Numer Algor
DOI 10.1007/s11075-016-0098-7

ORIGINAL PAPER

An algorithm to compute the polar decomposition of a 3×3 matrix

Nicholas J. Higham[1] · **Vanni Noferini**[2]

Received: 29 September 2015 / Accepted: 8 January 2016

Abstract We propose an algorithm for computing the polar decomposition of a 3×3 real matrix that is based on the connection between orthogonal matrices and quaternions. An important application is to 3D transformations in the level 3 Cascading Style Sheets specification used in web browsers. Our algorithm is numerically reliable and requires fewer arithmetic operations than the alternative of computing the polar decomposition via the singular value decomposition.

Keywords Polar decomposition · 3×3 matrix · Singular value decomposition · Numerical stability · Quaternions

This work was supported by European Research Council Advanced Grant MATFUN (267526) and Engineering and Physical Sciences Research Council grant EP/I03112X/1

Electronic supplementary material The online version of this article (doi: 10.1007/s11075-016-0098-7) contains supplementary material, which is available to authorized users.

✉ Vanni Noferini
vnofer@essex.ac.uk
URL: http://www.maths.manchester.ac.uk/∼noferini

Nicholas J. Higham
nick.higham@manchester.ac.uk
URL: http://www.maths.manchester.ac.uk/∼higham

[1] School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK

[2] Department of Mathematical Sciences, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK

1 Introduction

In designing algorithms for dense numerical linear algebra problems we are usually driven by the need to make the algorithms efficient for large values of the dimension, n. This leads to certain design principles, such as to prefer an $O(n^3)$ flops algorithm to an $O(n^4)$ flops one and to aim to develop algorithms that are rich in matrix–matrix operations. However, there is another class of problems that is rarely considered, though practically important: to solve a large number of very small problems.

For small problems ($2 \leq n \leq 8$, say), the exponent and constant of the highest order term in the flop count are not necessarily indicative of the cost, since that term may not dominate, and the prevalence of matrix–matrix or matrix–vector operations is also of little relevance. These problems therefore need a different approach.

In this work we are concerned with computing the polar decomposition of a 3×3 matrix. This problem arises in a number of applications, but the main motivation for our work comes from 3D graphics transformations, where the polar decomposition provides a convenient way to parametrize and interpolate transformations [22, 28]. In particular, the polar decomposition is of interest for the level 3 specification of the Cascading Style Sheets (CSS) language that is under development by the World Wide Web Consortium (W3C) [31, sect. 20.3]. The requirements are for an implementation of the polar decomposition that can run in any web browser on any computing platform (desktop and mobile), runs fast enough to support animation, never fails, is numerically stable, and has minimal dependence on libraries [27].

Other applications requiring the polar decomposition of a 3×3 matrix are in classical continuum mechanics [3], in orthogonalization of approximate direction cosine matrices in aerospace systems [1, 10, 18], and for modeling soft tissue deformation in virtual surgery [32].

An obvious question is whether we can compute the polar decomposition of a 3×3 matrix explicitly. Formulae are available for the 2×2 case [14, 29], and for companion matrices [30]. In [16] it is explained how to obtain analytic formulae for the eigendecomposition of a symmetric 3×3 matrix. These yield complicated formulae for the singular value decomposition (SVD), and hence the polar decomposition via the eigendecompositions of A^TA and AA^T. However, this procedure suffers from numerical instability.

We are not the first to treat efficient solution of linear algebra problems of small dimension. An early investigation is that of Pitsianis and Van Loan [24], who focus on multiple instances of small problems and stack the problems in order to exploit vector hardware. More recently, Dong et al. [7] have investigated how to implement LU factorizations of many small matrices on GPUs. Previous work exists on computing the polar decomposition or SVD of 3×3 matrices A, but none of the algorithms of which we are aware satisfies the requirements listed above. In particular, some algorithms (such as that in [19]) begin by computing A^TA and are numerically unstable.

In Section 2 we use an argument based on quaternions to explain how the orthogonal polar factor of a 3×3 matrix can be obtained from an eigenvector corresponding to the dominant eigenvalue of a related 4×4 symmetric matrix. This relation is the basis of the algorithm that we develop in Section 3. The algorithm of Section 3 is presented in stages, beginning with the simplest and most practically importance case in which A has a sufficiently large second largest singular value. The operation counts given in Section 4 show that the new algorithm generally requires fewer operations than the alternative that computes the polar decomposition via the SVD. Numerical experiments presented in Section 5 show that the new algorithm has an advantage in speed over its competitors without sacrificing numerical stability. Concluding remarks are given in Section 6.

2 Polar decomposition of 3 × 3 matrices and quaternions

We recall that a polar decomposition of a matrix $A \in \mathbb{R}^{n\times n}$ is a factorization $A = QH$, where Q is orthogonal and H is symmetric positive semidefinite [13, Chap. 8]. Clearly, $H = (A^TA)^{1/2}$ is always unique, and when A is nonsingular H is positive definite and $Q = AH^{-1}$ is unique.

The polar decomposition can be obtained from the SVD and vice versa. Indeed if $A = U\Sigma V^T$ is an SVD then $A = UV^T \cdot V\Sigma V^T = QH$ is a polar decomposition.

We will denote by $O(3)$ the group of 3×3 orthogonal matrices and by $SO(3)$ the subgroup of $O(3)$ of orthogonal matrices with determinant 1. The orthogonal polar factor Q of $A \in \mathbb{R}^{3\times 3}$ has the trace maximizing property

$$Q = \underset{Z\in O(3)}{\operatorname{argmax}} \operatorname{trace} Z^T A.$$

This can be shown using the SVD by noting that $\operatorname{trace}(Z^TA) = \operatorname{trace} Z^TU\Sigma V^T = \operatorname{trace} \Sigma V^TZ^TU \le \operatorname{trace} \Sigma$, and equality is attained at $Z = UV^T = Q$. Now we consider the sign of the determinant of Q. Note that if $\det A > 0$ then $\det Q = 1$. If $\det A = 0$ it is not hard to show that there is a choice of the polar decomposition satisfying $\det Q = 1$. If $\det A < 0$ then $\det(-A) > 0$ (since n is odd), so we have the polar decomposition $-A = QH$, with $\det Q = 1$, that is, $A = (-Q)H$. Hence, defining

$$\eta = \operatorname{sign}(\det A) = \begin{cases} 1, & \det A \ge 0, \\ -1, & \det A < 0, \end{cases} \tag{2.1}$$

we have

$$A = (\eta Q) \cdot H \iff Q = \underset{Z\in SO(3)}{\operatorname{argmax}}\, \eta \operatorname{trace} Z^T A, \tag{2.2}$$

This means that we can look for the matrix $Q \in SO(3)$ that maximizes the trace if $\det A \ge 0$ and minimizes the trace if $\det A < 0$. The polar factor is equal to ηQ.

Throughout this paper we use the 2-norm on $\mathbb{R}^n$, $\|x\|_2 = (x^Tx)^{1/2}$, and the Frobenius norm on $\mathbb{R}^{n\times n}$, $\|A\|_F = \operatorname{trace}(A^TA)^{1/2}$.

In the rest of this section we develop a relation between the the orthogonal polar factor of a 3×3 matrix and the eigensystem of a related 4×4 symmetric matrix. Most of the results given are essentially known but have not previously been collected together in this unified and self-contained way.

2.1 Quaternions and rotations

Our approach to computing a polar decomposition of a 3×3 real matrix is based on the trace maximization property, and it is best explained using quaternions. The ideas in this subsection about the relation between quaternions and rotations were mostly known to Cayley [4] and Hamilton [8]. They have also reappeared several times in the modern literature, for example [5, 15, 17], and are known to experts of graphic animation: see [19, 22, 28] and the references therein. An excellent reference on quaternions in linear algebra is the book by Rodman [25].

Let $\mathbb{H}$ be the ring of quaternions, with the standard basis $\{1, i, j, k\}$. $\mathbb{H}$ is a vector space over the real numbers, so any $q \in \mathbb{H}$ can be written

$$q = w + xi + yj + zk, \qquad w, x, y, z \in \mathbb{R}. \tag{2.3}$$

It is known that algebraically $\mathbb{H}$ is an associative normed division algebra. All the rules for multiplication can be deduced from the defining equations $i^2 = j^2 = k^2 = ijk = -1$. In particular, multiplication is not commutative since, for example, $ij = k = -ji$. Since $\mathbb{H}$ is isomorphic to $\mathbb{R}^4$ through (2.3), we can define a Euclidean norm on $\mathbb{H}$ by $|q| = \|[w\ x\ y\ z]\|_2 = \sqrt{w^2 + x^2 + y^2 + z^2}$. There is a conjugation defined by $(w + xi + yj + zk)^* = w - xi - yj - zk$ (note that $|q| = \sqrt{qq^*} = \sqrt{q^*q}$). If $w = 0$ we say that the quaternion $xi + yj + zk$ is *pure imaginary*. Moreover, if there is some $0 \neq \alpha \in \mathbb{R}$ such that two quaternions q_1, q_2 satisfy $q_1 = \alpha q_2$, we write $q_1 \sim q_2$. It is easy to check that $\sim$ is an equivalence relation. Recall that the quotient space $\mathcal{Q} := \mathbb{H}/\sim$ is defined as the set of equivalence classes of $\mathbb{H}$ by $\sim$. There are precisely two unimodular quaternions, equal to each other up to sign, in each equivalence class (except for the equivalence class $\{0\}$). Hence, $\mathcal{Q} \setminus \{0\}$ is one-to-one with the unit hemisphere in $\mathbb{R}^4$, and the latter is precisely the set of all possible eigenvectors of a 4×4 symmetric matrix up to normalization.

The embedding

$$\phi : \mathbb{R}^3 \to \mathbb{H} \cong \mathbb{R}^4, \qquad v = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \mapsto \phi(v) = a_1 i + a_2 j + a_3 k$$

can be applied column-by-column to matrices, inducing a mapping

$$\Phi : \mathbb{R}^{3\times 3} \to \mathbb{H}^3, \quad \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \mapsto \begin{bmatrix} a_{11}i + a_{21}j + a_{31}k \\ a_{12}i + a_{22}j + a_{32}k \\ a_{13}i + a_{23}j + a_{33}k \end{bmatrix}^T.$$

Moreover we have the mapping

$$\psi : \mathbb{H} \to \mathbb{R}^{3\times 3},$$
$$q \mapsto \psi(q) = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & w^2 + y^2 - x^2 - z^2 & 2yz - 2xw \\ 2xz - 2yw & 2yz + 2xw & w^2 + z^2 - x^2 - y^2 \end{bmatrix}. \tag{2.4}$$

Here are some properties of the mapping ψ in (2.4) that will be useful below. They are also discussed in [4], [5, sect. 3], [17, sect. 7.1], for example.

Theorem 2.1 *The following properties hold.*

(a) $\psi(q^*) = \psi(q)^T$ *and* $\psi(q) = \psi(-q)$.

(b) *For any* $v \in \mathbb{R}^3$ *and any* $q \in \mathbb{H}$, $q\phi(v)q^*$ *is pure imaginary and* $q\phi(v)q^* = \phi(\psi(q)v)$. (*Note that the order of the factors in* $q\phi(v)q^*$ *is important, because* $\mathbb{H}$ *is not a commutative ring.*)

(c) *For any* $q \in \mathbb{H}$, $\psi(q)\psi(q^*) = |q|^4 I_3$.

(d) *The map* $\hat{\psi} = \frac{\psi(q)}{|q|^2} : \mathcal{Q} \setminus \{0\} \to SO(3)$ *is a bijection.*

Proof (a) Both properties are manifest from the definition.

(b) This can be checked by a direct computation; see also [4, 5, sect. 3], [17, sect. 7.1].

(c) Observe that the previous property implies $\phi(\psi(q_1)\psi(q_2)v) = q_1 q_2 \phi(v) q_2^* q_1^*$. Hence for any $v \in \mathbb{R}^3$ we have $\psi(q)\psi(q^*)v = \phi^{-1}(qq^*\phi(v)qq^*) = |q|^4 v$, and therefore $\psi(q)\psi(q^*) = |q|^4 I_3$.

(d) To see that $\hat{\psi}(q)$ is orthogonal, we may assume $|q| = 1$ without loss of generality: then (c) yields $\psi(q)\psi(q^*) = \psi(q)\psi(q)^T = I_3$. Moreover, it is evident from a direct computation that $\det \psi(q) = 1$. Furthermore, the proof of (c) shows that $\hat{\psi}$ is indeed a group isomorphism (hence, in particular, a bijection): for more details see, for example, [5, sect. 3]. □

Note that Theorem 21 (d) implies that there is a bijection between the set of 3×3 orthogonal matrices of unit determinant and the set of all unit vectors in $\mathbb{R}^4$ (up to the sign). Indeed, both sets are one-to-one with $\mathcal{Q} \setminus \{0\}$. Furthermore, the bijection is explicitly given by (2.4).

2.2 Polar decomposition via trace maximization

Using Theorem 2.1 (b), for a 3×3 matrix A that we partition by columns $A = [a_1\ a_2\ a_3]$, we have

$$\psi(q)A = [\,\phi^{-1}(q\phi(a_1)q^*) \quad \phi^{-1}(q\phi(a_2)q^*) \quad \phi^{-1}(q\phi(a_3)q^*)\,].$$

Hence, for $Q^T = \psi(q)$, using the algebra of $\mathbb{H}$ and denoting $\mathrm{Re}(w + ix + jy + zk) = w$,

$$\tau(w, x, y, z) = \mathrm{trace}(Q^T A) = -\mathrm{Re}(iq\phi(a_1)q^* + jq\phi(a_2)q^* + k\phi(a_3)q^*).$$

The latter equation defines a function $\tau : \mathbb{R}^4 \to \mathbb{R}$. Since ψ in (2.4) maps $q \in \mathbb{H}$ with $|q| = 1$ to some $Q \in SO(3)$ by Theorem 2.1 (d), we can now use a Lagrange multiplier method to solve our constrained maximum problem, by looking for the stationary points of

$$f(\lambda, w, x, y, z) = \tau(w, x, y, z) + \lambda(1 - w^2 - x^2 - y^2 - z^2).$$

Let $v = [w\ x\ y\ z]^T$ and introduce the symmetric matrix

$$B = \begin{bmatrix} a_{11}+a_{22}+a_{33} & a_{23}-a_{32} & a_{31}-a_{13} & a_{12}-a_{21} \\ a_{23}-a_{32} & a_{11}-a_{22}-a_{33} & a_{12}+a_{21} & a_{13}+a_{31} \\ a_{31}-a_{13} & a_{12}+a_{21} & a_{22}-a_{11}-a_{33} & a_{23}+a_{32} \\ a_{12}-a_{21} & a_{13}+a_{31} & a_{23}+a_{32} & a_{33}-a_{11}-a_{22} \end{bmatrix}. \tag{2.5}$$

Only a few simple algebraic manipulations are needed to see that

$$\tau(w,x,y,z) = v^T Bv, \qquad f(\lambda,w,x,y,z) = v^T Bv + \lambda(1 - v^T v). \tag{2.6}$$

Therefore the condition for a stationary point, $\nabla f = 0$, is equivalent to the equations

$$Bv = \lambda v, \qquad \|v\|_2 = 1.$$

Now we derive some properties of B. Recall that the adjugate of a matrix A, denoted $\operatorname{adj} A$, is the matrix whose (i, j) element is $(-1)^{i+j}$ times the minor obtained by deleting the jth row and the ith column of A. Here, and throughout, $\|A\|_F = \operatorname{trace}(A^TA)^{1/2}$ is the Frobenius norm.

Lemma 2.2 *For $B \in \mathbb{R}^{4\times4}$ in (2.5), defined in terms of $A \in \mathbb{R}^{3\times3}$,*

$$\det(xI - B) = x^4 - 2\|A\|_F^2 x^2 - 8\det(A)x + \|A\|_F^4 - 4\|\operatorname{adj}(A)\|_F^2.$$

Proof The proof is by direct verification. □

For the following results we denote the singular values of A by $\sigma_1, \sigma_2, \sigma_3$, with $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Recall that η is defined in (2.1). Expressing the various norms in terms of the singular values yields an alternative expression for the determinant.

Corollary 2.3 $\det(xI - B) = x^4 - 2(\sigma_1^2 + \sigma_2^2 + \sigma_3^2)x^2 - 8\eta\sigma_1\sigma_2\sigma_3 x + \sigma_1^4 + \sigma_2^4 + \sigma_3^4 - 2(\sigma_1^2\sigma_2^2 + \sigma_1^2\sigma_3^2 + \sigma_2^2\sigma_3^2)$.

The next theorem gives the spectrum of B in terms of the SVD of A. It has been previously proved by Mackey [17, Prop. 6], also using the connection with quaternions.

Theorem 2.4 *The set of eigenvalues of B in (2.5) is*

$$\{\eta(\sigma_1 + \sigma_2 + \sigma_3), \eta(\sigma_1 - \sigma_2 - \sigma_3), \eta(\sigma_2 - \sigma_1 - \sigma_3), \eta(\sigma_3 - \sigma_1 - \sigma_2)\}.$$

In particular, if $\operatorname{rank} A \leq 1$ *the set of eigenvalues is* $\{\sigma_1, \sigma_1, -\sigma_1, -\sigma_1\}$ *and if* $\operatorname{rank} A = 2$ *it is* $\{\sigma_1 + \sigma_2, \sigma_1 - \sigma_2, \sigma_2 - \sigma_1, -\sigma_1 - \sigma_2\}$.

Proof The result can be obtained directly, by solving the quartic equation of Corollary 2.3. □

Recall that a dominant eigenvalue is an eigenvalue whose absolute value is maximal. Theorem 2.4 shows that the dominant eigenvalue of B is always (up to sign) the trace (or nuclear) norm of A, which we denote by

$$\mu = \sigma_1 + \sigma_2 + \sigma_3.$$

Any eigenvector v of B, normalized so that it has unit norm, gives a stationary point for the trace, via the mapping ψ. However, it must be a dominant eigenvector in order to give a global maximum for $\eta\,\mathrm{trace}(Q^T A) = \eta\tau$. Here and below, by a "dominant eigenvector" we mean an eigenvector, normalized to have unit norm, associated with a dominant eigenvalue μ. If there is a unique dominant eigenvalue, we will correspondingly speak about "the" dominant eigenvector.

To check the statement above, observe that as long as w, x, y, z lie on the unit sphere $\|v\| = 1$ then $\tau(w, x, y, z)$ is equal to the Rayleigh quotient $v^T Bv/(v^T v)$, by (2.6). Therefore, if $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$ are the eigenvalues of B we have $\lambda_4 \leq \tau(x, y, w, z) \leq \lambda_1$. Hence, when $\eta = 1$ the maximum of $\tau(w, x, y, z)$ is $\lambda_1 = \mu$, and it is achieved when $v = [w\ x\ y\ z]^T$ is a corresponding (dominant) eigenvector. Similarly if $\eta = -1$ then the maximum of $-\tau(w, x, y, z)$ is equal to $-\lambda_4 = \mu$, and this maximum is achieved when v is a corresponding (dominant) eigenvector. Note that the maximum is achieved at a unique vector (up to sign) if and only if $\det A \neq 0$.

We summarize the analysis in the following theorem, where we simplify the formula for Q by using the normalization $v_1^2 + v_2^2 + v_3^2 + v_4^2 = 1$. Recall also the transpose in (2.2), which corresponds to a conjugation, by Theorem 2.1 (a).

Theorem 2.5 *Let $A \in \mathbb{R}^{3\times 3}$ and let $B \in \mathbb{R}^{4\times 4}$ be defined as in (2.5). Let v be the dominant eigenvector of B when A is full rank, or any vector in any dominant eigenspace when A is rank deficient, normalized to have unit norm. Then a polar decomposition $A = QH$ is given by*

$$\begin{aligned} Q &= \psi(v_1 - iv_2 - jv_3 - kv_4) \\ &= \begin{bmatrix} 1 - 2(v_3^2 + v_4^2) & 2(v_2v_3 + v_1v_4) & 2(v_2v_4 - v_1v_3) \\ 2(v_2v_3 - v_1v_4) & 1 - 2(v_2^2 + v_4^2) & 2(v_3v_4 + v_1v_2) \\ 2(v_2v_4 + v_1v_3) & 2(v_3v_4 - v_1v_2) & 1 - 2(v_2^2 + v_3^2) \end{bmatrix} \end{aligned} \tag{2.7}$$

and $H = Q^T A$.

The next result has not previously appeared, and will be needed below.

Lemma 2.6 *Let $p(x) = \det(xI - B)$ be the characteristic polynomial of B and μ an eigenvalue of B with largest absolute value. Then $|p'(\mu)| = 8(\sigma_1 + \sigma_2)(\sigma_1 + \sigma_3)(\sigma_2 + \sigma_3)$.*

Proof From Corollary 2.3 we have that $p'(x) = 4x^3 - 4(\sigma_1^2 + \sigma_2^2 + \sigma_3^2)x - 8\eta\sigma_1\sigma_2\sigma_3$. Observe that for $\eta \in \{+1, -1\}$ it holds $p'(\eta x) = \eta p'(x)$, and hence $|p'(\eta x)| = |p'(x)|$.

Suppose first $\sigma_3 > 0$. By Theorem 2.4, $\mu = \eta(\sigma_1 + \sigma_2 + \sigma_3)$. Now write $p'(x) = 4x(x^2 - \sigma_1^2 - \sigma_2^2 - \sigma_3^2) - 8\sigma_1\sigma_2\sigma_3$. When $x = \mu = \sigma_1 + \sigma_2 + \sigma_3$, this yields $\frac{p'(\mu)}{8} = \mu(\sigma_1\sigma_2 + \sigma_1\sigma_3 + \sigma_2\sigma_3) - \sigma_1\sigma_2\sigma_3 = (\sigma_1 + \sigma_2)\sigma_3\mu + \sigma_1\sigma_2(\mu - \sigma_3) = (\sigma_1 + \sigma_2)(\sigma_1\sigma_2 + \sigma_1\sigma_3 + \sigma_2\sigma_3 + \sigma_3^2) = (\sigma_1 + \sigma_2)(\sigma_1 + \sigma_3)(\sigma_2 + \sigma_3)$.

If rank $A = 2$, then $\mu = \pm(\sigma_1 + \sigma_2)$ (any possible choice of the sign is allowed by the statement) and $p'(x) = 4x(x^2 - \sigma_1^2 - \sigma_2^2)$, and hence $|p'(\mu)| = 8\sigma_1\sigma_2(\sigma_1 + \sigma_2)$.

Finally, if rank $A \leq 1$, then $\mu = \pm\sigma_1$, and $p'(x) = 4x(x^2 - \sigma_1^2)$, implying $|p'(\mu)| = 0$. □

Finally we note a bound for the eigenvalue of B of largest absolute value, derived via standard inequalities on matrix norms. We may assume without loss of generality that $\eta = 1$. Then the largest eigenvalue λ_1 is nonnegative and satisfies

$$\|A\|_F \leq \lambda_1 \leq \sqrt{3}\|A\|_F. \tag{2.8}$$

3 Algorithms for the polar decomposition

The theoretical analysis in Section 2 suggests an algorithm to compute the polar decomposition of $A \in \mathbb{R}^{3\times 3}$: form the matrix B, approximate its dominant eigenvector v, form the polar factor Q via Theorem 2.5, and finally compute $H = Q^T A$. The most basic implementation of the idea is to compute a complete eigendecomposition of B via any reliable eigensolver, such as the QR algorithm.

Algorithm 3.1 This algorithm computes the polar decomposition $A = QH$ of $A \in \mathbb{R}^{3\times 3}$.

```
1. Form $B \in \mathbb{R}^{4\times 4}$ in (2.5) from $A$.
2. Compute an eigendecomposition $B = Z\Lambda Z^T$ using the QR algorithm.
3. Select a dominant eigenvector $v$.
4. Form the matrix $Q$ from $v$ using (2.7).
5. Compute the upper triangle of $H = Q^T A$ and set the lower triangle equal to the
   upper triangle.
```

This idea is very simple, but it is clearly overkill: only one eigenvector of B is needed, but Algorithm 3.1 computes all four of them. A method that computes only a dominant eigenvector, such as the power method, inverse iteration, or the Rayleigh quotient iteration, could be cheaper. However, the linear convergence of the power method and inverse iteration can make them too slow. The cubically convergent Rayleigh quotient method, or the conjugate gradient method used for instance in [3], have a different disadvantage: they do not guarantee convergence to the dominant eigenvector, at least if we do not have a good initial guess.

To overcome these difficulties, we will first compute a cheap estimate $\widetilde{\lambda}_1$ of λ_1, the dominant eigenvalue of B. Then we will apply inverse iteration to the shifted matrix $B_s = \widetilde{\lambda}_1 I - B$.

Both for the analysis, and in the algorithm, it is convenient to scale A to have unit Frobenius norm; thus the algorithm preprocesses $A \leftarrow A/\|A\|_F$. Hence throughout this section we assume that $\sigma_1^2 + \sigma_2^2 + \sigma_3^2 = 1$. We will see below that there are three different regimes, depending on the estimated magnitude of σ_2, the second largest singular value of A. However, our algorithm turns out to be very simple when σ_2 is large enough, and so we discuss this case first.

3.1 Main branch of the algorithm

For simplicity of the exposition, we begin by presenting Algorithm 3.2, which is intended for the case where σ_2 is large enough. In practice, most inputs satisfy this criterion. Algorithm 3.2 is the main branch of Algorithm 3.5, which handles the general case.

Algorithm 3.2 This algorithm, intended for the case when σ_2 is sufficiently large, computes the polar decomposition $A = QH$ of $A \in \mathbb{R}^{3\times 3}$ with $\|A\|_F = 1$.

```
 1.  Form $B \in \mathbb{R}^{4\times 4}$ in (2.5) from $A$.
 2.  Compute $b = \det B$ from an $LU$ factorization with partial pivoting.
 3.  Compute $d = \det A$ from an $LU$ factorization with partial pivoting.
 4.  if $d < 0$, $B = -B$, $d = -d$, end
 5.  Estimate $\lambda_1$, a dominant eigenvalue of $B$, via Algorithm 3.3.
 6.  $B_s = \lambda_1 I - B$
 7.  Compute an $LDL^T$ factorization with diagonal pivoting, $P^T B_s P = LDL^T$.
 8.  $v = PL^{-T}e_4/\|L^{-T}e_4\|_2$
 9.  Form the matrix $Q$ using (2.7).
10.  Compute the upper triangle of $H = Q^T A$ and set the lower triangle equal to the
     upper triangle.
```

In the factorization on line 7 the pivot at each stage is chosen as the largest element in absolute value on the diagonal. The factorization exists in view of B_s having three positive eigenvalues.

In the next subsections we explain in more detail how Algorithm 3.2 works. As indicated above, the basic strategy is to first approximate λ_1 and then to apply inverse iteration to the shifted matrix B_s.

3.1.1 Computing the characteristic polynomial of B

The cheapest way to estimate the dominant eigenvalue λ_1 is via the characteristic polynomial of B. Of course, this is not always a numerically stable approach, but it provides a cheap estimate that in some circumstances can be reasonably good. The first ingredient is the computation of the constant term in the characteristic polynomial of B, which is equal to the determinant of B. The latter can be computed with various methods: Leibniz's formula (for $B \in \mathbb{R}^{n\times n}$: $\det B = \sum_{\sigma\in S} \operatorname{sign}\sigma \prod_{i=1}^{n} a_{i,\sigma_i}$, where the sum is over all permutations of the set $\{1, 2, \ldots, n\}$), or as $1-4\|\operatorname{adj}(A)\|_F^2$ (by Lemma 2.2), with each element of $\operatorname{adj}(A)$ computed as a determinant of a 2×2 matrix, or via an LU factorization. As the last method requires the fewest flops, this is our choice.

We now assume that $b = \det B$ (on line 2) has been computed with an absolute forward error cu with u the machine precision and c a moderate constant.

We then need to compute $\det A$, which is done via an LU factorization of A with partial pivoting. In principle, a computation via Leibniz's formula would be less expensive.[1] However, it is important to obtain the correct sign of $\det A$, as the code uses this information to decide whether the dominant eigenvalue of B is positive or negative: if $\det A < 0$ then we preprocess $B \leftarrow -B$. A careless evaluation of $\operatorname{sign}\det A$ could result in instability in the computation of the polar decomposition. However, by the analysis in [23] the computation of a determinant via Gaussian elimination yields the correct sign unless the smallest singular value is a moderate multiple of u, and in this case it can be shown that either sign is acceptable. Therefore, from now on we may assume without loss of generality that $\det A \geq 0$ and that $d = \det A$ has been computed with a small absolute forward error.

3.1.2 Estimating λ_1

At this point, we need to compute λ_1. As described by the pseudocode below, the default strategy is to use an exact formula to solve the quartic equation in Lemma 2.2. The algorithm below is based on a slight modification of the general method given in [26, sect. 1].

Algorithm 3.3 This algorithm computes an estimate of λ_1, a dominant eigenvalue λ_1 of B in (2.5) given $b = \det B$ and $d = \det A$.

```
1.  $\tau_1 = 10^{-4}$    % Tolerance.
2.  if $b + 1/3 > \tau_1$
3.      $c = 8d$
4.      $\delta_0 = 1 + 3b$
5.      $\delta_1 = -1 + (27/16)c^2 + 9b$
6.      $\alpha = \delta_1/\delta_0^{3/2}$
7.      $z = (4/3)(1 + \delta_0^{1/2}\cos(\arccos(\alpha)/3))$
8.      $s = z^{0.5}/2$
9.      $\lambda_1 = s + (\max(0, 4 - z + c/s))^{1/2}/2.$
10. else
11.     Use Newton's method (Algorithm 3.4) to approximate $\lambda_1$.
12. end
```

We now analyze the analytic formulae to spot potential dangers. One comes from the potential ill conditioning of the scalar root-finding problem. In the approach discussed above, a potential danger is obvious a priori: Algorithm 3.3 computes the dominant eigenvalue of a symmetric matrix (whose condition number is 1) as the dominant root of a scalar quartic equation (whose condition number can potentially be large). This may be inadvisable when λ_1 and λ_2 are close, i.e., λ_1 is a near multiple root. The bigger the gap between λ_1 and λ_2 the more we can trust our cheap estimate of λ_1 and hence the less work is needed to obtain the dominant eigenvector.

[1]In spite of its $\mathcal{O}(n!)$ asymptotic cost, for $n \leq 3$ using the explicit formula is cheaper than performing an LU factorization!

In Section 3.1.3, we will see that the simple logical test $1 - b > \tau_2$, where τ_2 is a tolerance, implies

$$\lambda_1 - \lambda_2 > \sqrt{\tau_2}. \tag{2.9}$$

In the more sophisticated Algorithm 3.5, this condition is actually checked. Although we will not explicitly use this fact, it is worth noting that, in view of Theorem 2.4 or Lemma 2.6, λ_1 and λ_2 are close precisely when $\sigma_2 + \sigma_3$ is small, i.e., the numerical rank of A is 1.

A subtler pitfall lies within the intermediate computations in Algorithm 3.3. The quantity $\alpha = \delta_1/\delta_0^{3/2}$ cannot be stably computed when δ_0 is small, i.e., $b \approx -1/3$. To explain why, we interpret the quantities involved as bivariate functions of real variables. Let us parametrize $\sigma_1 = \cos\theta$, $\sigma_2 = \sin\theta\cos\phi$, $\sigma_3 = \sin\theta\sin\phi$, with $0 \le \phi \le \frac{\pi}{4}$ and $0 \le \theta \le \arctan(1/\cos\phi)$. It is not difficult to show that, when $b = 1 - 4\sin^2\theta(\cos^2\theta + \sin^2\theta\sin^2\phi\cos^2\phi)$ is close to $-1/3$, then necessarily (θ, ϕ) is close to $(\arctan\sqrt{2}, \pi/4)$, or in other words, $\sigma_1 \approx \sigma_2 \approx \sigma_3 \approx 1/\sqrt{3}$. This in turn implies that $d = \cos\theta\sin^2\theta\sin\phi\cos\phi$ must be close to $1/\sqrt{27}$. Clearly, this implies that, for any sequence $(\theta_n, \phi_n) \to (\arctan\sqrt{2}, \pi/4)$, one has $\delta_0(\theta_n, \phi_n) \to 0$ and $\delta_1(\theta_n, \phi_n) \to 0$. Unfortunately, α is not continuous at $(\theta, \phi) = (\arctan\sqrt{2}, \pi/4)$. One way to see this is to compare the Taylor expansions of δ_0^2 and δ_1^3 to show that α^2 is not continuous, and hence α cannot be either. However, it is not hard to show that $0 \le \alpha^2 \le 1$, and hence $\cos(\frac{1}{3}\arccos\alpha)$ remains bounded, implying that $\sqrt{\delta_0}\cos(\frac{1}{3}\arccos\alpha) \to 0$. Numerically, we cannot expect any correct significant digits from the intermediate computation of α in this scenario. Therefore, the analytic method is best avoided when δ_0 is small, i.e., b is close to $-1/3$. In Algorithm 3.3, we check whether $b + 1/3$ is smaller than a tolerance τ_1. If this happens, instead of using the analytic formula we employ the more expensive but numerically safer Newton's method (Algorithm 3.4).

Algorithm 3.4 This algorithm uses Newton's method to estimate a dominant eigenvalue of B in (2.5).

```
1. $x = \sqrt{3}$
2. $x_{\text{old}} = 3$
3. while $x_{\text{old}} - x > 10^{-15}$
4.     $x_{\text{old}} = x$
5.     Evaluate $p = p(x) = \det(xI - B)$ by Horner's method.
6.     Evaluate $p_d = p'(x)$ by Horner's method.
7.     $x = x - p/p_d$
8. end
```

At first sight, one might think that Newton's method is a good idea in general. In fact, by (2.8) the dominant eigenvalue of B must lie in the interval $\mathcal{I} := [1, \sqrt{3}]$, and since $\det(xI - B)$ is a convex function on $\mathcal{I}$, the method is guaranteed to converge quadratically and monotonically from above if we use $\sqrt{3}$ as a starting point. However, Newton's method is much more expensive than the analytic formula when many more than a couple of iterations are required for convergence. Fortunately, the scenario when the analytic formula might suffer from numerical instability is favorable to Newton's method. Indeed, $(\theta, \phi) \approx (\arctan\sqrt{2}, \pi/4)$ implies that $\lambda_1 = \cos\theta + \sin\theta(\cos\phi + \sin\phi) \approx \sqrt{3}$, and, since the starting point is $\sqrt{3}$, Newton's method is guaranteed to converge within only a few iterations (with the current choice of tolerance τ_1 in Algorithm 3.3, one iteration suffices).

3.1.3 Obtaining the dominant eigenvector

Finally, we need to use the approximation of the dominant eigenvalue λ_1 of B to obtain the dominant eigenvector v. Suppose that $1 - b > \tau_2$. Then, $\tau_2 < 4\sigma_1^2(\sigma_2^2 + \sigma_3^2) + 4\sigma_2^2\sigma_3^2 \leq 4\sigma_1^2(\sigma_2^2 + \sigma_3^2) + (\sigma_2^2 + \sigma_3^2)^2 = 4\sin^2\theta - 3\sin^4\theta$, where we have used the fact that for any $x, y > 0$, $(x+y)^2 - 4xy = (x-y)^2 \geq 0$ and the parametrization of the previous subsection. This yields

$$\sin^2\theta > 2\frac{1 - \sqrt{1 - \frac{3\tau_2}{4}}}{3} > \frac{\tau_2}{4}.$$

Therefore $\lambda_1 - \lambda_2 = 2(\sigma_2 + \sigma_3) \geq 2\sin\theta > \sqrt{\tau_2}$. So we are guaranteed that the dominant eigenvalue is well separated, and hence inverse iteration will be fast to converge.

Furthermore, $1 - b > \tau_2$ implies by Lemma 2.6 that the absolute value of the derivative of the characteristic polynomial of B, evaluated at λ_1, is at least $\beta = \frac{4}{3}\sqrt{\tau_2} + \frac{2}{\sqrt{3}}\tau_2$, giving the upper bound $1/\beta$ for the condition number of λ_1 as a root of the characteristic polynomial. In Algorithm 3.5, to be described in the next section, we check the condition $1 - b > \tau_2$, with a current choice of tolerance $\tau_2 = 10^{-4}$. Here and below, we denote by $\hat{\lambda}_1 := \lambda_1 + 2\epsilon$ the computed value, as opposed to the exact value λ_1, of the dominant eigenvalue of B. We assume that the forward error, $2|\epsilon|$, is of order the unit roundoff u times a moderate constant. (The factor 2 in the definition is just for notational convenience in the following).

At this point, Algorithm 3.2 computes an LDL^T factorization with diagonal pivoting of a 4×4 symmetric matrix B_s of numerical rank 3 whose smallest eigenvalue is bounded below by -2ϵ. Indeed, recall from (2.9) that the third largest eigenvalue of B_s is $\lambda_1 - \lambda_2 > \sqrt{\tau_2}$. The analysis in [11, 12, Chap. 10] can be applied to argue that the computed $\hat{L}$ and $\hat{D}$ satisfy $\|B_s - \hat{L}\hat{D}\hat{L}^T\|_F \leq \theta u$, with θ a moderate constant. In particular θ is the sum of two terms: the bound in the error analysis of the Cholesky factorization for a semidefinite matrix (note that the right hand side of [12, eq. (10.19)] is $\sqrt{21}$ for $n = 4$ and rank 3), plus another term coming from the approximation of λ_1.

From Theorem 2.4, we see that inverse iteration converges linearly with rate $|\hat{\lambda}_1 - \lambda_2|/|\hat{\lambda}_1 - \lambda_1| > \gamma$ for some constant $\gamma \approx \sqrt{\tau_2}/(2|\epsilon|)$. Had we computed both λ_1 and the LDL^T factorization of B_s in exact arithmetic then we would have had $d_{44} = 0$. (In floating point arithmetic, we have $|d_{44}| \leq \theta u$, and in practice $|d_{44}|$ is usually much smaller). This suggests that $v := \hat{L}^{-T}e_4/\|\hat{L}^{-T}e_4\|_2$ is already a good estimate of the dominant eigenvector of B. Indeed, for the residual we have $\|B_s v\|_2 \leq \|B_s - \hat{L}\hat{D}\hat{L}^T\|_F\|v\|_2 + \|\hat{L}\hat{D}\hat{L}^T v\|_2 \leq \theta u(1 + \|L^{-T}e_4\|_2^{-1}) \leq 2\theta u$. The vector v can be easily computed directly, using the exact formula for the inverse of a unit lower triangular 4×4 matrix.

3.2 Complete algorithm for the general case

Now we give the complete algorithm, which is applicable to any $A \in \mathbb{R}^{3\times 3}$.

Algorithm 3.5 This algorithm computes the polar decomposition $A = QH$ of $A \in \mathbb{R}^{3\times 3}$ with $\|A\|_F = 1$.

```
 1. $\tau_2 = 10^{-4}$    % Tolerance.
 2. Form $B \in \mathbb{R}^{4\times 4}$ from $A$ via (2.5).
 3. Compute $b = \det B$ from an $LU$ factorization with partial pivoting.
 4. if $b < 1 - \tau_2$
       % Dominant eigenvalue of $B$ is well separated.
 5.    Call Algorithm 3.2.
 6. else
 7.    Compute $d = \det A$ using an $LU$ factorization with complete
       pivoting.
 8.    If $d < 0$, $B = -B$, end
 9.    Estimate $\lambda_1$ using Algorithm 3.3.
10.    $B_s = \lambda_1 I - B$
11.    if $\log_{10} |u_{22}| > -7.18$
12.        $n_{it} = \lceil 15/(16.86 + 2\log_{10} |u_{22}|) \rceil$
13.        Compute $B_s = LDL^T$ by block $LDL^T$ factorization with
           Bunch–Parlett pivoting.
14.        $v = L^{-T} e_4 / \|L^{-T} e_4\|_2$   % Initial guess.
15.        for $i = 1{:}\, n_{it}$
16.            Update $v$ using one step of inverse iteration with $LDL^T$.
17.        end
18.    else
19.        Compute $B_s = LDL^T$ by block $LDL^T$ factorization with
           Bunch–Parlett pivoting.
20.        $V = L^{-T} [e_3\ e_4]$    % Initial guess.
21.        for $i = 1 : 2$
22.            Orthonormalize $V$ via QR factorization.
23.            Update $V$ using one step of inverse subspace iteration
               with $LDL^T$ factorization.
24.        end
25.        Orthonormalize $V$ via QR factorization.
26.        $B_p = V^T B_s V \in \mathbb{R}^{2\times 2}$
27.        Find $w$, eigenvector of smallest eigenvalue of $B_p$, by
           analytic formula.
28.        $v = Vw$
29.    end
30.    Form the matrix $Q$ from $v$ as in Theorem 2.5.
31.    Compute $H = Q^T A$.
32. end
```

We now explain the computations in Algorithm 3.5 when $b \geq 1 - \tau_2$. In this case we need to estimate σ_2 in order to decide whether to apply inverse iteration or subspace iteration on a 4×2 matrix.

Recall that we have defined $\epsilon = \frac{1}{2}(\hat{\lambda}_1 - \lambda_1)$. Then the convergence ratio of inverse iteration is

$$\left|\frac{\hat{\lambda}_1 - \lambda_2}{\hat{\lambda}_1 - \lambda_1}\right| = \left|\frac{\sigma_2 + \sigma_3 + \epsilon}{\epsilon}\right|.$$

On the other hand the convergence ratio for subspace iteration with a two-dimensional subspace is

$$\left|\frac{\hat{\lambda}_1 - \lambda_3}{\hat{\lambda}_1 - \lambda_2}\right| = \left|\frac{\sigma_1 + \sigma_3 + \epsilon}{\sigma_2 + \sigma_3 + \epsilon}\right|.$$

However, if subspace iteration is chosen, we do not really care that our ansatz 4×2 matrix V completes its convergence towards an invariant subspace (corresponding to the dominant and second dominant eigenvectors): for our goals it suffices that a good approximation of the dominant eigenvector lies in V. This happens with a potentially faster rate,

$$\left|\frac{\hat{\lambda}_1 - \lambda_3}{\hat{\lambda}_1 - \lambda_1}\right| = \left|\frac{\sigma_1 + \sigma_3 + \epsilon}{\epsilon}\right|.$$

We want to make a decision on what choice to make according to the maximum number of iterations that we expect from either method. Let $\omega = -\log_{10}\sigma_2$ and suppose $2 < \omega < 8$ so that $\sigma_2 \gg \epsilon$. The absolute condition number of the polynomial root finding problem $p(x) = 0$ is $|p'(\lambda)|^{-1}$ for the root λ. Observe that

$$1 - b = 4(\sigma_1^2\sigma_2^2 + \sigma_1^2\sigma_3^2 + \sigma_2^2\sigma_3^2) \geq 4\sigma_1^2(1 - \sigma_1^2).$$

Thus, $\tau_2 \geq 1 - b$ implies that

$$\sigma_1^2 \geq \frac{1 + \sqrt{1 - \tau_2}}{2}$$

and therefore $\sigma_1^2 > 0.99997$ (for $\tau_2 = 10^{-4}$). Hence, we can estimate the forward error in $\hat{\lambda}_1$ via Lemma 2.6:

$$2|\epsilon| \lesssim \frac{u}{8(\sigma_1 + \sigma_2)(\sigma_1 + \sigma_3)(\sigma_2 + \sigma_3)} < \frac{u}{8\sigma_1^2\sigma_2} < \frac{u}{7.99979\sigma_2}.$$

Hence, setting $u = 1.1 \times 10^{-16}$, i.e., assuming an implementation in double precision floating point arithmetic, we get $\log_{10}|\epsilon| \lesssim \omega - 16.86$.

Assume without loss of generality that $\epsilon > 0$. A pessimistic bound for the convergence ratio of inverse iteration is $\rho_I = (\sigma_2 - \epsilon)/\epsilon$; similarly, a pessimistic bound for the convergence of the dominant eigenvector within subspace iteration is $\rho_S = (\sigma_1 - \epsilon)/\epsilon$. Subspace iteration costs at least twice as much per iteration, plus some extra fixed costs. Heuristically, we estimate the cost of subspace iteration as about three times that of inverse iteration. Hence, we do not wish to employ subspace iteration unless $\rho_I^3 < \rho_S$. From the above, we have $\log_{10}\rho_I \gtrsim 16.86 - 2\omega$

and $\log_{10}\rho_S \gtrsim 16.86 - \omega$. Hence, $3\log_{10}\rho_I < \log_{10}\rho_S$ corresponds (approximately) to $\omega > 6.74$. In practice, as we have no access to ω, we use complete pivoting in $P_1AP_2 = LU$ and evaluate $\hat{\omega} = -\log_{10}|u_{22}|$. A discussion of why LU with complete pivoting is a reliable rank revealing factorization can be found in [12, sect. 9.12], and we note that the key constant $n2^{n-1}$ appearing there is 12 for $n = 3$.

If $\hat{\omega} < 7.18$, we use inverse iteration, otherwise we use subspace iteration. (7.18 is approximately the value corresponding to 6 inverse iterations). If $\hat{\omega} < 7.18$ we also estimate the number of inverse iterations needed for convergence, with the formula $n_{it} = \lceil 15/(16.86 - 2\hat{\omega})\rceil$. Note that this is equivalent to setting a tolerance 10^{-15}, and that it yields $1 \le n_{it} \le 6$. The reason to estimate a priori the number of iterations is to avoid having to evaluate a stopping criterion, thus saving flops.

Assuming now $\hat{\omega} < 7.18$, we compute the LDL^T factorization of B_s. We then apply inverse iteration for n_{it} steps, with starting vector $\hat{L}^{-T}e_4$.

It remains to complete the analysis to include the possibility $\hat{\omega} \ge 7.18$. In this case, given the analysis above, due to efficiency issues we resort to subspace iteration, with starting guess $V = L^{-T}[e_3\ e_4]$. Within the subspace iteration we orthonormalize at each step, for two steps. We also expect potential loss of accuracy in approximating λ_1, because the latter is close to being a double root of the characteristic polynomial. We can expect an accuracy ϵ of order $\sqrt{u}$, and now we are in the regime $\epsilon \gtrsim \sigma_2$. Hence a lower bound for the convergence rate is $|(\hat{\lambda}_1 - \lambda_3)/(\hat{\lambda}_1 - \lambda_1)| \gtrsim |\frac{2-\sqrt{3\tau_2}}{\epsilon}|$. Thus, we expect convergence after about $-15/\log(\sqrt{u}) \approx 2$ steps. This justifies the fact that we implement a fixed number of steps (equal to 2) without checking against any stopping criterion.

At this point, we project onto the 2×2 symmetric matrix $H = V^TBV$ and compute the dominant eigenvector w of H via an analytic expression. The dominant eigenvector of B is then computed as $v = Vw$.

4 Flop counts

Here we give a brief discussion of the operation count of Algorithm 3.5. A detailed breakdown of the operation count can be found in the Supplementary Materials.

The most advantageous situation occurs when Algorithm 3.5 calls Algorithm 3.2. It is also the most common scenario in graphics applications, where the input matrices are typically well conditioned [27]. Assuming that the analytic formula is used to estimate λ_1 in Algorithm 3.3 the total cost of Algorithm 3.5 is 250 flops (if Newton's method is used instead then not many iterations are needed and hence this has little effect on the flop count). This is comparable to the number of flops needed to compute the polar decomposition by applying the analytic method in [16] to diagonalize AA^T. If Algorithm 3.2 cannot be applied then within Algorithm 3.5 the LU factorization of A is performed with complete pivoting in order to obtain an order of magnitude estimate of σ_2. As explained in Section 3.2, this is done by testing whether the quantity $\hat{\omega}$ is large enough or smaller than a certain threshold. In the former case, inverse iteration is applied. The cost for this case is between 314 and 524 flops, according to how many iterations are needed. Finally, if $\hat{\omega}$ is small we employ subspace iteration, for which the cost is at most 540 flops. Hence the proposed algorithm typically requires about 250 flops and in the worst case about 550 flops.

This should be compared against the standard method employing an SVD. With this approach, one first computes an SVD (including the singular vectors) of A. Then the polar factor is computed via one matrix multiplication (costing $27M+18A$, where M denotes a multiplication and A an addition) and the Hermitian factor is recovered from its eigendecomposition (cost $36M + 18A$). So the total cost is that of an SVD, which clearly dominates, plus a further 99 flops for postprocessing.

What is the cost of an SVD? It costs 119 flops to bidiagonalize a 3×3 matrix (employing Householder reflectors), including the cost of storing the information that is needed to recover the singular vectors. Concerning the iterative part, LAPACK implements the Demmel–Kahan QR iteration [6] until one singular value (usually the smallest) has converged. Then the SVD of the deflated 2×2 bidiagonal is computed directly. The cost of a step of the Demmel–Kahan iteration, excluding the computation of a shift (if any), is 108 flops (36 for the singular values plus 72 to update the singular vectors). Since shifted QR is eventually cubically convergent, one can argue that a "typical" sequence for a superdiagonal element is $\mathcal{O}(10^{-1}) \to \mathcal{O}(10^{-3}) \to \mathcal{O}(10^{-9}) \to$ converged, which means 3 iterations. Indeed, we have run the iteration on a sample of random bidiagonal matrices of Frobenius norm 1, observing on average about 2.7 iterations for convergence. As a result of the above observation and of the experiments, we feel that it is fair to assume that at least 1 and at most 4 iterations will be required. Finally, the SVD of the 2×2 matrix is computed via a direct method, and it costs 35 flops to determine the singular values and further 36 flops for updating the singular vectors. In total the SVD approach thus costs about 400 flops in the best case and about 700 flops in the worst case, depending on the number of iterations required for convergence.

5 Numerical experiments

5.1 Timings

For timing purposes, we have implemented in Julia [2] the main branch of Algorithm 3.5, i.e., Algorithm 3.2, and compared it with several alternative algorithms, also implemented in Julia. The other algorithms are the standard method of obtaining the polar decomposition from the SVD (the implementation makes calls to the

Table 1 Timings (seconds) for 10^5 calls to Julia implementations of several algorithms

	Algorithm 3.1	Algorithm 3.2	SVD	Newton	QDWH
Random inputs	2.86	1.41	2.07	3.57	12.0
Matrix (5.1)	2.62	1.38	1.89	3.48	8.35

Table 2 Relative forward errors in U for various algorithms, for input (5.2)

y^2	Condition number	Algorithm 3.2	Algorithm 3.5	`polar_svd`
1	1	2.64e-16	2.64e-16	2.42e-16
$10^{-4} = \tau_2$	57.7	3.62e-14	3.62e-14	2.74e-16
10^{-8}	5.77e3	2.38e-10	3.53e-14	4.46e-14
10^{-12}	5.77e5	2.82e-6	2.37e-11	1.61e-11
10^{-16}	5.77e7	4.84e-2	1.47e-9	2.83e-9

LAPACK algorithms for the SVD), the Newton iteration [13, Alg. 8.20], and the QDWH algorithm [20, 21]. The experiments were done on a MacBook Pro with a 2.5 GHz Intel Core i5 processor. Table 1 gives the timings for 10^5 calls to the different algorithms for normally distributed random inputs. For reproducibility, we have also tested the algorithm for a specific (well conditioned) input:

$$A = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.1 & -0.1 & 0 \\ 0.3 & 0.2 & 0.1 \end{bmatrix}. \tag{5.1}$$

The experiments show a clear advantage in execution time for Algorithm 3.2 over the other algorithms. We observe that the ratio of timings between Algorithm 3.2 and the method based on the SVD is very close to the theoretical value expected from the flop count.

5.2 Accuracy

Here we report some tests of accuracy based on a MATLAB implementation of Algorithm 3.5. We compare with the code `polar_svd` from the Matrix Function toolbox [9], which applies the standard method of obtaining the polar decomposition from the SVD.

First, we provide a simple test to illustrate that Algorithm 3.2 can be inaccurate if the condition $1 - b > \tau_2$, where $b = \det B$, is not satisfied. Consider the matrix, parametrized by $y \in [0, 1]$,

$$A = \frac{1}{1275}\left(\begin{bmatrix} 720 & -650 & 710 \\ 396 & -145 & 178 \\ 972 & 610 & -529 \end{bmatrix} y + \begin{bmatrix} -25 & 300 & 300 \\ 70 & -840 & -840 \\ -10 & 120 & 120 \end{bmatrix}\right). \tag{5.2}$$

Table 3 Singular values prescribed to be equal to 1, 10^{-1}, and 10^{-2}. Here we are sure that Algorithm 3.5 calls Algorithm 3.2. The condition number of U is $\kappa_U \approx 10.6$.

	Algorithm 3.1	Algorithm 3.5	`polar_svd`
Forward error in H	3.3e-15	9.7e-16	1.7e-15
Forward error in U	7.3e-15	6.0e-15	8.1e-15
Backward error	5.5e-15	1.3e-15	2.1e-15

Table 4 Singular values prescribed to be equal to 1, 10^{-5}, and 10^{-12}. Here, σ_2 is too small for Algorithm 3.2 to work properly. However, ω is small enough that inverse iteration will be applied by Algorithm 3.5. The condition number of U is $\kappa_U \approx 1.15 \times 10^5$

	Algorithm 3.1	Algorithm 3.5	`polar_svd`
Forward error in H	2.3e-15	6.0e-15	9.4e-16
Forward error in U	8.6e-11	1.6e-11	3.3e-11
Backward error	5.2e-15	1.6e-15	1.6e-15

We generated this example as $A = Q_1 \operatorname{diag}(1, y, y) Q_2$ for certain rational orthogonal matrices Q_1 and Q_2. Hence, for this input, $1 - b = 4y^2$. Therefore, we expect Algorithm 3.2 to struggle when $y^2 \ll \tau_2$. (For $y^2 \geq \tau_2$, Algorithm 3.5 calls Algorithm 3.2, and hence the experimental results coincide.) From [13, Thm. 8.9], the relative condition number of the polar factor U is

$$\kappa_U = \frac{2\sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2}}{\sqrt{3}(\sigma_2 + \sigma_3)}, \tag{5.3}$$

which for this particular input is

$$\kappa_U = \sqrt{\frac{1 + 2y^2}{3y^2}}.$$

Table 2 reports the relative forward errors $\|\hat{U} - U\|_F / \|U\|_F = \|\hat{U} - U\|_F / \sqrt{3}$, where $\hat{U}$ is the computed orthogonal factor in the polar decomposition of A while U is a reference result obtained by running the SVD algorithm in high precision arithmetic. The results show that, unlike Algorithm 3.2, Algorithm 3.5 and `polar-SVD` exhibit forward stable behaviour for all y.

We have performed further experiments comparing the accuracy of Algorithm 3.1, Algorithm 3.5, and `polar_svd` on random matrices with specified singular value distributions. Each experiment was repeated on a sample of 10000 matrices. The results are shown in Tables 3, 4, 5, 6. In each case the first line in the table reports the worst-case relative error $\|\hat{H} - H\|_F / \|H\|_F$ in the computed symmetric positive semidefinite factor $\hat{H}$ for the three codes, where H is a reference result. Note that H is always a well conditioned function of A [13, Thm. 8.9]. The second line (included in all but the last experiment) reports the worst-case relative error for the computed orthogonal polar factor $\hat{U}$, $\|\hat{U} - U\|_F / \|U\|_F = \|\hat{U} - U\|_F / \sqrt{3}$. Depending on the conditioning of the problem, given by (5.3), the forward error in U may be large even

Table 5 Singular values prescribed to be equal to 1, 10^{-10}, 10^{-13}. Here, $\omega = 10$, and Algorithm 3.5 will need to apply subspace iteration. The condition number of U is $\kappa_U \approx 1.15 \times 10^{10}$

	Algorithm 3.1	Algorithm 3.5	`polar_svd`
Forward error in H	2.0e-15	1.7e-15	9.6e-16
Forward error in U	8.3e-6	1.4e-6	2.9e-6
Backward error	4.2e-15	1.6e-15	1.8e-15

Table 6 Singular values prescribed to be equal to 1, 0, 0 (rank 1 matrix). Here, U is not uniquely defined so we do not report any forward error for its computation

	Algorithm 3.1	Algorithm 3.5	`polar_svd`
Forward error in H	2.8e-15	4.0e-15	1.1e-15
Backward error	4.7e-15	2.4e-15	1.8e-15

for a backward stable method. The last line reports the worst-case backward error $\|A - \hat{U}\hat{H}\|_F/\|A\|_F$. We observed that, for each of the measures that we analyzed, the average case error was typically 5 to 10 times smaller than the worst case error for each code.

From the results of the experiments we may conclude that the proposed approach yields results of comparable accuracy to `polar_svd`, with an advantage in computation time in the most common situation of an input with large σ_2.

6 Conclusions

The standard algorithms in numerical linear algebra are designed to be efficient for large matrices and are not necessarily optimal for small ones. Moreover, standard software relies on libraries such the Basic Linear Algebra Subprograms and LAPACK that are not necessary for small dimensions. Our new algorithm for computing the polar decomposition of a 3×3 real matrix, Algorithm 3.5, exploits a connection between orthogonal matrices and quaternions to reduce the problem to computing the dominant eigenvector of a 4×4 symmetric matrix. The algorithm has a flop count generally lower than the alternative of obtaining the polar decomposition from the SVD and it is faster than the SVD approach, as well as several other alternatives, in our numerical experiments. Moreover, the algorithm does not rely on library routines for computing eigenvalues or singular values. The algorithm therefore satisfies all the requirements of the Web CSS application that motivated this work, and in the usual case in this application in which the matrix is well conditioned the algorithm reduces to the simpler form of Algorithm 3.2.

Acknowledgments We thank Ken Shoemake for introducing us to the Cascading Style Sheets application and Weijian Zhang for his help with the Julia implementation of Algorithm 3.2.

References

1. Bar-Itzhack, I.Y.: New method for extracting the quaternion from a rotation matrix. J. Guidance **23**(6), 1085–1087 (2000)

2. Bezanson, J., Karpinski, S., Shah, V.B., Edelman, A.: Julia: A fast dynamic language for technical computing (2012). http://arxiv.org/abs/1209.5145
3. Bouby, C., Fortuné, D., Pietraszkiewicz, W., Vallée, C.: Direct determination of the rotation in the polar decomposition of the deformation gradient by maximizing a Rayleigh quotient. Z. Angew. Math. Mech. **85**(3), 155–162 (2005)
4. Cayley, A.: On certain results relating to quaternions. Philos. Mag. **26**(171), 141–145 (1845)
5. Coxeter, H.S.M.: Quaternions and reflections. Amer. Math. Monthly **53**(3), 136–146 (1946)
6. Demmel, J.W., William, K.: Accurate singular values of bidiagonal matrices. SIAM J. Sci. Statist. Comput. **11**(5), 873–912 (1990)
7. Dong, T., Haidar, A., Luszczek, P., Harris, J.A., Tomov, S., Dongarra, J.: LU factorization of small matrices: Accelerating batched DGETRF on the GPU. In: 2014 IEEE International Conference on High Performance Computing and Communications, 2014 IEEE 6th International Symposium on Cyberspace Safety and Security, and 2014 IEEE 11th International Conference on Embedded Software and Systems, IEEE Computer Society, pp. 157–160 (2014)
8. William Rowan Hamilton: On quaternions. Proc. Royal Irish Academy **3**, 1–16 (1847)
9. Higham, N.J.: The Matrix Function Toolbox., http://www.maths.manchester.ac.uk/~higham/mftoolbox
10. Higham, N.J.: Computing the polar decomposition—with applications. SIAM J. Sci. Statist. Comput. **7**(4), 1160–1174 (1986)
11. Higham, N.J.: Analysis of the Cholesky decomposition of a semi-definite matrix. In: Reliable Numerical Computation, M. G. Cox and S. J. Hammarling, editors, Oxford University Press, pp. 161–185 (1990)
12. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. xxx+680 pp. ISBN 0-89871-521-0
13. Higham, N.J.: Functions of Matrices: Theory and Computation. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008. xx+425 pp. ISBN 978-0-898716-46-7
14. Horn, R.A., Piazza, G., Politi, T.: Explicit polar decompositions of complex matrices. Electron. J. Linear Algebra **18**, 693–699 (2009)
15. Johnson, C.R., Laffey, T., Li, C.-K.: Linear transformations on $M_n(\mathbb{R})$ that preserve the Ky Fan k-norm and a remarkable special case when $(n, k) = (4, 2)$. Linear and Multilinear Algebra **23**(4), 285–298 (1988)
16. Kopp, J.: Efficient numerical diagonalization of Hermitian 3×3 matrices. Int. J. Mod. Phys. C **19**(3), 523–548 (2008)
17. Mackey, N.: Hamilton and Jacobi meet again: Quaternions and the eigenvalue problem. SIAM J. Matrix Anal. Appl. **16**(2), 421–435 (1995)
18. Mao, J.: Optimal orthonormalization of the strapdown matrix by using singular value decomposition. Computers Math. Applic. **12A**(3), 353–362 (1986)
19. McAdams, A., Zhu, Y., Selle, A., Empey, M., Tamstorf, R., Teran, J., Sifakis, E.: Efficient elasticity for character skinning with contact and collisions. ACM Trans. Graph. **11**(4), 37:1–37 (2011)
20. Nakatsukasa, Y., Bai, Z., Gygi, F.: Optimizing Halley's iteration for computing the matrix polar decomposition. SIAM J. Matrix Anal. Appl. **31**(5), 2700–2720 (2010)
21. Nakatsukasa, Y., Higham, N.J.: Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. SIAM J. Sci. Comput. **35**(3), A1325–A1349 (2013)
22. Ochiai, H., Anjyo, K.: Mathematical basics of motion and deformation in computer graphics. SIGGRAPH 2014 course notes. In: ACM SIGGRAPH 2014 Courses, SIGGRAPH '14, 2014, pages 19:1–19:47
23. Pan, V.Y., Yu, Y.: Certification of numerical computation of the sign of the determinant of a matrix. Algorithmica **30**(4), 708–724 (2001)
24. Pitsianis, N., Van Loan, C.F.: Vectorization of multiple small matrix problems. Technical Report CTC 94 TR172, Advanced Computing Research Institute, Cornell Theory Center, Cornell University, Ithaca, New York, March 1994. 13 pp
25. Rodman, L.: Topics in Quaternion Linear Algebra. Princeton University Press, NJ (2014). xii+363 pp. ISBN 978-0-691-16185-3
26. Shmakov, S.L.: A universal method of solving quartic equations. Int. J. Pure Appl. Math. **71**(2), 251–259 (2011)
27. Shoemake, K.: Private communication (2014)

28. Shoemake, K., Duff, T.: Matrix animation and polar decomposition. In: Proceedings of the Conference on Graphics Interface ’92, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992, pages 258–264
29. Uhlig, F.: Explicit polar decomposition and a near-characteristic polynomial: The 2×2 case. Linear Algebra Appl. **38**, 239–249 (1981)
30. van Den Driessche, P., Wimmer, H.K.: Explicit polar decomposition of companion matrices. Electron. J. Linear Algebra **1**, 64–69 (1996)
31. World Wide Web Consortium (W3C). CSS transforms module level 1. W3C working draft, 26 (2013). http://www.w3.org/TR/css3-transforms
32. Ye, X., Zhang, J., Li, P.: A hybrid rotational matrix extraction method for soft tissue deformation modeling. In: Proceedings of the 2015 IEEE International Conference on Mechatronics and Automation, pp. 727–732 (2015)