SIAM J. MATRIX ANAL. APPL.
Vol. 31, No. 5, pp. 2700–2720

OPTIMIZING HALLEY'S ITERATION FOR COMPUTING THE MATRIX POLAR DECOMPOSITION*

YUJI NAKATSUKASA†, ZHAOJUN BAI‡, AND FRANÇOIS GYGI§

Abstract. We introduce a dynamically weighted Halley (DWH) iteration for computing the polar decomposition of a matrix, and we prove that the new method is globally and asymptotically cubically convergent. For matrices with condition number no greater than 10^{16}, the DWH method needs at most six iterations for convergence with the tolerance 10^{-16}. The Halley iteration can be implemented via QR decompositions without explicit matrix inversions. Therefore, it is an inverse free communication friendly algorithm for the emerging multicore and hybrid high performance computing systems.

Key words. polar decomposition, Halley's iteration, Newton's iteration, inverse free iterations, QR decomposition, numerical stability

AMS subject classifications. 15A15, 15A23, 65F30

DOI. 10.1137/090774999

1. Introduction.

We consider the computation of the unitary polar factor U of the polar decomposition of $A \in \mathbb{C}^{m\times n}$ $(m \geq n)$,

$$A = UH, \tag{1.1}$$

where $U \in \mathbb{C}^{m\times n}$ is a unitary matrix $U^H U = I$ and $H \in \mathbb{C}^{n\times n}$ is a unique Hermitian positive semidefinite matrix. The unitary polar factor U is unique if A has full column rank [20, p. 193]. Applications of the polar decomposition include factor analysis, satellite tracking, and calculation of the nearest orthogonal matrix [18]. Our motivation is from solving a large scale orthogonal Procrustes problem arising from the subspace alignment in the first-principles molecular dynamics simulations of electronic structure calculations [2, 9, 13, 14].

The most popular method for computing the polar factor of a square nonsingular matrix is the scaled Newton (SN) method [20, p. 202]. Recently, Byers and Xu [4] presented a suboptimal scaling strategy for the Newton method. They showed that the convergence to within a tolerance of 10^{-16} can be reached in at most nine iterations for matrices with condition number no greater than 10^{16}. Furthermore, they claim that Newton's method with suboptimal scaling is backward stable, provided that the matrix inverses are computed in a mixed forward-backward stable way. We note that there is a recent note [24] to indicate some incompleteness of rounding error analysis presented in [4].

Successful as Newton's method is, it requires explicit matrix inversion at each iteration. Besides the potential numerical stability issue in finite precision arithmetic,

*Received by the editors October 26, 2009; accepted for publication (in revised form) July 26, 2010; published electronically September 29, 2010. This work was partially supported by NSF grant OCI-0749217 and DOE grant DE-FC02-06ER25794.

http://www.siam.org/journals/simax/31-5/77499.html

†Department of Mathematics, University of California, Davis, CA 95616 (ynakatsukasa@ucdavis.edu).

‡Department of Computer Science and Department of Mathematics, University of California, Davis, CA 95616 (bai@cs.ucdavis.edu).

§Department of Applied Science and Department of Computer Science, University of California, Davis, CA 95616 (fgygi@ucdavis.edu).

explicit matrix inversion is also expensive in communication costs. On the emerging multicore and heterogeneous computing systems, communication costs have exceeded arithmetic costs by orders of magnitude, and the gap is growing exponentially over time [3, 12, 27]. The purpose of this paper is to investigate numerical methods for computing the polar decomposition to minimize the communication costs by using communication friendly matrix operations such as the QR decomposition (without pivoting) [8].

In fact, inverse free methods for computing the polar decomposition have been studied in [7, 5]. A QR decomposition-based implementation of a variant of the SN method is investigated. Unfortunately, the numerical instability of such an inverse free method has been independently discovered by both studies.

In this paper, we first propose a dynamically weighted Halley (DWH) method for computing the polar decomposition. We prove that the DWH method converges globally with asymptotically cubic rate. We show that in exact arithmetic, for matrices with condition number $\kappa_2(A) \leq 10^{16}$, no more than six iterations are needed for convergence with the tolerance 10^{-16}. We then discuss an implementation of the DWH method based on the QR decomposition. Extensive numerical tests indicate that the QR-based DWH (QDWH) method is backward stable. The arithmetic cost of the QDWH method is about two to three times that of the SN method, depending on the specific implementation one uses. However, the communication cost of the QDWH method is significantly lower than that of the SN method. The QDWH method is an attractive alternative method for the emerging multicore and heterogeneous computing architectures.

In this paper, we focus on the study of the polar decomposition of square and nonsingular matrices. The QDWH method is readily applicable to rectangular and singular matrices, whereas the SN method needs to initially use a rank-revealing QR factorization to enable its applicability to more general matrices [20, p. 196].

The rest of this paper is organized as follows. In section 2, we review Newton's method and its variants. In section 3, we study Halley's iteration and derive a dynamical weighting scheme. A convergence proof of the new method is given. We also show that the cubic convergence makes acceptable a looser convergence criterion than that for the SN iteration. Section 4 discusses implementation issues, in which we show how the DWH method can be computed based on the matrix QR decompositions. Numerical examples are shown in section 5. Concluding remarks are given in section 6. In Appendix A, we give a detailed solution for the max-min problem that arises in the derivation of the DWH method.

Throughout this paper, $\|\cdot\|_p$ denotes the matrix or vector p-norm ($p = 1, 2, \infty$) and $\|\cdot\|_F$ the Frobenius norm. $\|\cdot\|$ denotes a unitarily invariant norm such as $\|\cdot\|_2$ and $\|\cdot\|_F$. $\sigma_i(X)$ denotes the ith singular value of X. $\sigma_{\min}(X)$ and $\sigma_{\max}(X)$ denote the smallest and largest singular values of X, respectively. $\kappa_2(A)$ denotes the 2–norm condition number of A: $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2$. α and β denote $\alpha = \|A\|_2 = \sigma_{\max}(A)$ and $\beta = \|A^{-1}\|_2^{-1} = \sigma_{\min}(A)$. To avoid confusion between the unitary polar factor and the singular value decomposition (SVD) of A, U always denotes the polar factor of A. The SVD of A is expressed by $A = U_*\Sigma V_*^H$ so that $U = U_*V_*^H$.

2. Newton's method.

2.1. Scaled Newton iteration.

The most well-known method for computing the unitary polar factor of a nonsingular matrix A is the Newton iteration

$$X_{k+1} = \frac{1}{2}\left(X_k + X_k^{-H}\right), \quad X_0 = A. \tag{2.1}$$

It can be shown that the iterates X_k converge quadratically to the polar factor U of A and that all singular values $\sigma_i(X_k) \to 1$ as $k \to \infty$ [20, p. 202]. However, the initial phase of convergence is slow when A has a singular value that has large relative distance from 1, that is, when a singular value σ exists such that $\max(|1-\sigma|/\sigma, |1-\sigma|/1) \gg 1$. In order to speed up the initial phase, we can apply the SN iteration

$$X_{k+1} = \frac{1}{2}\left(\zeta_k X_k + (\zeta_k X_k)^{-H}\right), \quad X_0 = A, \tag{2.2}$$

where ζ_k is a scaling factor. The frequently used $(1,\infty)$–norm scaling and Frobenius norm scaling are known to work well in practice [18, 20]. The rigorous convergence theory is established for the so-called optimal scaling $\zeta_k^{\text{opt}} = (\sigma_{\min}(X_k)\sigma_{\max}(X_k))^{-1/2}$ [18]. However, it is not a practical scaling since it is too expensive to compute $\sigma_{\min}(X_k)$ and $\sigma_{\max}(X_k)$ at every iteration. Recently, Byers and Xu [4] proposed the following suboptimal scaling:

$$\zeta_0 = 1/\sqrt{\alpha\beta}, \quad \zeta_1 = \sqrt{2\sqrt{\alpha\beta}/(\alpha+\beta)}\,, \quad \zeta_k = 1/\sqrt{\rho(\zeta_{k-1})} \quad \text{for} \quad k = 2,3,\ldots, \tag{2.3}$$

where $\alpha = \|A\|_2$, $\beta = \|A^{-1}\|_2^{-1}$ and $\rho(\zeta) = (\zeta + \zeta^{-1})/2$. It is called a suboptimal scaling since at the kth iteration, it minimizes the width of the interval containing all the singular values of the kth iterate X_k.

THEOREM 2.1 (see [4]). *The iterates X_k generated by the SN iteration* (2.2) *with the suboptimal scaling* (2.3) *converge quadratically to the polar factor U of A. Moreover, convergence to within a tolerance 10^{-16} is reached within nine iterations if $\kappa_2(A) \le 10^{16}$.*

In practice, it is sufficient to use some rough estimates $\widehat{\alpha}$ and $\widehat{\beta}$ of α and β. For example, one may take $\widehat{\alpha} = \|A\|_F$ and $\widehat{\beta} = 1/\|A^{-1}\|_F$. It is shown that, for any estimates $\widehat{\alpha}$ and $\widehat{\beta}$ such that $0 < \widehat{\beta} \le \|A^{-1}\|_2^{-1} \le \|A\|_2 \le \widehat{\alpha}$ and $\widehat{\alpha}/\widehat{\beta} < 10^{16}$, the iteration converges within nine iterations [4]. It is also known experimentally that the SN iteration with Higham's $(1,\infty)-$scaling [18] needs about the same number of iterations.

It is claimed in [23, 4] that the SN iteration is backward stable provided that the inverse X_k^{-1} is computed in a mixed forward-backward stable way. For example, one can use a bidiagonal reduction-based matrix inverse algorithm as presented in [4]. In that case, the arithmetic cost of each iteration increases to $6n^3$ instead of $2n^3$ when the inverse is computed using the LU factorization with partial pivoting. We note that inversion based on QR factorization without column pivoting does not guarantee backward stability of the SN iteration (see [23]).

2.2. Newton iteration variant. The Newton variant (NV) iteration is

$$Y_{k+1} = 2Y_k\left(I + Y_k^H Y_k\right)^{-1}, \quad Y_0 = A. \tag{2.4}$$

It can be shown that $Y_k = X_k^{-H}$ for $k \ge 1$, where X_k is generated by the Newton iteration (2.1) [21], [20, p. 202]. Note that iteration (2.4) is applicable to singular and rectangular matrices. To speed up the convergence, we can use a scaled version of iteration (2.4). Substituting $\eta_k Y_k$ into Y_k in (2.4) yields the scaled Newton variant (SNV) iteration

$$Y_{k+1} = 2\eta_k Y_k\left(I + \eta_k^2 Y_k^H Y_k\right)^{-1}, \quad Y_0 = A, \tag{2.5}$$

where η_k is the scaling factor. A proper choice of η_k is the one such that $Y_0 = X_0$ and $Y_k = X_k^{-*}$ for $k \geq 1$, where X_k is generated by the SN iteration (2.2). It implies that $\eta_0 = \zeta_0$ and $\eta_k = 1/\zeta_k$.

Since Y_k^{-1} is not computed in the SNV iteration (2.5), the $(1,\infty)$–norm scaling or Frobenius norm scaling is not applicable. How to efficiently scale the SNV iteration (2.5) is listed as Problem 8.26 in [20, p. 219]. One solution to the problem is to use the suboptimal scaling (2.3), which yields the following iteration for the scaling of the SNV iteration (2.5):

$$\eta_0 = 1/\sqrt{\alpha\beta}, \quad \eta_1 = \sqrt{\frac{\alpha+\beta}{2\sqrt{\alpha\beta}}}, \quad \eta_k = \sqrt{\rho(\eta_{k-1})} \quad \text{for} \quad k = 2, 3, \ldots. \tag{2.6}$$

From the connection with the Newton iteration, it follows from Theorem 2.1 that $Y_k \to U^{-H} = U$ as $k \to \infty$.

The SN iteration with the suboptimal scaling (2.3) and the SNV iteration with the scaling (2.6) are mathematically equivalent, provided that the same scalars α and β are used. However, the practical implementation of the SN iteration involves explicit matrix inverses. This is usually done by means of the LU factorization with partial pivoting. Pivoting makes necessarily a large amount of data communication and slows down the total computation time [3, 27]. As pointed out in [20, p. 219], the SNV iteration (2.5) can be implemented using a QR decomposition (without column pivoting). Computing a QR decomposition can be done in a communication friendly way with great performance on modern multicore and heterogeneous systems [15]. Therefore, the QR-based SNV method is an attractive alternative. Unfortunately, as shown in section 5, the SNV iteration (2.5) is not stable for ill-conditioned matrices, even with the QR decomposition-based implementation. The instability had also been independently reported in early studies [7, 5]. In the next section, we will exploit an alternative iteration to develop an inverse free method.

3. Halley's method. Halley's iteration for computing the polar factor of a nonsingular matrix A is

$$X_{k+1} = X_k(3I + X_k^H X_k)(I + 3X_k^H X_k)^{-1}, \quad X_0 = A. \tag{3.1}$$

It is a member of the Padé family of iterations [22]. It is proven that X_k converges globally and that the convergence rate is cubic [10, 11]. However, the initial steps of convergence can still be slow when A has a singular value that has large relative distance from 1. For example, consider the 2×2 matrix

$$A = X_0 = \begin{bmatrix} 1 & \\ & x_0 \end{bmatrix}, \quad x_0 = 10^{-10}. \tag{3.2}$$

The polar factor of A is the 2×2 identity matrix. The kth iterate X_k is given by

$$X_k = \begin{bmatrix} 1 & \\ & x_k \end{bmatrix}, \quad x_k = \frac{x_{k-1}(3 + x_{k-1}^2)}{1 + 3x_{k-1}^2}.$$

After one Halley's iteration, $x_1 \approx 3 \times 10^{-10}$. It takes 24 iterations for the iterate X_k to converge to the polar factor within IEEE double precision machine precision, namely, $\|X_{24} - I\|_2 \leq \epsilon_{\mathrm{M}} = 2.2 \times 10^{-16}$.

To accelerate the convergence of Halley's iteration (3.1), let us consider the following DWH iteration:

$$X_{k+1} = X_k(a_k I + b_k X_k^H X_k)(I + c_k X_k^H X_k)^{-1}, \quad X_0 = A/\alpha, \tag{3.3}$$

where $\alpha = \|A\|_2$ and scalars a_k, b_k, and c_k are nonnegative weighting parameters. We choose these weighting parameters suboptimally[1] in the sense that it maximizes ℓ_{k+1} such that the interval $[\ell_{k+1}, 1]$ contains all the singular values of X_{k+1}. Specifically, let $X_k = U_* \Sigma_k V_*^H$ be the SVD of X_k and ℓ_k be such that[2]

$$[\sigma_{\min}(X_k), \sigma_{\max}(X_k)] \subseteq [\ell_k, 1] \subset (0, 1] \tag{3.4}$$

with initial $\sigma_{\min}(X_0) = \beta/\alpha \equiv \ell_0$ and $\beta = 1/\|A^{-1}\|_2$. Then one step of the DWH iteration (3.3) yields

$$X_{k+1} = U_* \Sigma_k (a_k I + b_k \Sigma_k^2)(I + c_k \Sigma_k^2)^{-1} V_*^H.$$

Hence the singular values $\sigma_i(X_{k+1})$ of X_{k+1} are given by

$$\sigma_i(X_{k+1}) = g_k(\sigma_i(X_k)), \tag{3.5}$$

where g_k is a rational function defined as

$$g_k(x) = x \frac{a_k + b_k x^2}{1 + c_k x^2}.$$

By (3.4) and (3.5), we have

$$[\sigma_{\min}(X_{k+1}), \sigma_{\max}(X_{k+1})] \subseteq \left[\min_{\ell_k \le x \le 1} g_k(x), \max_{\ell_k \le x \le 1} g_k(x) \right]. \tag{3.6}$$

Since the closeness of the iterate X_{k+1} to the polar factor can be measured by the maximum distance between singular values $\sigma_i(X_{k+1})$ and 1, a suboptimal choice of the triplet (a_k, b_k, c_k) should make the function g_k be bounded

$$0 < g_k(x) \le 1 \quad \text{for} \quad \ell_k \le x \le 1 \tag{3.7}$$

and attain the max-min

$$\max_{a_k, b_k, c_k} \left\{ \min_{\ell_k \le x \le 1} g_k(x) \right\}. \tag{3.8}$$

Once these parameters a_k, b_k, and c_k are found to satisfy (3.7) and (3.8), all singular values of X_{k+1} satisfy

$$[\sigma_{\min}(X_{k+1}), \sigma_{\max}(X_{k+1})] \subseteq [\ell_{k+1}, 1] \subset (0, 1], \tag{3.9}$$

where $\ell_{k+1} = \min_{\ell_k \le x \le 1} g_k(x)$.

[1] The term "suboptimal" follows that of the suboptimal scaling (2.3) for the SN iteration, which minimizes b_{k+1} such that $[1, b_{k+1}]$ contains all the singular values of X_{k+1}.

[2] In (3.4) one can assume a more general interval $[\widehat{\ell}_k, L]$ for any $L > 0$, but setting $L \equiv 1$ causes no loss of generality since the simple scaling $a_{k-1} \leftarrow a_{k-1}/L$, $b_{k-1} \leftarrow b_{k-1}/L$ maps the interval $[\widehat{\ell}_k, L]$ containing $[\sigma_{\min}(X_k), \sigma_{\max}(X_k)]$ to $[\widehat{\ell}_k/L, 1] \equiv [\ell_k, 1]$.

Let us now consider how to solve the optimization problem (3.7) and (3.8). To satisfy $g_k(x) > 0$, we can impose

$$a_k, b_k, c_k > 0 \tag{3.10}$$

and

$$g_k(1) = 1. \tag{3.11}$$

These conditions ensure that the function $g_k(x)$ is positive and continuously differentiable for $x > 0$ and has a fixed point at 1. Note that (3.11) implies $c_k = a_k + b_k - 1$. By the assumptions (3.10) and (3.11), the optimization problem (3.7) and (3.8) can be stated as follows.

The bounded max-min problem: find $a_k, b_k > 0$ such that $c_k = a_k + b_k - 1 > 0$,

$$0 < g_k(x) \le 1 \quad \text{for} \quad \ell_k \le x \le 1, \tag{3.12}$$

and

$$\max_{a_k, b_k > 0} \left\{ \min_{\ell_k \le x \le 1} g_k(x) \right\} \tag{3.13}$$

is attained.

In Appendix A, we show that the solution of the optimization problem (3.12) and (3.13) is given by

$$a_k = h(\ell_k), \quad b_k = (a_k - 1)^2/4, \tag{3.14}$$

where

$$h(\ell) = \sqrt{1+d} + \frac{1}{2}\sqrt{8 - 4d + \frac{8(2-\ell^2)}{\ell^2\sqrt{1+d}}}, \quad d = \sqrt[3]{\frac{4(1-\ell^2)}{\ell^4}}. \tag{3.15}$$

Similar to the SN iteration (2.2) with the suboptimal scaling (2.3), we see that the weighting parameters a_k, b_k and $c_k = a_k + b_k - 1$ of the DWH iteration (3.3) can be generated by simple scalar iterations in which the initial value ℓ_0 depends on the extreme singular values of the original matrix A.

In summary, we derive the DWH iteration (3.3) for computing the polar factor of A, where the weighting parameters a_k and b_k are generated by the scalar iterations (3.14), c_k is defined by $c_k = a_k + b_k - 1$, and

$$\ell_0 = \frac{\beta}{\alpha}, \quad \ell_k = \frac{\ell_{k-1}(a_{k-1} + b_{k-1}\ell_{k-1}^2)}{1 + c_{k-1}\ell_{k-1}^2} \quad \text{for} \quad k = 1, 2, \ldots, \tag{3.16}$$

where $\alpha = \|A\|_2$ and $\beta = 1/\|A^{-1}\|_2$.

Before we prove the global convergence of the DWH iteration (3.3), let us recall the 2×2 matrix A defined as (3.2). The kth DWH iterate X_k is given by

$$X_k = \begin{bmatrix} 1 & \\ & x_k \end{bmatrix}, \quad x_k = \frac{x_{k-1}(a_k + b_k x_{k-1}^2)}{1 + c_k x_{k-1}^2}.$$

Since $\alpha = 1$ and $\ell_0 = 10^{-10}$, by (3.14), we have $a_0 \simeq 1.17 \times 10^7$, $b_0 \simeq 3.42 \times 10^{13}$, and $c_0 \simeq 3.42 \times 10^{13}$. After one iteration, x_0 is mapped to $x_1 \simeq 1.17 \times 10^{-3}$, which is much

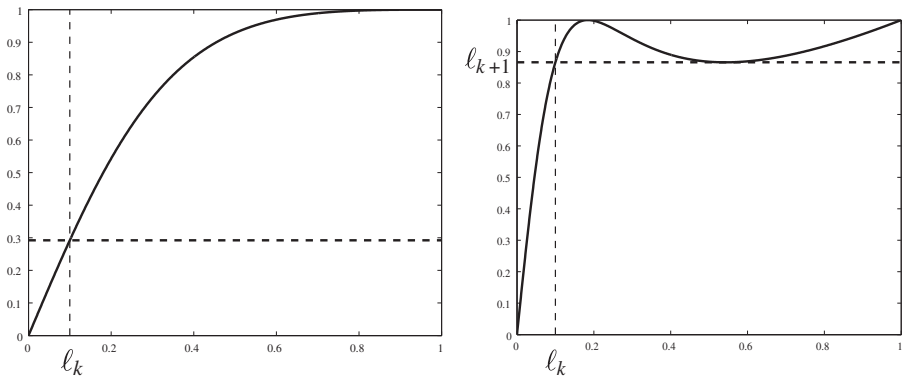

FIG. 3.1. *The mapping functions* $g_k(\sigma) = \sigma(3+\sigma^2)/(1+3\sigma^2)$ *of the Halley iteration (left) and* $g_k(\sigma) = \sigma(a_k + b_k\sigma^2)/(1+c_k\sigma^2)$ *of the DWH iteration (right).*

closer to the target value 1 than the first iterate computed by Halley's iteration (3.1). In fact, it takes only five DWH iterations to approximate the polar factor within the machine precision $\|X_5 - I\|_2 \le \epsilon_{\mathrm{M}}$. It is a factor of five times faster than the Halley iteration. To explain the fast convergence of the DWH iteration, the plots of Figure 3.1 show the typical mapping functions g_k from the singular values of X_k to that of X_{k+1} by the Halley iteration (3.1) and the DWH iteration (3.3). We can see that the singular values of X_k are mapped much closer to 1 by the DWH iteration than the Halley iteration.

THEOREM 3.1. *For a nonsingular* A*, the iterates* X_k *generated by the DWH iteration* (3.3) *converge to the polar factor* U *of* A*. The asymptotic convergence factor is cubic.*

Proof. We first prove the convergence of the iterates X_k. This is equivalent to showing that the singular values $\sigma_i(X_k) \to 1$ as $k \to \infty$ for all $1 \le i \le n$. By (3.9), we have $[\sigma_{\min}(X_k), \sigma_{\max}(X_k)] \subseteq [\ell_k, 1]$. Hence it suffices to prove $\ell_k \to 1$ as $k \to \infty$.

Using (3.14), (3.16), and $c_k = a_k + b_k - 1$, we derive

$$\frac{1}{\ell_{k+1}} - 1 = F(a_k, \ell_k)\left(\frac{1}{\ell_k} - 1\right),$$

where

$$F(a,\ell) = \frac{((a-1)\ell - 2)^2}{4a + (a-1)^2\ell^2}.$$

Note that $F(a_k, \ell_k) \ge 0$ since $a_k > 0$. All we need to show is that there is a positive constant $\delta < 1$ such that $F(a_k, \ell_k) \le \delta$ for all k. In fact, we have $3 \le a \le \frac{2+\ell}{\ell}$ (see (A.9) in Appendix A), and on this interval $F(a,\ell)$ is a decreasing function of a:

$$\frac{\partial F}{\partial a} = \frac{4(1+\ell)(\ell^2(a-1)^2 - 4)}{(\ell^2 + a^2\ell^2 + 2a(2-\ell^2))^2} \le 0 \quad \text{on} \quad 3 \le a \le \frac{2+\ell}{\ell}.$$

Therefore, we have

$$F(a_k, \ell_k) \le F(3, \ell_k) = \frac{(3-1)^2\ell_k^2 - 4(3-1)\ell_k + 4}{(3-1)^2\ell_k^2 + 4\cdot 3} = \frac{(1-\ell_k)^2}{\ell_k^2 + 3} \le \frac{1}{3} = \delta.$$

This completes the proof of the global convergence of the DWH iteration.

Now we consider the asymptotic rate of convergence. By the above argument,

$$|1-\ell_{k+1}| = \left| F(a_k,\ell_k)\left(\frac{1}{\ell_k}-1\right)\ell_{k+1}\right| \le \left|\frac{\ell_{k+1}(1-\ell_k)^2}{\ell_k^2+3}\left(\frac{1}{\ell_k}-1\right)\right| = \frac{\ell_{k+1}\,|1-\ell_k|^3}{\ell_k(\ell_k^2+3)}.$$

By the fact that $\ell_k \to 1$, we conclude that the DWH is asymptotically cubically convergent. ∎

Remark 1. It is shown in Appendix A that a_k satisfies

$$3 \le a_k \le \frac{2+\ell_k}{\ell_k} \qquad \text{for} \quad k \ge 0,$$

where $0 < \ell_k \le 1$. Note that as $\ell_k \to 1$, $(a_k, b_k, c_k) \to (3, 1, 3)$. These are the weighting parameters of the Halley iteration (3.1). It is an open problem how large a_k influences the accuracy of the computed unitary polar factor U.

Remark 2. For simplicity of exposition, we used the exact extreme singular values of the original matrix A in the analysis, namely, $\alpha = \sigma_{\max}(A)$, $\beta = \sigma_{\min}(A)$, and $\ell_0 = \beta/\alpha = 1/\kappa_2(A)$. In fact, estimates $\widehat{\alpha}$ and $\widehat{\beta}$ of α and β are sufficient as long as the inclusion property $[\sigma_{\min}(A/\widehat{\alpha}), \sigma_{\max}(A/\widehat{\alpha})] \subseteq [\widehat{\ell}_0, 1]$ holds, where $\widehat{\ell}_0 = \widehat{\beta}/\widehat{\alpha}$.

Remark 3. In [11], Gander has observed the slow convergence with respect to small singular values in Halley's iteration. As a remedy and generalization to rectangle and rank-deficient matrices, he proposed the following weighting parameters:

$$a_k = \frac{2\tau-3}{\tau-2}, \quad b_k = \frac{1}{\tau-2}, \quad c_k = \frac{\tau}{\tau-2}, \tag{3.17}$$

where τ is a prescribed parameter. When $\tau = 3$, it is the Halley iteration. It is proved that, for any $\tau > 2$, the resulting method converges globally and quadratically [25]. In practice, Gander [11] suggests taking $\tau = 2 + \epsilon_{\mathrm{M}}/\delta$ for $\delta > 10\epsilon_{\mathrm{M}}$ and $\tau = 2.1$ for $\epsilon_{\mathrm{M}} < \delta \le 10\epsilon_{\mathrm{M}}$, where ϵ_{M} is the machine epsilon and δ is the convergence tolerance. This stems from the observation that taking τ close to 2 results in both speed-up and instability. We here set the tolerance δ small enough, in which case $\tau = 2.1$. Note that Gander's iteration switches from iteration (3.3) with static weighting parameter (3.17) to the standard Halley iteration (3.1) after a certain number of iterations. To find the appropriate switching strategy, it is noticed that about $s = -\log(\ell_0)$ steps are needed for the smallest singular value to increase to the size of 1, where $\ell_0 = \beta/\alpha = \sigma_{\min}(X_0)$. Therefore, the switching is done after s iterations using $\tau = 2.1$. Unfortunately, the convergence of Gander's iteration can still be slow. For the 2×2 matrix in (3.2), Gander's iteration needs 14 iterations to converge. In section 5, we see that as many as 20 iterations are needed for some cases.

To derive a stopping criterion for the DWH iteration (3.3), we note that, once convergence sets in, $\ell_k \simeq 1$ so that $(a_k, b_k, c_k) \simeq (3, 1, 3)$. Therefore, we will just need to consider a proper stopping criterion for Halley's iteration (3.1). We note that in the SN iteration with Higham's $(1,\infty)$–norm scaling, switching to the unscaled Newton iteration is recommended [20, 23]. As for the DWH iteration, this switching is not necessary because we have $(a_k, b_k, c_k) \to (3, 1, 3)$. This is also true for the SN iteration with suboptimal scaling, which guarantees the scaling factor $\zeta_k \to 1$.

We first have the following lemma.

LEMMA 3.2. *For Halley's iteration* (3.1)*, if* $\|X_{k-1} - U\|_2 = \|I - \Sigma_{k-1}\|_2 = \epsilon \ll 1$*, then up to the first order in* ϵ*,*

$$\|X_{k-1} - U\| = (1 + O(\epsilon^2))\|X_k - X_{k-1}\|.$$

Proof. Writing $X_{k-1} = U_*\Sigma_{k-1}V_*$ we have

$$\begin{aligned} X_k - X_{k-1} &= X_{k-1}(3I + X_{k-1}^H X_{k-1})(I + 3X_{k-1}^H X_{k-1})^{-1} - X_{k-1} \\ &= 2X_{k-1}(I - X_{k-1}^H X_{k-1})(I + 3X_{k-1}^H X_{k-1})^{-1} \\ &= 2U_*(I - \Sigma_{k-1}^2)\Sigma_{k-1}(I + 3\Sigma_{k-1}^2)^{-1}V_*^H. \end{aligned} \tag{3.18}$$

Taking an unitarily invariant norm and using the inequality $\|AB\| \le \|A\| \cdot \|B\|_2$, we get

$$\begin{aligned} \|X_k - X_{k-1}\| &\le 2\|U_*(I - \Sigma_{k-1})V_*^H\| \cdot \|\Sigma_{k-1}(I + \Sigma_{k-1})(I + 3\Sigma_{k-1}^2)^{-1}\|_2 \\ &= 2\|X_{k-1} - U\| \cdot \|\Sigma_{k-1}(I + \Sigma_{k-1})(I + 3\Sigma_{k-1}^2)^{-1}\|_2 \\ &\equiv 2\|X_{k-1} - U\| \cdot \|f(\Sigma_{k-1})\|_2, \end{aligned} \tag{3.19}$$

where $f(x) = x(1+x)/(1+3x^2)$ is a continuouse and differentiable function. It is easy to see that $f(x)$ is increasing on $(0,1)$ and decreasing on $(1,\infty)$. It attains the maximum $1/2$ at $x = 1$. Hence we can write $f(1-\epsilon) = 1/2 - O(\epsilon^2)$ for $\epsilon \ll 1$. Consequently, we have $\|f(\Sigma_{k-1})\|_2 = \max_i |f(\sigma_i)| = 1/2 - O(\epsilon^2)$. By (3.19), it follows that $\|X_k - X_{k-1}\| \le (1 - O(\epsilon^2))\|X_{k-1} - U\|$.

We can prove $\|X_k - X_{k-1}\| \ge (1 - O(\epsilon^2))\|X_{k-1} - U\|$ similarly by noticing from (3.18) that $X_{k-1} - U = \frac{1}{2}(X_k - X_{k-1})(V_*(I + \Sigma_{k-1})\Sigma_{k-1}(I + 3\Sigma_{k-1}^2)^{-1}V_*^H)^{-1}$ and using $1/f(1+\epsilon) = 2 + O(\epsilon^2)$. ∎

Now, by the identity $U_*(\Sigma_{k-1} - I)V_*^H = X_{k-1} - U$, we have

$$\begin{aligned} \|X_k - U\| &= \|U_*(\Sigma_{k-1}(3I + \Sigma_{k-1}^2)(I + 3\Sigma_{k-1}^2)^{-1} - I)V_*^H\| \\ &= \|U_*(\Sigma_{k-1} - I)^3(I + 3\Sigma_{k-1}^2)^{-1}V_*^H\| \\ &\le \|X_{k-1} - U\|^3 \cdot \|(I + 3X_{k-1}^H X_{k-1})^{-1}\|_2, \end{aligned}$$

where we used the inequality $\|AB\| \le \|A\| \cdot \|B\|_2$ again. When close to convergence, $\|X_{k-1} - U\|_2 \ll 1$. Hence, by Lemma 3.2, we have

$$\|X_k - U\| \lesssim \|X_k - X_{k-1}\|^3 \cdot \|(I + 3X_{k-1}^H X_{k-1})^{-1}\|_2.$$

This suggests that we accept X_k when

$$\|X_k - X_{k-1}\|_F \le \left(\frac{\epsilon_{\mathrm{M}}}{\|(I + 3X_{k-1}^H X_{k-1})^{-1}\|_2} \right)^{1/3}. \tag{3.20}$$

Close to convergence $X_{k-1}^H X_{k-1} \simeq I$, the test (3.20) is effectively

$$\|X_k - X_{k-1}\|_F \le (4\epsilon_{\mathrm{M}})^{1/3}. \tag{3.21}$$

We recall that for quadratically convergent methods such as the SN method (2.2) and its variant (2.5), the following stopping criterion is suggested [20, p. 208]:

$$\|X_k - X_{k-1}\|_F \le (2\epsilon_{\mathrm{M}})^{1/2}. \tag{3.22}$$

In [4], it is noted that theoretically the SN iteration with the suboptimal scaling converges in at most nine steps for any matrix of condition number less than 10^{16}. It is based on the bound $\|X_k - U\|_2 \le b_k - 1$, where b_k can be obtained by a simple

scalar iteration. Consequently, the first k satisfying $|1-b_k| < 10^{-16}$ provides an upper bound on the number of iteration counts.

We can derive a similar result for the DWH iteration (3.3). By the interval (3.9) that bounds the singular values of the iterate X_k, we have

$$\|X_k - U\|_2 = |1 - \sigma_{\min}(X_k)| \le |1 - \ell_k|.$$

Hence, by finding the first k such that $|1 - \ell_k| < 10^{-16}$, we obtain the number of iterations needed for the DWH iteration to convergence. Specifically, by using the scalar recursion (3.16) with $\ell_0 = 1/\kappa_2(A)$, we have the following upper bounds for the number of DWH iterations:

$\kappa_2(A)$	10^1	10^2	10^5	10^8	10^{10}	10^{12}	10^{16}
SN, SNV	5	6	7	8	8	9	9
DWH	3	4	5	5	5	5	6

The result suggests that the DWH iteration converges within at most six steps for any matrix with condition number $\kappa_2(A) \le 10^{16}$. The number of DWH iterations is about one-third fewer than the number of SN iterations (2.2) with the suboptimal scaling (2.3).

4. QR-based implementations. In this section, we discuss an implementation of the DWH iteration (3.3) using the QR decomposition. The QR-based implementation is more desirable than those involving explicit inverses for enhancing parallelizability. Numerical examples in section 5 suggest that it also improves the numerical stability.

First we have the following basic result, given in [20, p. 219] and based on the results in [29].

THEOREM 4.1. *Let* $\begin{bmatrix} \eta X \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$ *be a QR decomposition of* $\begin{bmatrix} \eta X \\ I \end{bmatrix}$, *where* $X, Q_1 \in \mathbb{C}^{m\times n}$ *and* $Q_2, R \in \mathbb{C}^{n\times n}$. *Then*

$$Q_1 Q_2^H = \eta X (I + \eta^2 X^H X)^{-1}. \tag{4.1}$$

Proof. By the polar decomposition

$$\begin{bmatrix} \eta X \\ I \end{bmatrix} = \widetilde{U}\widetilde{H}, \tag{4.2}$$

we have $\widetilde{H}^2 = I + \eta^2 X^H X$ and $\widetilde{H} = (I + \eta^2 X^H X)^{1/2}$. Note that $\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$ and $\widetilde{U}$ span the column space of $\begin{bmatrix} \eta X \\ I \end{bmatrix}$ and that they are orthogonal matrices. Hence it follows that

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \widetilde{U} W \tag{4.3}$$

for some orthogonal matrix W. By (4.2) and (4.3), we have

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \begin{bmatrix} \eta X \\ I \end{bmatrix} (I + \eta^2 X^H X)^{-1/2} W.$$

The identity (4.1) can now be verified by a straightforward calculation. ∎

By Theorem 4.1, we immediately derive that the SNV iteration (2.5) is mathematically equivalent to the following iteration, referred to as a QR-based scaled Newton variant (QSNV):

$$\begin{cases} \begin{bmatrix} \eta_k X_k \\ I \end{bmatrix} &= \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R, \\ X_{k+1} &= 2Q_1 Q_2^H, \end{cases} \tag{4.4}$$

with the initial $X_0 = A$, where the scaling factor η_k is defined as (2.6). The following is a pseudocode of the QSNV iteration:

QSNV algorithm:

$X_0 = A$
$\eta_0 = 1/\sqrt{\alpha\beta}$ and $k = 0$
repeat
  compute QR decomposition $\begin{bmatrix} \eta_k X_k \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$
  $X_{k+1} = 2Q_1 Q_2^H$
  if $k = 0$ **then**
    $\eta_1 = \sqrt{(\alpha+\beta)/(2\sqrt{\alpha\beta})}$
  else
    $\eta_{k+1} = \sqrt{(\eta_k + 1/\eta_k)/2}$
  end if
  $k = k + 1$
until convergence
$U = X_k$

Now we consider the DWH iteration (3.3). Iteration (3.3) can be equivalently written as

$$X_{k+1} = \frac{b_k}{c_k} X_k + \left(a_k - \frac{b_k}{c_k}\right) X_k (I + c_k X_k^H X_k)^{-1}, \quad X_0 = A/\alpha, \tag{4.5}$$

where the weighting triplet (a_k, b_k, c_k) is defined as (3.14). By Theorem 4.1, iteration (4.5) can be written using the QR decomposition as follows, referred to as the QR-based dynamically weighted Halley (QDWH) iteration:

$$\begin{cases} \begin{bmatrix} \sqrt{c_k} X_k \\ I \end{bmatrix} &= \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R, \\ X_{k+1} &= \dfrac{b_k}{c_k} X_k + \dfrac{1}{\sqrt{c_k}} \left(a_k - \dfrac{b_k}{c_k}\right) Q_1 Q_2^H. \end{cases} \tag{4.6}$$

The following is a pseudocode of the QDWH iteration:

QDWH algorithm:

$X_0 = A/\alpha$, $\ell_0 = \beta/\alpha$
$k = 0$
repeat
  $a_k = h(\ell_k)$, $b_k = (a_k - 1)^2/4$, $c_k = a_k + b_k - 1$
  compute QR decomposition $\begin{bmatrix} \sqrt{c_k} X_k \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$
  $X_{k+1} = (b_k/c_k) X_k + (1/\sqrt{c_k})\,(a_k - b_k/c_k)\, Q_1 Q_2^H$

$\ell_{k+1} = \ell_k(a_k + b_k\ell_k^2)/(1 + c_k\ell_k^2)$
$k = k + 1$
until convergence
$U = X_k$

For the practical implementations of the QSNV and QDWH methods, we only need estimates $\widehat{\alpha}$ and $\widehat{\beta}$ of α and β satisfying $0 < \widehat{\beta} \le \sigma_{\min}(A) \le \sigma_{\max}(A) \le \widehat{\alpha}$. We can simply use $\widehat{\alpha} = \|A\|_F$. An estimate of $\beta = \sigma_{\min}(A)$ is a nontrivial task [6, 16, 17]. For the SN method, the estimate $\widehat{\beta} = 1/\|A^{-1}\|_F$ is suggested in [4]. However, it is not practical for the QSNV and QDWH methods since A^{-1} is not calculated explicitly. By the inequality $\|A\|_1/\sqrt{n} \le \|A\|_2 \le \sqrt{n}\|A\|_1$, we have $\beta = \sigma_{\min}(A) = \|A^{-1}\|_2^{-1} \ge (\sqrt{n}\|A^{-1}\|_1)^{-1}$. Therefore, we may use the lower bound of β as an estimate, i.e.,

$$\widehat{\beta} = (\gamma\sqrt{n})^{-1}, \tag{4.7}$$

where γ is the LAPACK 1-norm estimate of A^{-1} [19, Chap. 15]. In section 5, we will examine the effect of the estimate $\widehat{\beta}$ on the convergence of the QDWH iteration. The numerical examples suggest that it is harmless to use a rough estimate $\widehat{\beta}$ as far as $\widehat{\ell}_0 = \widehat{\beta}/\widehat{\alpha}$ is a lower bound of $\sigma_{\min}(X_0)$. We note that QDWH and Gander's algorithm use the normalized initial matrix $X_0 = A/\alpha$, whereas SN and QSNV use $X_0 = A$. However, the scalars α and β need to be provided in all these methods.

To end this section, let us consider the arithmetic cost of the QDWH method. Note that the QSNV and QDWH iterations share the same computational kernel, namely,

(a) compute $\begin{bmatrix} \eta X \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$, and
(b) form $\widehat{X} = Q_1 Q_2^H$.

A straightforward implementation is to first compute the QR decomposition of the $2n \times n$ matrix by a dense QR decomposition using the LAPACK routine `DGEQRF` [1]. The cost is $\frac{10}{3}n^3$ flops. Then we form Q_1 and Q_2 explicitly by using `DORGQR`. Its cost is $\frac{10}{3}n^3$ flops. Finally, we compute the product $Q_1 Q_2^H$ by the matrix-matrix multiplication routine `DGEMM` in BLAS with the cost $2n^3$ flops. In summary, the arithmetic cost of each QDHW iteration is $\frac{26}{3}n^3$ flops. Since it generally takes at most six iterations to converge, the total cost of the QDWH method is at most $52n^3$ flops.

In contrast, the cost of each SN iteration is $2n^3$ flops if the matrix inversion is computed by LU factorization-based routines `DGETRF` and `DGETRI` in LAPACK. Together with the fact that it generally needs at most nine steps to converge, the total cost of the SN method is at most $18n^3$ flops. Therefore, the cost of the QDWH method is about three times more than that of the SN method. If the matrix inversion in the SN iteration is calculated using a bidiagonal reduction-based algorithm for backward stability [4], then it increases the cost to $6n^3$ flops per iteration. This makes the total cost up to $54n^3$ flops. In this case, the costs of the SN and QDWH methods are about the same.

We note that it is possible to reduce the cost of the QDWH method by exploiting the diagonal block in the QR decomposition step. We can first compute the QR decomposition of ηX and then carefully reduce the augmented matrix into a triangular form by using Givens rotations. The cost per QDWH iteration is reduced to $(16/3)n^3$ flops. Thus the cost of six iterations of QDWH iterations is thereby bounded by $32n^3$ flops. We plan to report the detail of this algorithm and its parallel implementation in future work.

5. Numerical examples. This section shows several numerical experiments to demonstrate the numerical behaviors of the QDWH method. All numerical experiments were performed in MATLAB 7.4.0 and run on a PC with Intel® Core™ 2 Duo processor. The machine epsilon is $\epsilon_{\mathrm{M}} \simeq 2.2 \times 10^{-16}$. The stopping criterion (3.21) is used for the cubically convergent methods, namely, Halley, Gander, DWH, and QDWH iterations. For the quadratically convergent Newton-type methods, namely, SN, NV, SNV, and QSNV iterations, the stopping criterion (3.22) is applied. Since A^{-1} is computed explicitly in the SN iteration, the estimates of extreme singular values are $\widehat{\alpha} = \|A\|_F$ and $\widehat{\beta} = 1/\|A^{-1}\|_F$. Otherwise, we use the estimates $\widehat{\alpha} = \|A\|_F$ and $\widehat{\beta}$ as in (4.7).

The accuracy of the computed polar decomposition is tested by the residual norm $\mathrm{res} = \|A - \widehat{U}\widehat{H}\|_F/\|A\|_F$, where $\widehat{U}$ is the computed polar factor of A and $\widehat{H}$ is the computed Hermitian factor given by $\widehat{H} = \frac{1}{2}(\widehat{U}^H A + (\widehat{U}^H A)^H)$. A method is said to have behaved in a numerically backward stable manner when the residual norm is smaller than $c\epsilon_{\mathrm{M}}$ for a moderate constant c [20, p. 209].

Example 1. This example shows the effectiveness of the dynamical weighting in terms of the number of iterations. Let A be 20×20 diagonal matrices such that the diagonal elements form a geometric series with $a_{11} = 1/\kappa$ and $a_{nn} = 1$. The condition numbers of the matrices A are $\kappa = 10, 10^2, 10^5, \ldots, 10^{20}$. The reason for picking a diagonal matrix is to minimize the effects of rounding errors. The following data show the iteration counts and residual norms of three variants of Halley's method.

κ		10	10^2	10^5	10^{10}	10^{15}	10^{20}
iter	Halley (3.1)	5	7	14	24	35	45
	Gander (3.17)	6	7	9	14	18	24
	DWH (3.3)	4	4	5	5	6	6
res	Halley (3.1)	4.7e-16	5.4e-16	2.4e-16	1.1e-16	1.0e-16	1.1e-16
	Gander (3.17)	7.6e-16	7.5e-16	8.0e-16	7.4e-16	8.0e-16	6.4e-16
	DWH (3.3)	4.9e-16	3.8e-16	3.1e-16	5.7e-16	6.6e-16	5.4e-16

From the above table, we see that Gander's iteration is faster than Halley's iteration but still increases substantially with the increase of the condition numbers. The DWH iteration converges the fastest, all within six steps as predicted in section 3.

Example 2. The purpose of this example is to show that three variants of the Newton iteration are numerically unstable. Consider the simple 3×3 matrix $A = U_* \Sigma V_*^T$, where $\Sigma = \mathrm{diag}\{10^8, 1, 10^{-8}\}$,

$$U_* = \begin{bmatrix} \sin\theta & 0 & \cos\theta \\ 0 & 1 & 0 \\ -\cos\theta & 0 & \sin\theta \end{bmatrix} \quad \text{and} \quad V_* = \begin{bmatrix} \sin\theta & \cos\theta & 0 \\ -\cos\theta & \sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \theta = \pi/3.$$

The following table shows that three variants of Newton's iteration, namely, the NV iteration (2.4), SNV iteration (2.5), and QSNV iteration (4.4), are numerically unstable. The QR-based implementation in the QSNV iteration improves the backward stability, but it is still not numerically backward stable to machine precision.

	SN	NV	SNV	QSNV	DWH	QDWH
iter	9	31	9	9	6	6
res	1.1e-16	4.9e-3	5.1e-3	1.1e-9	3.1e-11	3.3e-16

The instability of the SNV method, including QSNV, has been observed in previous studies [7, 5]. This numerical observation led us to give up QSNV and turn to the study

of a Halley-type iteration. We note that, from the last column of the previous table, the QDWH method performed in a backward stable manner to machine precision.

Example 3. This example is to compare the SN and QDWH methods on numerical stability and convergence rate. The bidiagonal reduction-based matrix inversion method is used in the SN method to guarantee the numerical backward stability. We construct three groups of 20×20 test matrices using the MATLAB function `gallery('randsvd',20,kappa)`, where the condition number `kappa` is set to be 10^2, 10^8, and 10^{15}, respectively. The following table shows the minimum and maximum numbers of iterations and residual norms from 100 test runs.

$\kappa_2(A)$		10^2		10^8		10^{15}	
		min	max	min	max	min	max
iter	QDWH	4	5	5	5	6	6
	SN	6	6	8	8	9	9
res	QDWH	4.2e-16	7.8e-16	4.7e-16	8.1e-16	2.8e-16	7.1e-16
	SN	4.3e-16	6.5e-16	5.8e-16	9.5e-16	3.4e-16	1.2e-15

We observe that both SN and QDWH methods exhibit excellent numerical stability. The QDWH method needs about two-thirds as many iterations as the SN method does, as discussed in section 3. We have also tested many other types of matrices such as extremely ill-conditioned Hilbert matrices. In all our experiments, the QDWH method converged within six iterations and performed in a backward stable manner.

Example 4. In this example, we examine the sufficiency of the QDWH stopping criterion (3.21), which is looser than the one used for SN and QSNV. We generated 100 test matrices as in Example 3, where the condition number `kappa` is set to be 10^8. The following table shows the values $\|X_k - X_{k-1}\|_F$, the corresponding residual norms $\|A - \widehat{U}\widehat{H}\|_F/\|A\|_F$, and the distance from orthogonality $\|X_k^H X_k - I\|_F$ at the iterations $k = 4, 5, 6$.

k	4		5		6	
	min	max	min	max	min	max
$\|X_k - X_{k-1}\|_F$	4.2e-2	6.1e-2	1.7e-7	5.1e-7	1.5e-15	2.4e-15
res	6.6e-8	2.2e-7	4.7e-16	8.1e-16	4.8e-16	7.8e-16
$\|X_k^H X_k - I\|_F$	3.6e-7	1.0e-6	1.9e-15	3.0e-15	2.0e-15	3.2e-15

As we can see, when the QDWH stops at $k = 5$ after satisfying the stopping criterion (3.21), the residual norms and the distance from orthogonality are at the order of 10^{-15} or smaller. Therefore, the stopping criterion (3.21) is a reliable and realistic stopping criterion.

Example 5. In this example, we investigate the impact of estimates $\widehat{\alpha}$ and $\widehat{\beta}$ of $\alpha = \sigma_{\max}(A)$ and $\beta = \sigma_{\min}(A)$ on the convergence of the QDWH method. Since $\|A\|_F/\sqrt{n} \le \|A\|_2 \le \|A\|_F$, $\widehat{\alpha} = \|A\|_F$ is a safe and reliable choice (see Remark 2). Let us focus on the effect of the estimate $\widehat{\beta}$. Let $A \in \mathbb{R}^{20\times 20}$ be generated by using `randsvd` as in Example 3 with $\kappa_2(A) = 10^8$. The following table shows the number of QDWH iterations and residual errors for different estimates $\widehat{\beta}$.

$\widehat{\beta}/\beta$	10^{-9}	10^{-6}	10^{-3}	1	10^3	10^6	10^9
iter	6	6	6	5	12	18	24
res	5.8e-16	6.2e-16	7.3e-16	5.8e-16	6.1e-16	8.2e-16	9.3e-16

These results suggest that a severely overestimated $\widehat{\beta}$ slows down the convergence substantially but that an underestimated $\widehat{\beta}$ is essentially harmless on the convergence

rate and numerical stability. We further performed many tests for other types of matrices and drew the same conclusion. Hence, in practice, it is important to make sure that $\widehat{\beta} \le \sigma_{\min}(A)$ if possible. This observation has led us to use the estimate in (4.7). Why such crude estimates of $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ work so well is a topic of future study.

6. Conclusion. A dynamical weighting scheme for the Halley iteration is introduced in this paper. It is proven that the DWH method is globally and asymptotically cubically convergent. The DWH method can be implemented using QR decompositions without explicit matrix inversions, which is desirable for the emerging multicore and heterogeneous computing systems. Extensive numerical results indicate that the QDWH method performs in the same backward stable way as the SN method. The QDWH method is more expensive in arithmetic cost than the SN iteration with LU-based inversions, and it is about the same if the SN iteration is implemented using the bidiagonal reduction-based matrix inversions. Theoretical proof of the numerical backward stability of the QDWH method is a subject of future study.

Appendix A: Solving the max-min problem. In this appendix, we consider an analytic solution of the optimization problem (3.12) and (3.13). In [26], Nie describes a scheme to reformulate the problem as a standard semidefinite programming (SDP) problem so that we can solve it by using an SDP software such as SeDuMi [28].

Let us restate the optimization problem (3.12) and (3.13) as follows:

Let

$$g(x;a,b) = \frac{x(a+bx^2)}{1+(a+b-1)x^2},$$

where $(a,b) \in \mathcal{D} = \{(a,b) \mid a > 0, b > 0 \text{ and } a+b > 1\}$. Let ℓ be a prescribed constant and $0 < \ell \le 1$. Find $(a_*, b_*) \in \mathcal{D}$ such that

$$0 < g(x;a_*,b_*) \le 1 \quad \text{for} \quad \ell \le x \le 1, \tag{A.1}$$

and (a_*, b_*) attains the max-min

$$\max_{(a,b)\in\mathcal{D}} \left\{ \min_{\ell\le x\le 1} g(x;a,b) \right\}. \tag{A.2}$$

We first consider the case $0 < \ell < 1$ and treat the case $\ell = 1$ at the end.

A.1 Partition of $\mathcal{D}$. First we note that $g(x;a,b)$ is a continuously differentiable odd function of x. The first and second partial derivatives of $g(x;a,b)$ with respect to x are

$$\partial_x g(x;a,b) = \frac{b(a+b-1)x^4 - (a(a+b-1)-3b)x^2 + a}{(1+(a+b-1)x^2)^2} \tag{A.3}$$

and

$$\partial_{xx} g(x;a,b) = \frac{2(a-1)(a+b)x((a+b-1)x^2-3)}{(1+(a+b-1)x^2)^3}. \tag{A.4}$$

The derivative of $g(x;a,b)$ with respect to a is given by

$$\partial_a g(x;a,b) = \frac{x(1-x^2)(1+bx^2)}{(1+(a+b-1)x^2)^2}. \tag{A.5}$$

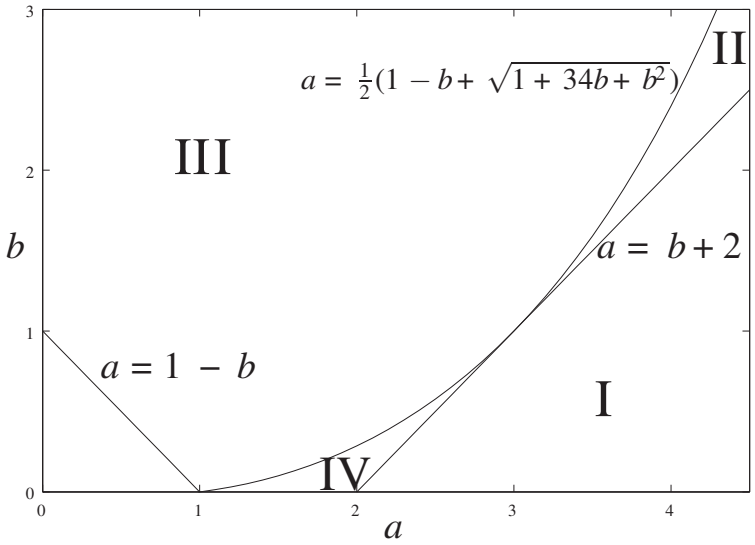

FIG. A.1. *Partition of domain* $\mathcal{D}$.

It is easy to see that $g(x; a, b)$ is a strictly increasing function of a on $0 < x < 1$.

By some basic algebra manipulation, we derive the following lemma.

LEMMA A.1. *Consider the domain* $(a, b) \in \mathcal{D}$. *If* $a > \Gamma \equiv \frac{1}{2}(1-b+\sqrt{1+34b+b^2})$, *then* $g(x; a, b)$ *has two real positive critical points* $0 < x_m(a, b) < x_M(a, b)$. *If* $a = \Gamma$, *then* $g(x; a, b)$ *has one critical point* $0 < x_m(a, b)$. *If* $a < \Gamma$, *then* $g(x; a, b)$ *has no real critical points. Furthermore,* $x_m(a, b) > 1$ *if and only if* $1 < a < 3$ *and* $a < b + 2$, *and* $x_M(a, b) > 1$ *if and only if* $1 < a < 3$ *or* $a > b + 2$.

In view of Lemma A.1, we partition $\mathcal{D}$ into the following four domains:

- $\mathcal{D}_{\rm I} = \{(a, b) \mid a > b + 2\}$.
- $\mathcal{D}_{\rm II} = \{(a, b) \mid \Gamma \le a \le b + 2,\ b \ge 1\}$.
- $\mathcal{D}_{\rm III} = \{(a, b) \mid 1 - b < a < \Gamma\}$.
- $\mathcal{D}_{\rm IV} = \{(a, b) \mid \Gamma \le a \le b + 2,\ b < 1\}$.

These four domains are illustrated by Figure A.1.

A.2 Exclusion of $\mathcal{D}_{\rm I}$, $\mathcal{D}_{\rm III}$, and $\mathcal{D}_{\rm IV}$. We show that domains $\mathcal{D}_{\rm I}$, $\mathcal{D}_{\rm III}$, and $\mathcal{D}_{\rm IV}$ can be immediately excluded for further considerations since, when (a, b) are in these domains, either the condition (A.1) is violated or there is no maximum value satisfying (A.2).

When $(a, b) \in \mathcal{D}_{\rm I}$, $g(x; a, b)$ has the critical points $x_m(a, b) < 1$ and $x_M(a, b) > 1$. By (A.3), we have $\partial_x g(1; a, b) < 0$. Noting that $g(1; a, b) = 1$, there must be an x such that $\ell < x \le 1$ and $g(x; a, b) > 1$. This violates the constraint (A.1). Hence, domain $\mathcal{D}_{\rm I}$ is excluded from further consideration.

When $(a, b) \in \mathcal{D}_{\rm III}$, $g(x; a, b)$ has no critical point. By (A.3), we have $\partial_x g(x; a, b) > 0$ for $x \in [0, 1]$, so $g(x; a, b)$ is strictly increasing on $[0, 1]$. In addition, $g(0; a, b) = 0$, and $g(1; a, b) = 1$. The condition (A.1) is satisfied. However, it follows from (A.5) that $h(a, b) = \min_{\ell \le x \le 1} g(x; a, b)$ is a strictly increasing function of a. Since $\mathcal{D}_{\rm III}$ is right-end open with respect to a, i.e., the boundary curve $a = \Gamma$ is not included,

$h(a,b)$ will not have a maximum on $\mathcal{D}_{\rm III}$.[3] Hence the domain $\mathcal{D}_{\rm III}$ can be removed from consideration.

Finally, when $(a,b) \in \mathcal{D}_{\rm IV}$, the critical points $x_m(a,b), x_M(a,b) > 1$. Similar to the discussion of domain $\mathcal{D}_{\rm III}$, we can show that $\partial_x g(x;a,b) > 0$ on $x \in [0,1]$ and that $g(0;a,b) = 0$ and $g(1;a,b) = 1$. Hence, the condition (A.1) is satisfied. By (A.5), $h(a,b) = \min_{l \le x \le 1} g(x;a,b)$ is a strictly increasing function of a. Since $\mathcal{D}_{\rm IV}$ includes the boundary line $a = b+2$, $h(a,b)$ has the maximum (with respect to a) only on the boundary line $a = b+2$. On the boundary line, $g(x;b+2,b)$ is an increasing function of b since $\partial_b g(x;b+2,b) > 0$. Hence, $H(b) = \min_{\ell \le x \le 1} g(x;b+2,b)$ is a strictly increasing function of b. However, $\mathcal{D}_{\rm IV}$ does not include the point $(a,b) = (3,1)$; therefore, $H(b)$ has no maximum. Consequently, domain $\mathcal{D}_{\rm IV}$ can be removed from consideration.

A.3 Searching on domain $\mathcal{D}_{\rm II}$. Let us focus on domain $\mathcal{D}_{\rm II}$. When $(a,b) \in \mathcal{D}_{\rm II}$, the critical points satisfy $x_m(a,b), x_M(a,b) \le 1$. (We define $x_M(a,b) = x_m(a,b)$ when $a = \Gamma$.) By (A.4), we have $\partial_{xx} g(x;a,b) \le 0$ at $x = x_m(a,b)$ and $\partial_{xx} g(x;a,b) \ge 0$ at $x = x_M(a,b)$, where both equalities hold only when $a = \Gamma$. Therefore, we have the following lemma.

LEMMA A.2. *When $(a,b) \in \mathcal{D}_{\rm II}$ and $a > \Gamma$, $g(x;a,b)$ has the local maximum at $x_m(a,b)$ and the local minimum at $x_M(a,b)$. When $(a,b) \in \mathcal{D}_{\rm II}$ and $a = \Gamma$, $g(x;a,b)$ is monotonically increasing on $[0,1]$.*

A.3.1 Further partition of $\mathcal{D}_{\rm II}$. To find the subregion $\mathcal{D}_{\rm II}^0$ of $\mathcal{D}_{\rm II}$ in which (A.1) is satisfied, let us further divide domain $\mathcal{D}_{\rm II}$ into two subdomains:

- $\mathcal{D}_{\rm II}^a = \{(a,b) \mid (a,b) \in \mathcal{D}_{\rm II} \text{ and } x_m(a,b) < \ell\}$.
- $\mathcal{D}_{\rm II}^b = \{(a,b) \mid (a,b) \in \mathcal{D}_{\rm II} \text{ and } x_m(a,b) \ge \ell\}$.

When $(a,b) \in \mathcal{D}_{\rm II}^a$, by Lemma A.2, we know that $g(x;a,b)$ does not have a local maximum on $[\ell,1]$. Since a differentiable function on a closed interval takes its maximum at either the endpoints or the local maximum, we have $\max_{\ell \le x \le 1} g(x;a,b) = \max\{g(\ell;a,b), g(1;a,b)\}$. Noting that $g(1;a,b) = 1$, we have the following lemma.

LEMMA A.3. *For $(a,b) \in \mathcal{D}_{\rm II}^a$, $g(\ell;a,b) \le 1$ is the necessary and sufficient condition to meet* (A.1).

We now show that the condition $g(\ell;a,b) \le 1$ is violated for (a,b) in a subset of $\mathcal{D}_{\rm II}^a$. Consider the case $g(\ell;a,b) = 1$. It implies that $a = b\ell + 1 + 1/\ell \equiv a_1(b)$. Let us further partition $\mathcal{D}_{\rm II}^a$ into two subdomains:

- $\mathcal{D}_{\rm II}^{a,1} = \{(a,b) \mid (a,b) \in \mathcal{D}_{\rm II}^a \text{ and } a \le a_1(b)\}$.
- $\mathcal{D}_{\rm II}^{a,2} = \{(a,b) \mid (a,b) \in \mathcal{D}_{\rm II}^a \text{ and } a > a_1(b)\}$.

When $(a,b) \in \mathcal{D}_{\rm II}^{a,1}$, by (A.5), $g(\ell;a,b)$ is a strictly increasing function of a. Since $g(\ell;a_1(b),b) = 1$, it follows that, for any $\Delta a \ge 0$, we have $g(\ell;a_1(b) - \Delta a, b) \le 1$. Using Lemma A.3 and noting that any point in $\mathcal{D}_{\rm II}^{a,1}$ can be written as $(a_1(b) - \Delta a, b)$ for some $\Delta a \ge 0$, it follows that, for $(a,b) \in \mathcal{D}_{\rm II}^{a,1}$, the condition (A.1) is met.

When $(a,b) \in \mathcal{D}_{\rm II}^{a,2}$, we have $g(\ell;a,b) > 1$, and so (A.1) is violated since $g(\ell;a_1(b)+\Delta a, b) > 1$ for any $\Delta a > 0$. Therefore, $\mathcal{D}_{\rm II}^{a,2}$ is excluded from further consideration.

Next consider $(a,b) \in \mathcal{D}_{\rm II}^b$. By Lemma A.2, $g(x;a,b)$ is increasing on $[\ell, x_m(a,b)]$, decreasing on $[x_m(a,b), x_M(a,b)]$, and increasing on $[x_M(a,b), 1]$. Therefore, it follows that $\max_{\ell \le x \le 1} g(x(a,b);a,b) = \max\{g(x_m(a,b);a,b), g(1;a,b)\}$. Noting that $g(1;a,b) = 1$, we have the following result.

[3] Here we are using a basic result from calculus that says a strictly increasing function $f(x)$ has no maximum value on a right-open interval.

LEMMA A.4. *For* $(a,b) \in \mathcal{D}_{\rm II}^b$, $g(x_m(a,b);a,b) \le 1$ *is the necessary and sufficient condition to meet* (A.1).

We show that the condition $g(x_m(a,b);a,b) \le 1$ is violated for (a,b) in a subset of $\mathcal{D}_{\rm II}^b$. Consider the case $g(x_m(a,b);a,b) = 1$, which implies $a = 2\sqrt{b}+1 \equiv a_2(b)$, which we get by solving $g(x_m(a,b);a,b) = 1$ and $\partial_x g(x_m(a,b);a,b) = 0$ for a. Let us first partition $\mathcal{D}_{\rm II}^b$ into two subdomains:

- $\mathcal{D}_{\rm II}^{b,1} = \{(a,b) \mid (a,b) \in \mathcal{D}_{\rm II}^b \text{ and } a \le a_2(b)\}$.
- $\mathcal{D}_{\rm II}^{b,2} = \{(a,b) \mid (a,b) \in \mathcal{D}_{\rm II}^b \text{ and } a > a_2(b)\}$.

By the same argument as the one we used to exclude domain $\mathcal{D}_{\rm II}^{a,2}$, we can show that (A.1) is satisfied when $(a,b) \in \mathcal{D}_{\rm II}^{b,1}$ and is violated when $(a,b) \in \mathcal{D}_{\rm II}^{b,2}$. Therefore, $\mathcal{D}_{\rm II}^{b,2}$ is excluded.

A.3.2 Characterization of $\mathcal{D}_{\rm II}^0$. By the above arguments we conclude that, only when $(a,b) \in \mathcal{D}_{\rm II}^0 = \mathcal{D}_{\rm II}^{a,1} \bigcup \mathcal{D}_{\rm II}^{b,1}$, the condition (A.1) is satisfied. We next identify the boundary of $\mathcal{D}_{\rm II}^0$. We first note that the line $a = b+2$ cannot be the boundary of $\mathcal{D}_{\rm II}^0$ since on the line, $\partial_x g(1;b+2,b) = 0$ and $\partial_{xx} g(1;b+2,b) > 0$, there exists x such that $\ell < x \le 1$ and $g(x;a,b) > 1$, which violates the the condition (A.1). Consequently, the boundary of $\mathcal{D}_{\rm II}^0$ consists of the following:

- $a = \Gamma$ and
- $a = a_1(b)$ and $x_m(a,b) < \ell$ or
- $a = a_2(b)$ and $x_m(a,b) \ge \ell$.

Basic algebra shows that $a_1(b) > \Gamma$ and $a_2(b) > \Gamma$ on $b \ge 1$, so $a = \Gamma$ is the left-side boundary of $\mathcal{D}_{\rm II}^0$. To determine the right-side boundary, we note that $a_1(b)$ is the tangent line of the curve $a_2(b)$ at $(\widehat{a},\widehat{b}) \equiv (\frac{2+\ell}{\ell}, \frac{1}{\ell^2})$. This also means that $x_m(\widehat{a},\widehat{b}) = \ell$. Furthermore, through basic algebra manipulation, we can verify that

(i) $\frac{d}{db} x_m(a_1(b),b) < 0$ on $b > 1/\ell^2$,
(ii) $\frac{d}{db} x_m(a_2(b),b) < 0$ on $b \ge 1$.

From the above facts, we conclude that the right-side boundary with respect to a of $\mathcal{D}_{\rm II}^0$ is $a = a_2(b)$ for $1 \le b \le \widehat{b}$ and $a = a_1(b)$ for $\widehat{b} > b$. Using (A.5), we see that any point on the left of this boundary satisfies (A.1), so we conclude that

$$\mathcal{D}_{\rm II}^0 = \{(a,b) \mid (\Gamma \le a \le a_2(b) \text{ and } 1 \le b \le \widehat{b}) \text{ and } (\Gamma \le a \le a_1(b) \text{ and } b > \widehat{b})\}. \tag{A.6}$$

The shaded region in Figure A.2 illustrates the region $\mathcal{D}_{\rm II}^0$ for the case $\ell = 0.4$.

A.3.3 Optimal solution on boundary of $\mathcal{D}_{\rm II}^0$. Now we need to consider only the region $\mathcal{D}_{\rm II}^0$, defined in (A.6). Recall that $h(a,b) = \min_{\ell \le x \le 1} g(x;a,b)$ is a strictly increasing function of a. Hence, for a fixed b, $h(a,b)$ can be maximized at only the right-side boundary. Therefore, the optimal solution will occur on the right-side boundary of $\mathcal{D}_{\rm II}^0$, i.e., on the curve $a = a_2(b)$ for $b \in [1,\widehat{b}]$ or the line $a = a_1(b)$ for $b \in (\widehat{b},\infty)$.

In fact, the line $a = a_1(b)$ for $b \in (\widehat{b},\infty)$ is removed from consideration. This is because $\partial_b g(x;a_1(b),b) < 0$ on $\ell < x < 1$ and $\min_{\ell \le x \le 1} g(x;a_1(b),b)$ is a strictly decreasing function of b and does not reach its maximum on the left-open interval $b \in (\widehat{b},\infty)$.

Now let us consider the curve $a = a_2(b)$ for $b \in [1,\widehat{b}]$. Rewriting $a = a_2(b) = 2\sqrt{b}+1$ as a function of a, we have

$$b_2(a) = (a-1)^2/4, \quad 3 \le a \le \widehat{a}. \tag{A.7}$$

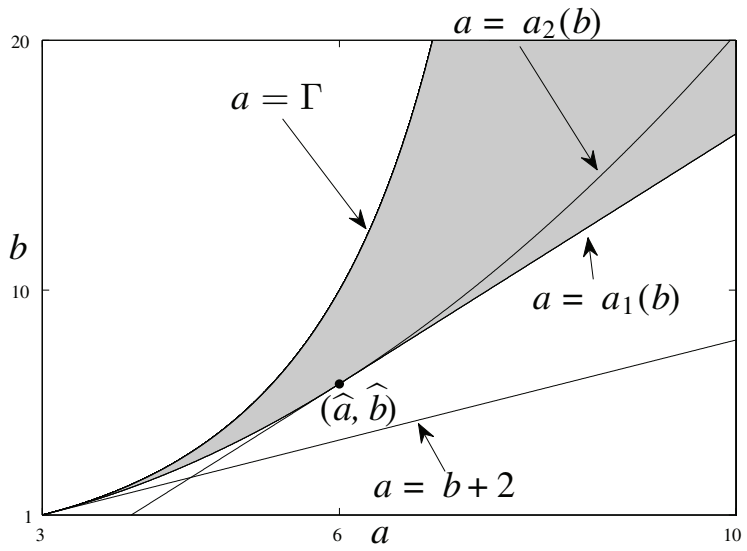

FIG. A.2. *Shaded region is* $\mathcal{D}_{\rm II}^0$ *for* $\ell = 0.4$.

By Lemma A.2 and the fact that $x_m(a,b) \geq \ell$ on the curve $a = a_2(b)$, we know that $g(x; a, b_2(a))$ is increasing on $x \in [\ell, x_m(a,b)]$, decreasing on $x \in [x_m(a,b), x_M(a,b)]$, and increasing again on $x \in [x_M(a,b), 1]$. It follows that

$$\min_{\ell \leq x \leq 1} g(x; a, b_2(a)) = \min\{s_1(a), s_2(a)\}, \tag{A.8}$$

where

$$s_1(a) \equiv g(\ell; a, b_2(a)) = \frac{\ell(4a + (a-1)^2\ell^2)}{4 + (a+3)(a-1)\ell^2},$$
$$s_2(a) \equiv g(x_M(a,b); a, b_2(a)) = \frac{4a^{3/2}}{(a+3)\sqrt{(a+3)(a-1)}}.$$

The following lemma is readily verified.

LEMMA A.5. *$s_1(a)$ is increasing and $s_2(a)$ is decreasing on $a \in [3, \widehat{a}]$. Furthermore, $s_1(3) \leq s_2(3)$, and $s_1(\widehat{a}) \geq s_2(\widehat{a})$.*

Lemma A.5 implies that there exists $a_* \in [3, \widehat{a}]$ such that

$$s_1(a_*) = s_2(a_*). \tag{A.9}$$

Solving (A.9) for a_* yields $a_* = h(\ell)$, where $h(\ell)$ is as defined in (3.15). Note that Lemma A.5 also implies that $\min_{\ell \leq x \leq 1} g(x; a, b_2(a))$ is increasing on $a \in [3, a_*]$ and decreasing on $a \in [a_*, \widehat{a}]$ with respect to a. Therefore, $\min_{\ell \leq x \leq 1} g(x; a, b_2(a))$ is maximized at $a = a_*$.

By (A.7), the optimal value of b is given by $b_* = \frac{1}{4}(a_* - 1)^2$. (a_*, b_*) attains the max-min in (A.2), and the value is given by

$$g(\ell; a_*, b_*) = \max_{a,b \in \mathcal{D}} \{ \min_{\ell \leq x \leq 1} g(x; a, b) \} = \frac{\ell(a_* + b_*\ell^2)}{1 + (a_* + b_* - 1)\ell^2}.$$

The max-min value $g(\ell; a_*, b_*)$ is used to update ℓ in (3.16). Finally, we note that if $\ell = 1$, the solution gives $a_* = 3$ and $b_* = 1$. In this case, the DWH iteration (3.3) and the Halley iteration (3.1) coincide.

Acknowledgments. We are grateful to Professor Hongguo Xu for sending us the unpublished work [5]. We thank Professor Nick Higham for his numerous comments and detailed suggestions on an early version of the manuscript and for sending us reference [7]. We thank the referees for their helpful suggestions and comments.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, 1999.

[2] T. A. Arias, M. C. Payne, and J. D. Joannopoulos, *Ab initio molecular-dynamics techniques extended to large-length-scale systems*, Phys. Rev. B, 45 (1992), pp. 1538–1549.

[3] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, *Minimizing Communication in Linear Algebra*, Technical report UCB/EECS-2009-62, Electrical Engineering and Computer Science, University of California, Berkeley, CA, 2009; also available as LAPACK Working Note 218.

[4] R. Byers and H. Xu, *A new scaling for Newton's iteration for the polar decomposition and its backward stability*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 822–843.

[5] R. Byers and H. Xu, unpublished, 2001.

[6] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368–375.

[7] S. Crudge, *The QR factorization and its applications*, Master's thesis, University of Manchester, 1998.

[8] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-avoiding parallel and sequential QR factorization*, Technical report UCB/EECS-2008-74, Electrical Engineering and Computer Science, University of California, Berkeley, CA, 2008.

[9] J.-L. Fattebert and F. Gygi, *Linear scaling first-principles molecular dynamics with controlled accuracy*, Comput. Phys. Comm., 162 (2004), pp. 24–36.

[10] W. Gander, *On Halley's iteration method*, Amer. Math. Monthly, 92 (1985), pp. 131–134.

[11] W. Gander, *Algorithms for the polar decomposition*, SIAM J. Sci. Comput., 11 (1990), pp. 1102–1115.

[12] S. L. Graham, M. Snir, and C. A. Patterson, eds., *Getting up to Speed, The Future of Supercomputing*, The National Academies Press, Washington, DC, 2005.

[13] F. Gygi, *Architecture of Qbox: A scalable first-principles molecular dynamics code*, IBM J. Res. Dev., 52 (2008), pp. 137–144.

[14] F Gygi, E. W. Draeger, M. Schulz, B. R. de Supinski, J. A. Gunnels, V. Austel, J. C. Sexton, F. Franchetti, S. Kral, C. W. Ueberhuber, and J. Lorenz, *Large-scale electronic structure calculations of high-Z metals on the Bluegene/L platform*, in Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, ACM, 2006.

[15] B. Hadri, H. Ltaief, E. Agullo, and J. Dongarra, *Tall and Skinny QR Matrix Factorization Using Tile Algorithms on Multicore Architectures*, Technical report UT-CS-09-645, University of Tennessee, Knoxville, TN, 2009; also available as LAPACK Working Note 222.

[16] W. W. Hager, *Condition estimates*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 311–316.

[17] D. J. Higham, *Condition numbers and their condition numbers*, Linear Algebra Appl., 214 (1995), pp. 193–213.

[18] N. J. Higham, *Computing the polar decomposition — with applications*, SIAM J. Matrix Anal. Appl., 7 (1986), pp. 1160–1174.

[19] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[20] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.

[21] N. J. Higham and P. Papadimitriou, *A parallel algorithm for computing the polar decomposition*, Parallel Comput., 20 (1994), pp. 1161–1173.

[22] C. Kenney and A. J. Laub, *On scaling Newton's method for polar decomposition and the matrix sign function*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 688–706.

[23] A. Kiełbasiński and K. Ziętak, *Numerical behaviour of Higham's scaled method for polar decomposition*, Numer. Algorithms, 32 (2003), pp. 105–140.

[24] A. Kiełbasiński and K. Ziętak, *Note on "A new scaling for Newton's iteration for the polar decomposition and its backward stability" by R. Byers and H. Xu*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1538–1539.
[25] B. Laszkiewicz and K. Ziętak, *Approximation of matrices and a family of Gander methods for polar decomposition*, BIT, 46 (2006), pp. 345–366.
[26] J. Nie, private communication, 2009.
[27] F. Song, A. YarKhan, and J. Dongarra, *Dynamic Task Scheduling for Linear Algebra Algorithms on Distributed-Memory Multicore Systems*, in Proceedings of the Conference on High Performance Computing, Networking, Storage and Analysis, ACM, 2009; also available as LAPACK Working Note 221.
[28] J. F. Sturm, *Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones*, Optim. Methods Softw., 11–12 (1999), pp. 625–653.
[29] Z. Zhang, H. Zha, and W. Ying, *Fast parallelizable methods for the Hermitian eigenvalue problem*, J. Comput. Math., 25 (2007), pp. 583–594.