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 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 6 iterations for convergence with the tolerance 10^{-16}. The Halley iteration can be implemented via QR decompositions without explicit matrix inversion. Therefore, it is an inverse free and a 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

1. Introduction. We consider the computation of the polar factor U of the polar decomposition of $A \in \mathbb{C}^{n\times n}$:

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

where U is an unitary matrix, $U^HU = I$ and H is a Hermitian positive semidefinite matrix. The polar decomposition is unique if A is nonsingular [20, Chap.8]. 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 principle molecular dynamics simulation of electronic structure calculations [2, 9, 13, 14].

The most popular method for computing the polar factor is the scaled Newton method [20]. Recently, Byers and Xu [5] 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 9 iterations for matrices with condition number no greater than 10^{16}. Furthermore, they proved that Newton's method with suboptimal scaling is backward stable, provided that the matrix inverses are computed in a mixed forward-backward stable way. The backward stability of Newton's method with Higham's $(1,\infty)$-norm scaling strategy was investigated in [23].

Successful as Newton's method is, it requires explicit matrix inversion at each iteration. Besides the potential numerical stability issue in finite precision arithmetic, 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, 26]. The purpose of this paper is to investigate numerical methods for computing the matrix polar decomposition to minimize the communication costs by using

*This work was partially supported by NSF grant OCI-0749217 and DOE grant DE-FC02-06ER25794.

† 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).

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, 4]. A QR decomposition based implementation of a variant of scaled Newton method is investigated. Unfortunately, the numerical instability of such an inverse free method has been independently discovered by both studies. We have also observed the numerical instability in our experiments shown in section 5.

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 6 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 illustrate that the QR-based DWH method, the QDWH method in short, is backward stable. The arithmetic cost of the QDWH method is about 2 to 3 times that of the scaled Newton method, depending on the implementation one uses. However, the communication cost of the QDWH method is significantly lower than that of the scaled Newton method. The QDWH method is an attractive alternative method for the emerging multi-core and heterogeneous computing architectures.

We note that in this paper we only study the polar decomposition of square and nonsingular matrices. The QDWH method is readily applicable to rectangular and singular matrices, whereas the scaled Newton method needs to initially use a rank-revealing QR factorization to enable its applicability to more generally 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 the cubic convergence makes acceptable a looser convergence criterion than that for the scaled Newton 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 in section 6

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 value 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. Newton's 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 all singular values $\sigma_i(X_k) \to 1$ as $k \to \infty$ [20, Thm 8.12]. However, the initial

phase of convergence is slow when A is ill-conditioned. In order to speed up the initial phase, we can apply the scaled Newton (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 $\zeta_k^{\text{opt}} = (\sigma_{\min}(X_k)\sigma_{\max}(X_k))^{-1/2}$ [22]. 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 [5] 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. [5]. *The iterates X_k generated by the scaled Newton iteration* (2.2) *with the BX scaling* (2.3) *converge quadratically to the polar factor U of A. Moreover, convergence to within a tolerance 10^{-16} is reached within 9 iterations if $\kappa_2(A) \le 10^{16}$.*

The following is a pseudo-code for the scaled Newton iteration with the BX scaling.

Scaled Newton's method:

$X_0 = A$ and $X_{-1} + I$.
$\zeta_0 = 1/\sqrt{\alpha\beta}$ and $k = 0$
repeat
  $X_{k+1} = (\zeta_k X_k + (\zeta_k X_k)^{-H})/2$
  if $k = 0$ **then**
    $\zeta_1 = \sqrt{2\sqrt{\alpha\beta}/(\alpha+\beta)}$
  else
    $\zeta_{k+1} = \sqrt{2/(\zeta_k + 1/\zeta_k)}$
  end if
  $k = k + 1$
until convergence
$U = X_k$

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 $\widehat{\beta} = 1/\|A^{-1}\|_F$. In fact, in [5] 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 9 iterations.

It is proved in [5] that the scaled Newton iteration is backward stable provided that the inverse X_k^{-1} is computed in a mixed forward-backward stable way, such as a bidiagonal reduction-based matrix inverse algorithm. In this case, the arithmetic cost of each iteration will increase to $6n^3$ instead of $2n^3$ when the inverse is computed using the conventional LU factorization with partial pivoting [5].

2.2. Newton iteration variant. The Newton iteration variant 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 $k \geq 1$, where X_k is generated by the Newton iteration (2.1) [21], [20, Sec.8.3]. Note that iteration (2.4) is applicable to singular and rectangular matrices.

To speed up the convergence, we have a scaled version of iteration (2.4). Substituting $\eta_k Y_k$ into Y_k in (2.4) yields the scaled Newton iteration variant (SNV):

$$Y_{k+1} = 2\eta_k Y_k \left(I + \eta_k^2 Y_k^H Y_k\right)^{-1}, \quad Y_0 = \eta_0 A, \tag{2.5}$$

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 scaled Newton 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 8.27 in [20, p.219]. One solution to the problem is to use the BX scaling (2.3). This leads to 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 BX 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 scaled Newton 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, 26]. As pointed out in [20, pp. 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 and great performance has been reported on modern multicore and heterogeneous systems [15]. Therefore, the QR-based SNV method is an attractive alternative. Unfortunately we have observed that SNV iteration (2.5) is not stable for ill-conditioned matrices, even with the QR-decomposition based implementation (see section 5 for numerical examples). This instability had been independently reported in early studies [7, 4]. 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 the asymptotic convergence rate is cubic [10, 11]. However, the initial steps of the Halley iteration (3.1) could be slow when A is ill-conditioned. 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, we have $x_1 \approx 3 \times 10^{-10}$. It takes 24 iterations for the iterate X_k to converge to the polar factor within IEEE double precision machine precision, namely $\|X_{24} - I\|_2 \le \epsilon_{\mathrm{M}} = 2.2 \times 10^{-16}$.

To accelerate the convergence of Halley's iteration (3.1), let us consider the following dynamically weighted Halley (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 the triplet (a_k, b_k, c_k) is properly chosen weighting parameters. To find an optimal weighting triplet (a_k, b_k, c_k) that accelerates the convergence rate at the $(k+1)$st iteration, let $X_k = U_* \Sigma_k V_*^H$ be the SVD of X_k and ℓ_k be such that

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

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

$$\begin{aligned} X_{k+1} &= X_k(a_k I + b_k X_k^H X_k)(I + c_k X_k^H X_k)^{-1} \\ &= U_* \Sigma_k V_*^H (a_k I + b_k V_* \Sigma_k^2 V_*^H)(I + c_k V_* \Sigma_k^2 V_*^H)^{-1} \\ &= U_* \Sigma_k (a_k I + b_k \Sigma_k^2)(I + c_k \Sigma_k^2)^{-1} V_*^H \\ &\equiv U_* \Sigma_{k+1} V_*^H. \end{aligned}$$

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 is measured by the maximum distance between singular values $\sigma_i(X_{k+1})$ and 1, an optimal 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 are in the interval $[\ell_{k+1}, 1]$:

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

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

Let us 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 $a_k + b_k > 1$,

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

and

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

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

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

Similar to the SN iteration (2.2) with the BX 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 following dynamically weighted Halley (DWH) iteration for computing the polar factor of A:

$$X_{k+1} = X_k(a_k I + b_k X_k^H X_k)(I + c_k X_k^H X_k)^{-1}, \quad X_0 = A/\alpha. \tag{3.15}$$

where the weighting parameters a_k and b_k are generated by the scalar iterations (3.14),

$$c_k = a_k + b_k - 1$$

and

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

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

Before we prove the global convergence property 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 closer to the target value 1 than the first iterate computed by Halley's iteration. In fact, it only takes 5 DWH iterations to approximate the polar factor within the machine precision $\|X_5 - I\|_2 \leq \epsilon_{\mathrm{M}}$. It is a factor of 5 times faster than the Halley iteration (3.1).

THEOREM 3.1. *The iterates X_k generated by the DWH iteration* (3.3) *converge cubically to the polar factor U of A.*

PROOF. Let us first show the convergence of the iterates X_k. This is equivalent to showing that singular values $\sigma_i^{(k)} \to 1$ as $k \to \infty$ for all $1 \leq i \leq n$, where $\sigma_i^{(k)}$ denotes the ith singular value of X_k. By (3.9), we have $[\sigma_{\min}^{(k)}, \sigma_{\max}^{(k)}] \subseteq [\ell_k, 1]$. 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 $F(a, \ell) \geq 0$ since $a > 0$. All we need to show is that there is a positive constant $\delta < 1$, such that $F(a_k, \ell_k) \leq \delta$ for all k. In fact, since $3 \leq a \leq \frac{2+\ell}{\ell}$ (see (6.8) in Appendix A), $F(a, \ell)$ is a decreasing function of a:

$$\frac{\partial F}{\partial a} = \frac{4(1+\ell)(\ell^2(a-1)^2 - 4)}{(\ell^2 + a^2\ell^2 + 2a(2-\ell^2))^2} \leq 0.$$

Therefore, we have

$$F(a, \ell) \leq F(3, \ell) = \frac{(3-1)^2\ell^2 - 4(3-1)\ell + 4}{(3-1)^2\ell^2 + 4 \cdot 3} = \frac{(1-\ell)^2}{\ell^2 + 3} \leq \frac{1}{3} = \delta.$$

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

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

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

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

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

$$3 \leq a_k \leq \frac{2 + \ell_k}{\ell_k} \qquad \text{for} \quad k \geq 0.$$

Hence 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).

REMARK 2. For the 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 following inclusion property holds:

$$[\sigma_{\min}(A/\widehat{\alpha}), \sigma_{\max}(A/\widehat{\alpha})] \subseteq [\widehat{\ell}_0, 1],$$

where $\widehat{\ell}_0 = \widehat{\beta}/\widehat{\alpha}$.

REMARK 3. In [11], Gander has observed the slow convergence with respect to the small singular values in Halley's iteration. He proposed a choice of static 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 [24]. In practice, Gander [11] suggests taking $\tau = 2 + \epsilon_{\mathrm{M}}/\delta$ for $\delta > 10\epsilon_{\mathrm{M}}$ and $\tau = 2.1$ for $\epsilon_{\mathrm{M}} < \delta \leq 10\epsilon_{\mathrm{M}}$, 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 $\tau = 2.1$.

Gander's iteration switches from iteration (3.15) 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 $\tau = 2.1$. In summary, Gander's method is as follows.

Gander's method:

$X_0 = A/\alpha$, $\ell_0 = \beta/\alpha$
$k = 0$, $\tau = 2.1$, $s = -\log(\ell_0)$
$a = (2\tau - 3)/(\tau - 2)$, $b = 1/(\tau - 2)$ and $c = \tau/(\tau - 2)$
while $k < s$ **do**
  $X_{k+1} = X_k(aI + bX_k^H X_k)(I + cX_k^H X_k)^{-1}$
  $k = k + 1$
end while
repeat
  $X_{k+1} = X_k(3I + X_k^H X_k)(I + 3X_k^H X_k)^{-1}$
  $k = k + 1$
until convergence
$U = X_k$

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 (3.15), we note that once convergence has set 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 first have the following Lemma.

LEMMA 3.2. *For Halley's iteration* (3.1)*, if* $\|X_k - U\|_2 = \|I - \Sigma_k\|_2 = \epsilon < 2/3$*, then up to first order of* ϵ*,*

$$\frac{2-3\epsilon}{(1+\epsilon)(2+\epsilon)}\|X_{k+1} - X_k\| \le \|X_k - U\| \le \frac{2+3\epsilon}{(1-\epsilon)(2-\epsilon)}\|X_{k+1} - X_k\|.$$

PROOF. Writing $X_k = U_*\Sigma_k V_*$ we have

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

Taking an unitarily invariant norm and using $\|AB\| \le \|A\| \cdot \|B\|_2$ [20, Sec B.7], we get

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

Recalling that $\Sigma_k = \mathrm{diag}\{\sigma_1, \dots, \sigma_n\}$ and $\|I - \Sigma_k\|_2 = \max_i |1 - \sigma_i| = \epsilon$, we have

$$1 - \epsilon \le \|\Sigma_k\|_2 (= \max_i \sigma_i) \le 1 + \epsilon, \quad 2 - \epsilon \le \|I + \Sigma_k\|_2 (= \max_i (1 + \sigma_i)) \le 2 + \epsilon,$$

and up to first order in ϵ, $1/(4+6\epsilon) \le \|(I+3\Sigma_k^2)^{-1}\|_2 (= \max_i 1/(1+3\sigma_i^2)) \le 1/(4-6\epsilon)$. Using these three inequalities we get

$$\|\Sigma_k(I + \Sigma_k)(I + 3\Sigma_k^2)^{-1}\|_2 \le \frac{(1+\epsilon)(2+\epsilon)}{4-6\epsilon}.$$

Plugging this into (3.19) yields the lower bound in the result. The upper bound is obtained similarly by noticing from (3.18) that $X_k - U = (X_{k+1} - X_k)(V_*(I + \Sigma_k)\Sigma_k(I + 3\Sigma_k^2)^{-1}V_*^H)^{-1}/2$ and taking norms. □

Note that Lemma 3.2 implies

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

Hence when $\epsilon = \|X_k - U\| \ll 1$, we have

$$\|X_{k+1} - X_k\| \simeq \|X_k - U\|. \tag{3.20}$$

Now, by $U_*(\Sigma_k - I)V_*^H = X_k - U$, we have

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

where we used the inequality $\|AB\| \le \|A\| \cdot \|B\|_2$ again. Now, close to convergence $\|X_k - U\|_2 \ll 1$, by (3.20) we have

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

This suggests that we accept X_{k+1} when

$$\|X_{k+1} - X_k\|_F \leq \left(\frac{\epsilon_{\mathrm{M}}}{\|(I + 3X_k^H X_k)^{-1}\|_2} \right)^{1/3}. \tag{3.21}$$

Close to convergence $X_k^H X_k \simeq I$, so the test (3.21) is effectively

$$\|X_{k+1} - X_k\|_F \leq (4\epsilon_{\mathrm{M}})^{1/3}. \tag{3.22}$$

For the quadratically convergent methods such as scaled Newton's method (2.2) and its variant (2.5), the following stopping criterion is suggested in [20, Sec.8.7]:

$$\|X_{k+1} - X_k\|_F \leq (2\epsilon_{\mathrm{M}})^{1/2}. \tag{3.23}$$

In [5] it is noted that theoretically the scaled Newton iteration with the BX scaling converges in at most 9 steps for any matrices of condition number less than 10^{16}. The basic observation in [5] is that it can be shown that $\|X_k - U\|_2 \leq b_k$, where b_k is obtained by a scalar iteration that can be computed easily. Hence finding the first k for which $|1 - b_k| < 10^{-16}$ provides an upper bound on the number of iteration counts.

We can derive a similar result for the DWH iteration (3.15). By the interval (3.9) that bounds the singular values of the iterate X_k, we have

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

Hence, by finding the first k such that $|1 - \ell_k| < 10^{-16}$, we derive 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 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 6 steps for any matrix with the condition number $\kappa_2(A) \leq 10^{16}$. The number of DWH iterations is about one third less than the number of scaled Newton iterations (2.2) with the BX scaling (2.3).

4. QR-based implementations. In this section, we examine an implementation of the DWH iteration (3.15) using the QR decomposition. The QR-based implementation is more desirable than those involving explicit inverses for enhancing parallelizability. Numerical examples suggest that it also improves the numerical stability, see section 5.

First, we have the following basic result to connect the QR decomposition and the term $X(I+\eta^2 X^H X)^{-1}$ that appears in the SNV iteration (2.5) and DWH (3.15).

THEOREM 4.1. [28],[20, p.219]. *Let* $\begin{bmatrix} \eta X \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$ *be a QR decomposition of* $\begin{bmatrix} \eta X \\ I \end{bmatrix}$. *Then*

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

PROOF. By the polar decomposition

$$\begin{bmatrix} \eta X \\ I \end{bmatrix} = UH, \tag{4.2}$$

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

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = UW \tag{4.3}$$

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):

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

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

QSNV algorithm:

$X_0 = A$, $\eta_0 = 1/\sqrt{\alpha\beta}$, $k = 0$
$k = 0$
repeat
  compute QR decomposition $\begin{bmatrix} \eta_k X_k \\ I \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$
  $X_{k+1} = 2 Q_1 Q_2^H$
  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
$U = X_k$

Similarly, for the DWH iteration (3.15), we first note that iteration (3.15) can be equivalently rewritten as

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

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:

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

The following is a pseudo-code 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 practical implementation, we need to provide estimates $\widehat{\alpha}$ and $\widehat{\beta}$ of the largest and smallest singular values α and β of the original matrix A for the QSNV and QDWH methods. We can simply use $\widehat{\alpha} = \|A\|_F \geq \alpha = \|A\|_2$. However, 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 [5]. However this is not practical for the QSNV and QDWH methods since A^{-1} is not calculated explicitly. By the inequality $\|A\|_1/\sqrt{n} \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$, we have $\beta = \sigma_{\min}(A) = \|A^{-1}\|_2^{-1} \geq (\sqrt{n}\|A^{-1}\|_1)^{-1}$. 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 choice of $\widehat{\beta}$ on the convergence of the QDWH method. The numerical examples suggest that it is harmless to use a rough estimate of $\widehat{\beta}$ ensuring that $\widehat{\ell_0} = \widehat{\beta}/\widehat{\alpha}$ is a lower bound of $\sigma_{\min}(X_0)$.

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, the cost is $2n^3$ flops. Therefore, the arithmetic cost of each QDHW iteration is $\frac{26}{3}n^3$ flops. Since it generally takes at most 6 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 9 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.

To guarantee backward stability of the SN iteration, the matrix inversion needs to be calculated using a bidiagonal reduction-based matrix inversion algorithm as described in [5]. 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, then carefully reduce the augmented matrix into a triangular form. In this way, the cost of per QDWH iteration is reduced to $(16/3)n^3$ flops. Thus the cost of 6 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. Let $\widehat{U}$ be a computed polar factor of A. Then the Hermitian factor of the polar decomposition is computed by $\widehat{H} = \frac{1}{2}(\widehat{U}^H A + (\widehat{U}^H A)^H)$. The accuracy of the computed polar decomposition is tested by the residual norm $\text{res} = \|A - \widehat{U}\widehat{H}\|_F / \|A\|_F$. A method is said to have behaved in a numerically backward stable way when the residual norm is smaller than $c\epsilon_{\text{M}}$ for a moderate constant c, where ϵ_{M} is the machine precision [20, p.209].

All numerical experiments were performed in MATLAB 7.4.0 and run on a PC with a Core 2 Duo processor. The machine epsilon $\epsilon_{\text{M}} \simeq 2.2 \times 10^{-16}$. The stopping criterion (3.22) 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 (3.23) is applied. Since A^{-1} is computed explicitly in the scaled Newton 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).

Example 1. In this example, we show the effectiveness of the dynamical weighting. 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 shows 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 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 6 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 $\Sigma = \text{diag}\{10^8, 1, 10^{-8}\}$,

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

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, 4]. This numerical observation led us to give up the QSNV, the natural candidate for an inverse-free iteration, and turned 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 show the numerical stability and convergence rate of the QDWH method, and compare with the SN method. The bidiagonal reduction-based matrix inversion method is used in the SN method to guarantee the numerical backward stability. Test matrices were generated as follows. We construct 3 groups of 20×20 matrices using the MATLAB function `gallery('randsvd',20,kappa)`, where the condition number `kappa` is set to be $10^2, 10^8, 10^{15}$, 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	2.9e-16	6.7e-16	2.8e-16	7.1e-16
	SN	4.3e-16	6.5e-16	3.1e-16	6.1e-15	3.4e-16	1.2e-15

We observe that both SN and QDWH methods exhibit excellent numerical stability. The QDWH method needs about 2/3 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 6 iterations and performed in a backward stable manner.

Example 4. In this example, we investigate the impact of the choice of estimates $\widehat{\alpha}$ and $\widehat{\beta}$ of extreme singular values $\alpha = \sigma_{\max}(A)$ 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). Hence we focus on the effect of misestimating β, or equivalently, $\ell_0 = \beta/\alpha$. Let $A \in \mathbb{R}^{20\times 20}$ be generated by using `randsvd` as in Example 3 with $\kappa_2(A) = 10^8$, and let $\widehat{\alpha} = \|A\|_F$ and $X_0 = A/\widehat{\alpha}$. The following table shows the convergence rate and stability of the QDWH method with the estimates $\widehat{\ell}_0$ from $10^{-9}\sigma_{\min}(X_0)$, severely underestimated, to $10^{9}\sigma_{\min}(X_0)$, severely overestimated:

$\widehat{\ell}_0/\sigma_{\min}(X_0)$	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 taking $\widehat{\ell}_0$ too large slows down the convergence substantially, but taking small $\widehat{\ell}_0$ 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 safe 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 is an inverse free method and is based on QR decompositions. Extensive numerical results show that the QDWH performed in the same backward stable way as the the scaled Newton method. The QDWH method is more expensive in arithmetic cost than the scaled Newton iteration with LU-based inversions, and is about the same if the scaled Newton iteration is implemented using the bidiagonal reduction-based matrix inversions. The QR-based implementation makes the QDWH a communication-friendly algorithm, which is desirable for the emerging multicore and heterogeneous computing systems. 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 the solution of the optimization problem (3.12) and (3.13), restated 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 $0 < \ell \leq 1$. Find $(a_*, b_*) \in \mathcal{D}$ such that

$$0 < g(x;a_*,b_*) \leq 1 \quad \text{for} \quad \ell \leq x \leq 1, \tag{6.1}$$

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

$$\max_{(a,b)\in\mathcal{D}} \left\{ \min_{\ell\leq x\leq 1} g(x;a,b) \right\}. \tag{6.2}$$

We note that in [25], Nie describes a scheme to reformulate the problem as a standard semidefinite programming (SDP) problem so that we can solve it by using a SDP software, such as SeDuMi [27]. In this appendix, we provide an analytic solution of the problem.

A.1 Properties of function g and 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 derivative of $g(x;a,b)$ with respect to x are

$$\partial_x g(x;a,b) = \frac{b(a+b-1)x^4 - (a(a+b-1)-3b)x^2 + a}{(1+(a+b-1)x^2)^2} \tag{6.3}$$

and

$$\partial_{xx} g(x;a,b) = \frac{2(a-1)(a+b)x((a+b-1)x^2-3)}{(1+(a+b-1)x^2)^3}. \tag{6.4}$$

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

$$\partial_a g(x;a,b) = \frac{x(1-x^2)(1+bx^2)}{(1+(a+b-1)x^2)^2}. \tag{6.5}$$

Note 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.

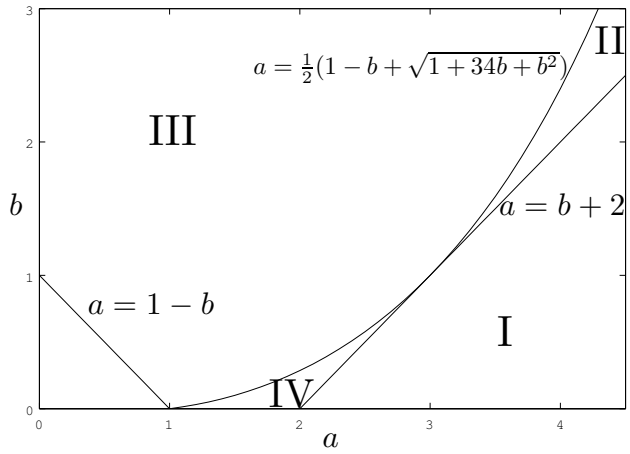

FIG. 6.1. *Partition of domain* $\mathcal{D}$.

LEMMA 6.1. *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* $0 < x_m(a, b) \le x_M(a, b)$. *Otherwise,* $g(x; a, b)$ *has no real critical points. Furthermore,* $x_m(a, b) > 1$ *if and only if* $1 < a < 3$ *and* $b > a - 2$, *and* $x_M(a, b) > 1$ *if and only if* $1 < a < 3$ *or* $b < a - 2$.

In view of Lemma 6.1, 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 6.1.

A.2 Excluding domains $\mathcal{D}_{\rm I}$, $\mathcal{D}_{\rm III}$ 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 (6.1) is violated or there is no maximum value satisfying (6.2).

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 (6.3), we have $\partial_x g(1; a, b) < 0$. Noting that $g(1; a, b) = 1$, there must be a x such that $\ell < x \le 1$ and $g(x; a, b) > 1$. This violates the constraint (6.1). 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 (6.3), 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, $g(0; a, b) = 1$ and $g(1; a, b) = 1$. Therefore, the condition (6.1) is satisfied. However, it follows from (6.5) that $h(a, b) = \min_{\ell \le x \le 1} g(x; a, b)$ is a strictly increasing function of a. Note that $\mathcal{D}_{\rm III}$ is right-end open with respect to a, i.e., the boundary curve $a = \Gamma$ is not included. Therefore, $h(a, b)$ will not have a maximum on $\mathcal{D}_{\rm III}$.[1] Hence domain $\mathcal{D}_{\rm III}$ can be removed from consideration.

Finally, when $(a, b) \in \mathcal{D}_{\rm IV}$, the critical points satisfy $x_m(a, b), x_M(a, b) > 1$. 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 $g(0; a, b) = 1$ and $g(1; a, b) = 1$. Hence the condition (6.1) is satisfied. By (6.5),

[1] 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.

$h(a,b) = \min_{l \le x \le 1} g(x;a,b)$ is a strictly increasing function of a. Note that $\mathcal{D}_{\rm IV}$ includes the boundary line $a = b+2$. Therefore, $h(a,b)$ has the maximum (w.r.t. a) only at 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, since $\mathcal{D}_{\rm IV}$ does not include the point $(a,b) = (3,1)$, $H(b)$ 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 $x_m(a,b), x_M(a,b) \le 1$. By (6.4), we have $\partial_{xx} g(x;a,b) < 0$ at $x = x_m(a,b)$ and $\partial_{xx} g(x;a,b) > 0$ at $x = x_M(a,b)$. Therefore, we have

LEMMA 6.2. *When $(a,b) \in \mathcal{D}_{\rm II}$, $g(x;a,b)$ has the local maximum at $x_m(a,b)$, and the local minimum at $x_M(a,b)$, respectively.*

A.3.1 Further partition of $\mathcal{D}_{\rm II}$. To examine the condition under which (6.1) 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 6.2, 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

LEMMA 6.3. *For $(a,b) \in \mathcal{D}_{\rm II}^a$, $g(\ell;a,b) \le 1$ is the necessary and sufficient condition to meet* (6.1).

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 $a = b - \ell - 1/l \ \equiv a_1(b)$. Let us partition $\mathcal{D}_{\rm II}^a$ into two subdomains; one is on the left of the line $a_1(b)$, including on the line, and the other is on the right of the line $a_1(b)$:

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

When $(a,b) \in \mathcal{D}_{\rm II}^{a,1}$, by (6.5), $g(\ell;a,b)$ 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 6.3 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 the condition (6.1) is met.

When $(a,b) \in \mathcal{D}_{\rm II}^{a,2}$, we have $g(\ell;a,b) > 1$ and so (6.1) is violated since $g(\ell;a_1(b) + \Delta a, b) > 1$ for any $\Delta a > 0$. Therefore, domain $\mathcal{D}_{\rm II}^{a,2}$ can be excluded for further consideration.

Next consider $(a,b) \in \mathcal{D}_{\rm II}^b$. By Lemma 6.2, $g(x;a,b)$ 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.

LEMMA 6.4. *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* (6.1).

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 $a = 2\sqrt{b} + 1 \ \equiv a_2(b)$. Let us partition $\mathcal{D}_{\rm II}^b$ into two subdomains: one is on the left of the curve $a_2(b)$, including the curve, and the other is on the right of the curve $a_2(b)$:

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

By the same argument as the one we used to exclude domain $\mathcal{D}_{\rm II}^{a,2}$, we can show that (6.1) is satisfied in $\mathcal{D}_{\rm II}^{b,1}$ but not in $\mathcal{D}_{\rm II}^{b,2}$. Therefore $\mathcal{D}_{\rm II}^{b,2}$ can be excluded.

A.3.2 Searching optimal solution in $\mathcal{D}_{\rm II}^{a,1}$ and $\mathcal{D}_{\rm II}^{b,1}$. For $(a,b) \in \mathcal{D}_{\rm II}^{a,1}$, by (6.5) $h(a,b) = \min_{\ell \le x \le 1} g(x;a,b)$ is a strictly increasing function of a. Noting that $\mathcal{D}_{\rm II}^{a,1}$ has a closed boundary line with respect to a, it follows that $h(a,b)$ can take the maximum (w.r.t. a) only on the (boundary) line $a_1(b)$.

Similarly for $(a,b) \in \mathcal{D}_{\rm II}^{b,1}$, using (6.5) we see that $h(a,b)$ can take the maximum (w.r.t. a) only at the (boundary) curve $a_2(b)$.

In addition, we observe 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 $x_m = \ell$ at this point. Furthermore, we can verify that
(i) $\partial_b x_m(a,b) < 0$ along $a_1(b)$.
(ii) $\partial_b x_m(a,b) < 0$ along $a_2(b)$.
By (i) we conclude that the right-side boundary of $\mathcal{D}_{\rm II}^{a,1}$ is $a = a_1(b)$ on $b \in (\widehat{b}, \infty)$. Similarly, by (ii), the right-side boundary of $\mathcal{D}_{\rm II}^{b,1}$ is $a = a_2(b)$ on $b \in [1, \widehat{b}]$. Moreover, it is easy to show that $\partial_b g(x; a_1(b), b) < 0$ on $\ell < x < 1$. Therefore $\min_{\ell \le x \le 1} g\,(x; a_1(b), b)$ is a strictly decreasing function of b, so it does not reach its maximum on the left-open interval $b \in (\widehat{b}, \infty)$. Hence the segment of the curve $a_1(b)$ for $b \in (\widehat{b}, \infty)$ can be removed from consideration.

Thus, we only need to focus on the segment of the curve $a = a_2(b)$ for $b \in [1, \widehat{b}]$. Write the function $a = a_2(b)$ as a function of a:

$$b_2(a) = (a-1)^2/4, \quad 3 \le a \le \widehat{a}. \tag{6.6}$$

Note that the function $g(x; a, b_2(a))$ is increasing on $x \in [\ell, x_m(a,b)]$, decreasing on $x \in [x_m(a,b), x_M(a,b)]$ and increasing on $x \in [x_M(a,b), 1]$. Hence it follows that $\min_{\ell \le x \le 1} g(x; a, b_2(a))$ is taken either at the endpoint $x = \ell$ or at the local minimum $x = x_M(a,b)$:

$$\min_{\ell \le x \le 1} g(x; a, b_2(a)) = \min\{s_1(a), s_2(a)\}, \tag{6.7}$$

where

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

The following Lemma is readily verified.

LEMMA 6.5. *$s_1(a)$ is an increasing function, and $s_2(a)$ is a decreasing function of a on $a \in [3, \widehat{a}]$. In addition, $s_1(3) \le s_2(3)$ and $s_1(\widehat{a}) \ge s_2(\widehat{a})$.*

Lemma 6.5 implies that there exists $a_* \in [3, \widehat{a}]$ such that

$$s_1(a_*) = s_2(a_*). \tag{6.8}$$

Solving (6.8) for a_* yields

$$a_* = \sqrt{1+d} + \frac{1}{2}\sqrt{8 - 4d + \frac{8(2-\ell^2)}{\ell^2\sqrt{1+d}}}, \quad \text{where} \quad d = \sqrt[3]{\frac{4(1-\ell^2)}{\ell^4}}.$$

Note that Lemma 6.5 also implies that $\min_{\ell\le x\le 1} g(x; a, b_2(a))$ is increasing on $a \in [3, a_*]$ and decreasing on $a \in [a_*, \widehat{a}]$ with respect to a. Therefore, $\min_{\ell\le x\le 1} g(x; a, b_2(a))$ is maximized at $a = a_*$. By (6.6), the optimal value of b is given by $b_* = \frac{1}{4}(a_* - 1)^2$.

Consequently, (a_*, b_*) attains the max-min in (6.2) and

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

which is used to update ℓ as in (3.16). We note that if $\ell = 1$, the solution gives $a_* = 3$ and $b_* = 1$, which is consistent because plugging this into (3.3) results in the Halley iteration (3.1).

Acknowledgments. We are grateful to Dr. Hongguo Xu for sending us his unpublished work [4]. We thank Nick Higham for his numerous comments and detailed suggestions on an early version of the manuscript and for sending us the reference [7].

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*. SIAM, Philadelphia, PA, third edition, 1999.

[2] T. A. Arias, M. C. Payne, and J. D. Joannopoulos. Ab initio molecular-dynamics techniques extended to large-length-scale systems. *Physical Review B*, 45(4):1538–1549, 1992.

[3] Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Minimizing Communication in Linear Algebra. Technical Report 218, LAPACK Working Note, May 2009.

[4] Ralph Byers and Hongguo Xu. An inverse free method for the polar decomposition, 2001. Unpublished note.

[5] Ralph Byers and Hongguo Xu. A new scaling for Newton's iteration for the polar decomposition and its backward stability. *SIAM J. Matrix Anal. Appl.*, 30:822–843, 2008.

[6] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. Estimate for the condition number of a matrix. *SIAM J. Numer. Anal.*, 16(2):368–375, 1979.

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

[8] J. Demmel, L. Grigoir, M. Hoemmen, and J. Langou. Communication-avoiding parallel and sequential QR factorization. *Technical Report No. UCB/EECS-2008-74, Electrical Engineering and Computer Science, UC Berkeley*, May 2008.

[9] Jean-Luc Fattebert and François Gygi. Linear scaling first-principles molecular dynamics with controlled accuracy. *Computer Physics Communications*, 162(1):24–36, 2004.

[10] Walter Gander. On Halley iteration method. *American Mathematical Monthly*, 92(2):131–134, 1985.

[11] Walter Gander. Algorithms for the polar decomposition. *SIAM J. Sci. Comp*, 11(6):1102–1115, 1990.

[12] Susan L. Graham, Marc Snir, and Cynthia A. Patterson (eds.). *Getting up to speed, the future of supercomputing*. The National Academies Press, 2005.

[13] F. Gygi. Architecture of Qbox: A scalable first-principles molecular dynamics code. *IBM Journal of Research and Development*, 52(1-2):137–144, 2008.

[14] François Gygi, Erik W. Draeger, Martin Schulz, Bronis R. de Supinski, John A. Gunnels, Vernon Austel, James C. Sexton, Franz Franchetti, Stefan Kral, Christoph W. Ueberhuber, and Juergen Lorenz. Large-scale electronic structure calculations of high-z metals on the bluegene/l platform. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 45, NY, USA, 2006. ACM.

[15] Bilel Hadri, Hatem Ltaief, Emmanuel Agullo, and Jack Dongarra. Tall and Skinny QR Matrix Factorization Using Tile Algorithms on Multicore Architectures. Technical Report 222, LAPACK Working Note, September 2009.

[16] William W. Hager. Condition estimators. *SIAM Journal on scientific and statistical computing*, 5(2):311–316, 1984.

[17] Desmond J. Higham. Condition numbers and their condition numbers. *Linear Algebra and its Applications*, 214:193–213, 1995.

[18] Nicholas J. Higham. Computing the polar decomposition - with applications. *SIAM J. Matrix Anal. Appl.*, 7(4):1160–1174, 1986.

[19] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, USA, second edition, 2002.
[20] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, PA, USA, 2008.
[21] Nicholas J. Higham and Pythagoras Papadimitriou. A parallel algorithm for computing the polar decomposition. *Parallel computing*, 20(8):1161–1173, 1994.
[22] Charles Kenney and Alan J. Laub. On scaling Newton's method for polar decomposition and the matrix sign function. *SIAM J. Matrix Anal. Appl.*, 13(3):688–706, 1992.
[23] Andrzej Kiełbasiński and Krystyna Ziętak. Numerical behaviour of Higham's scaled method for polar decomposition. *Numerical Algorithms*, 32(2-4):105–140, 2003.
[24] B. Laszkiewicz and K. Ziętak. Approximation of matrices and a family of Gander methods for polar decomposition. *BIT*, 46(2):345–366, 2006.
[25] Jiawang Nie. A solution to rational optimization. *Private Communication*, 2009.
[26] Fengguang Song, Asim YarKhan, and Jack Dongarra. Dynamic Task Scheduling for Linear Algebra Algorithms on Distributed-Memory Multicore Systems. Technical Report 221, LAPACK Working Note, April 2009.
[27] J. F. Sturm. SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11&12:625–653, 1999.
[28] Hongyuan Zha and Zhenyue Zhang. Fast parallelizable methods for the Hermitian eigenvalue problem. Technical Report CSE-96-041, Department of Computer Science and Engineering, Pennsylvania State University, May 1996.