (B_0 s_{k+1}, y_{k+1}) \).

The process of adding a new SR1 update to \( \hat{\Psi}_k \) is simpler since \( \hat{\Psi}_k \) is augmented by only one column: \( y_k - B_0 s_k \).

4.2.2. The full-rank assumption. The process described above requires \( \hat{\Psi}_k \) to be full rank so that there is a (unique) solution to (36). When \( \hat{\Psi}_k \) is not full rank, the QR decomposition must be computed from scratch. Fortunately, there is an \textit{a priori} way to determine when there is no unique solution: The matrix \( \hat{\Psi}_k \) has full rank if and only if \( R_1 \) in (36) is invertible; in particular, the diagonal of \( R_1 \) is nonzero. When \( R_1 \) is singular, the process described above to update the QR decomposition for \( \hat{\Psi}_{k+1} \) is skipped and the QR decomposition of \( \hat{\Psi}_{k+1} \) should be computed from scratch at a cost of \( 2l^2(n - l/3) \) flops. The process described in Section 4.2.1 can be reinstated to update the QR decomposition when the \( R_1 \) factor has nonzero diagonal entries, which may occur again once the limited-memory updates exceed the maximum number allowed, (i.e., \( k \geq m \)), and we are forced to delete the oldest pairs.

Similarly, when \( \hat{\Psi}_k \) is ill-conditioned, \( R_1 \) will also be ill-conditioned with at least one relatively small diagonal entry. In this case, (36) should not be solved; instead, the QR factorization should be computed from scratch. As with the rank-deficient case, it is possible to know this \textit{a priori}.
4.2.3. Deleting and adding columns to $S$ and $Y$. In this section, we detail the process to update the QR factorization in an efficient manner when $\hat{\Psi}_k$ is full rank and $k \geq m$. As in the previous section, we assume we are working with the Broyden convex class of updates.

Suppose $\hat{\Psi}_k = QR$ and we have stored the maximum number $(k + 1)$ limited-memory pairs $\{(s_i, y_i)\}$, $i = 0, \ldots, k$ allowed by the limited-memory quasi-Newton method. Further, suppose we have computed a $(k + 2)$nd pair $(s_{k+1}, y_{k+1})$. The process to obtain an updated QR factorization of $\hat{\Psi}_{k+1}$ from $\hat{\Psi}_k$ can be viewed as a two step process:

1. Delete a column of $S$ and $Y$.
2. Add a new column to $S$ and $Y$.

For the first step, we use ideas based on Daniel et al. \[7\] and Gill et al. \[9\]. Consider the Broyden class of updates. Suppose we rewrite $\hat{\Psi}_k$ and $R$ as

$$
\hat{\Psi}_k = (B_0 s_0 \ y_0 \ \hat{\Psi}_k) \quad \text{and} \quad R = (r_1 \ r_2 \ \bar{R}),
$$

where $\hat{\Psi}_k \in \mathbb{R}^{n \times (l-2)}$ and $\bar{R} \in \mathbb{R}^{n \times (l-2)}$. This gives that

$$
\hat{\Psi}_k = (B_0 s_0 \ y_0 \ \hat{\Psi}_k) = Q (r_1 \ r_2 \ \bar{R}).
$$

Deleting the first two columns of $\hat{\Psi}_k$ yields the matrix $\bar{\Psi}_k = \bar{Q} \bar{R}$. Notice that $\bar{R}$ has zeros beneath the second subdiagonal. For clarity, we illustrate the nonzero entries of $\bar{R}$ for the case $n = 8$ and $k = 3$:

$$
\begin{pmatrix}
* & * & * & * \\
* & * & * & * \\
0 & * & * & * \\
0 & 0 & * & * \\
0 & 0 & 0 & * \\
0 & 0 & 0 & 0 \\
\end{pmatrix},
$$

where * denotes possible nonzero entries. Givens rotations can be used to zero out the entries beneath the main diagonal in $\bar{R}$ at a cost of $24k^2 - 36k$. (In the above example, eight entries must be zeroed out to reduce (38) to upper triangular form; more generally, $4(k - 1)$ entries must zeroed out to reduce $\bar{R}$ to upper triangular form.) Let $G_{i,j} \in \mathbb{R}^{n \times n}$ denote the Givens matrix that zeros out the $(i, j)$th component of $\bar{R}$, and suppose $\hat{G}$ is given by

$$
\hat{G} \triangleq G_{2j-1,2j-2} G_{2j,2j-2} \cdots G_{2,1} G_{3,1}.
$$

Then, $\hat{G} \bar{R}$ is an upper triangular matrix. This yields the QR decomposition of the matrix $\bar{\Psi}_k$ defined as follows:

$$
\bar{\Psi}_k \triangleq \hat{Q} \bar{R},
$$

where $\hat{Q} = Q \hat{G}^T \in \mathbb{R}^{n \times n}$ is orthogonal and $\bar{R} \in \mathbb{R}^{n \times (l-2)}$ is an upper triangular matrix. With the computation of $\bar{\Psi}_k$ we have completed the first step. Notice that neither $Q$ nor $\hat{Q}$ must be stored in order to obtain $\bar{R}$. 
For the second step, the QR factorization of $\hat{\Psi}_{k+1}$ can be obtained from $\tilde{\Psi}_k$ using the procedure outlined in Section 4.2.1.

The process required for SR1 updates is simpler than for the Broyden convex class of updates since it is only a rank-one update. That is, only one column of $\hat{\Psi}_k$ must be deleted to remove the old pair of updates and only one column must be added to incorporate the newest pair of updates.

5. Concluding remarks

In this paper we produced the compact formulation of quasi-Newton matrices generated by the Broyden convex class of updates. Together with the QR factorization, this compact representation was used to compute the eigenvalues of any member of this class of updates. In addition, we presented an efficient procedure to update the QR factorization when a new pair of updates for the quasi-Newton matrix is computed. With this approach we are able to substantially reduce the computational costs of computing the eigenvalues of quasi-Newton matrices.

REFERENCES

[1] M. S. Apostolopoulou, D. G. Sotiropoulos, C. A. Botsaris, and Panayiotis E. Pintelas, A practical method for solving large-scale TRS, Optimization Letters, 5 (2011), pp. 207–227.
[2] M. S. Apostolopoulou, D. G. Sotiropoulos, and P. Pintelas, Solving the quadratic trust-region subproblem in a low-memory BFGS framework, Optimization Methods Software, 23 (2008), pp. 651–674.
[3] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, Rank-one modification of the symmetric eigenproblem, Numerische Mathematik, 31 (1978), pp. 31–48.
[4] O. Burdakov, L. Gong, Y.-X. Yuan, and S. Zikrin, On efficiently combining limited memory and trust-region techniques, Tech. Report 2013:13, Linkping University, Optimization, 2013.
[5] J. V. Burke, A. Wiegmann, and L. Xu, Limited memory BFGS updating in a trust-region framework, technical report, University of Washington, 1996.
[6] R. H. Byrd, J. Nocedal, and R. B. Schnabel, Representations of quasi-Newton matrices and their use in limited-memory methods, Math. Program., 63 (1994), pp. 129–156.
[7] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, Math. Comput., 30 (1976), pp. 772–795.
[8] J. B. Erway, V. Jain, and R. F. Marcia, Shifted limited-memory DFP systems, in Signals, Systems and Computers, 2013 Asilomar Conference on, Nov 2013, pp. 1033–1037.
[9] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, Methods for modifying matrix factorizations, Math. Comput., 28 (1974), pp. 505–535.
[10] G. H. Golub, Some modified matrix eigenvalue problems, SIAM Rev., 15 (1973), pp. 318–334.
[11] G. H. Golub and C. F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, Maryland, third ed., 1996.
[12] I. Griva, S. G. Nash, and A. Sofer, Linear and nonlinear programming, Society for Industrial and Applied Mathematics, Philadelphia, 2009.
[13] T.G. Kolda, D.P. O’leary, and L. Nazareth, BFGS with update skipping and varying memory, SIAM Journal on Optimization, 8 (1998), pp. 1060–1083.
[14] D.C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical programming, 45 (1989), pp. 503–528.
EIGENVALUES OF LIMITED-MEMORY QUASI-NEWTON MATRICES

[15] D.G. Luenberger and Y. Ye, Linear and nonlinear programming, vol. 116, Springer, 2008.

[16] 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 Journal on Optimization, 1 (1991), pp. 358–372.

[17] J. Nocedal, Updating quasi-newton matrices with limited storage, Mathematics of computation, 35 (1980), pp. 773–782.

[18] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

E-mail address: erwayjb@wfu.edu

Department of Mathematics, Wake Forest University, Winston-Salem, NC 27109

E-mail address: rmancia@ucmerced.edu

Applied Mathematics, University of California, Merced, Merced, CA 95343