Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

ELA
http:/repository.uwyo.edu/ela

ON THE MATRIX SQUARE ROOT VIA GEOMETRIC OPTIMIZATION*

SUVRIT SRA†

Abstract. This paper is triggered by the preprint [P. Jain, C. Jin, S.M. Kakade, and P. Netrapalli. Computing matrix squareroot via non convex local search. Preprint, arXiv:1507.05854, 2015.], which analyzes gradient-descent for computing the square root of a positive definite matrix. Contrary to claims of Jain et al., the author's experiments reveal that Newton-like methods compute matrix square roots rapidly and reliably, even for highly ill-conditioned matrices and without requiring commutativity. The author observes that gradient-descent converges very slowly primarily due to tiny step-sizes and ill-conditioning. The paper derives an alternative first-order method based on geodesic convexity; this method admits a transparent convergence analysis ($<$ 1 page), attains linear rate, and displays reliable convergence even for rank deficient problems. Though superior to gradient-descent, ultimately this method is also outperformed by a well-known scaled Newton method. Nevertheless, the primary value of the paper is conceptual: it shows that for deriving gradient based methods for the matrix square root, the manifold geometric view of positive definite matrices can be much more advantageous than the Euclidean view.

Key words. Geometric optimization, Matrix square root, Newton method, Geodesic convexity.

AMS subject classifications. 32F99, 15A16, 65F60, 90C25, 90C26.

1. Introduction.

Matrix functions such as A^α, $\log A$, or e^A ($A \in \mathbb{C}^{n\times n}$) arise in diverse contexts. Higham [9] provides an engaging overview of such matrix functions. A building block for efficient numerical evaluation of various matrix functions is the *matrix square root* $A^{1/2}$, a function whose computation has witnessed great interest in the literature [1, 2, 8, 10, 12, 20]; see also [9, Chapter 6].

The present paper revisits the problem of computing the matrix square root for a symmetric positive semidefinite (psd) matrix. (The same ideas extend trivially to the Hermitian psd case.) Our work is triggered by the recent note of Jain et al. [14], who analyze a simple gradient-descent procedure for computing $A^{1/2}$ for a given psd matrix A. Jain et al. [14] motivate their work by citing weaknesses of the simplified Newton method that is known to unstable and can even diverge [10].

However, it turns out that the modified "polar-Newton" (PN) method of [9, Algorithm 6.21] avoids these drawbacks, and offers a method with excellent practical

*Received by the editors on December 16, 2015. Accepted for publication on May 10, 2016. Handling Editor: James G. Nagy.

†Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, 02139, USA (suvrit@mit.edu).

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

behavior. At the same time, gradient-descent of [14], while initially appealing due to its simplicity, turns out to be empirically unacceptable: it converges extremely slowly, especially on ill-conditioned matrices because its stepsizes become too small to ensure progress on computers with limited floating point computation.

At the expense of introducing "yet another matrix square root" (YAMSR) method, we present a non-Euclidean first-order method grounded in the geometric optimization framework of [23]. Our method admits a transparent convergence analysis, attains linear (geometric) convergence rate, and displays reliable behavior even for highly ill-conditioned problems (including rank-deficient problems). Although numerically superior to the gradient-descent approach of [14], YAMSR is still outperformed by the well-known polar-Newton (scaled Newton) method.

In light of these observations, the two key messages delivered by our paper are:

- None of the currently known (to us) first-order methods are competitive with the established second-order polar-Newton algorithm for matrix square roots.[1]
- For deriving gradient based methods that compute matrix square roots, *the differential geometric view of positive definite matrices is much more advantageous than the Euclidean view.*

As a further attestation to the latter claim, our analysis yields as a byproduct a short convergence proof of a Euclidean iteration due to [2], which previously required a more intricate analysis. Moreover, it yields a geometric rate of convergence.

Finally, we note that the above conceptual view underlies several other applications too, where the geometric view of positive definite matrices has also proved quite useful, e.g., [7, 11, 22, 23, 25, 26, 27].

1.1. Summary of existing methods. To place our statements in wider perspective we cite below several methods for matrix square roots. We refer the reader to [9, Chapters 6 and 8] for more details and references, as well as a broader historical perspective. For a condensed summary, we refer the reader to the survey of Iannazzo [12] which covers the slightly more general problem of computing the matrix geometric mean.

1. **Eigenvectors**: Compute the decomposition $A = U\Lambda U^*$ and obtain $A^{1/2} = U\Lambda^{1/2}U^*$. While tempting, this can be slower for large and ill-conditioned matrices. In our case, since $A \succ 0$, this method coincides with the Schur-Cholesky method mentioned in [12].
2. **Matrix averaging**: Many methods exist that obtain the matrix geometric mean as a limit of averaging procedures. These include scaled averaging

[1] Presumably, the same conclusion may hold for many other matrix functions.

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

(which is essentially a matrix Harmonic-Arithmetic mean iteration that converges to the geometric mean) and its variants [12, Section 5.1]. In spirit, the algorithm that we propose in this paper also falls in this category.

3. **Newton-like**: The classic Newton iteration of [10], whose weaknesses motivate the gradient-descent procedure of [14]. However, a better way to implement this iteration is to use polar decomposition. Here, first we compute the Cholesky factorization $A = R^T R$, then obtain the square root as $A^{1/2} = UR$, where U is the unitary polar factor of R^{-1}. This polar factor could be computed using the SVD of R, but the polar-Newton (PN) iteration [9, Algorithm 6.21] avoids that and instead uses the *scaled Newton method*

$$U_{k+1} = \tfrac{1}{2}\left(\mu_k U_k + \mu_k^{-1} U_k^{-T}\right), \qquad U_0 = R.$$

This iteration has the benefit that the number of steps needed to attain a desired tolerance can be predicted in advance, and it can be implemented to be backward stable. It turns out that empirically this iteration works reliably even for extremely ill-conditioned matrices, and converges (locally) superlinearly. The scaled Halley-iteration of [20] converges even more rapidly and can be implemented in a numerically stable manner [19].

4. **Quadrature**: Methods based on Gaussian and Gauss-Chebyshev quadrature and rational minimax approximations to $z^{-1/2}$ are surveyed in [12, Section 5.4]. Among others, these methods easily tackle computation of general powers, logarithms, and some other matrix functions [8, 13].
5. **First-order methods**: The binomial iteration uses $(I-C)^{1/2} = I - \sum_j \alpha_j C^j$ for suitable $\alpha_j > 0$ and needs $\rho(C) < 1$ [9, Section 6.8.1]. The gradient-descent method of [14] minimize $\|X^2 - A\|_F^2$. Like the binomial iteration, this method does not depend on linear system solves (or matrix inversion) and uses only matrix multiplication. The method introduced in Section 2.1 is also a first-order method, though cast in a non-Euclidean space; it does, however, require (Cholesky based) linear system solves.

Additional related work includes fast computation of a matrix C such that $A^p \approx CC^T$ for SDD matrices [6]; various algorithms for computing the product $f(A)b$ [9, Chapter 13]; and the broader topic of multivariate matrix means [3, 5, 15, 17, 21, 22].

2. Geometric optimization. We present details of computing the matrix square root using geometric optimization. Specifically, we cast the problem of computing matrix square roots into the nonconvex optimization problem

$$\min_{X \succ 0} \quad \delta_S^2(X, A) + \delta_S^2(X, I), \tag{2.1}$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

whose unique solution is the desired matrix square root $X^* = A^{1/2}$; here δ_S^2 denotes the *S-Divergence* [22]:

$$\delta_S^2(X,Y) := \log\det\left(\tfrac{X+Y}{2}\right) - \tfrac{1}{2}\log\det(XY). \tag{2.2}$$

To present our algorithm for solving (2.1) let us first recall some background.

The crucial property that helps us solve (2.1) is *geodesic convexity* of δ_S^2, that is, convexity along geodesics in $\mathbb{P}^n$ endowed with its usual Riemannian geometry (see e.g., [4, Chapter 6]). The geodesic joining X to Y is given by

$$X\#_t Y := X^{1/2}(X^{-1/2}YX^{-1/2})^t X^{1/2}, \quad X, Y \succ 0, \quad t \in [0,1], \tag{2.3}$$

and a function $f : \mathbb{P}^n \to \mathbb{R}$ is called *geodesically convex* (*g-convex*) if

$$f(X\#_t Y) \le (1-t)f(X) + tf(Y). \tag{2.4}$$

THEOREM 2.1. *The S-Divergence* (2.2) *is jointly g-convex on psd matrices.*

Proof. See [22, Theorem 4.4]; we include a proof in the appendix for the reader's convenience. □

Similar to Euclidean convexity, g-convexity bestows the crucial property: "Local $\implies$ global". Consequently, we may use any algorithm that ensures local optimality; g-convexity will imply global optimality. We present one such algorithm below.

But first let us see why solving (2.1) yields the desired matrix square root.

THEOREM 2.2. *Let $A, B \succ 0$. Then,*

$$A\#_{1/2}B = \operatorname{argmin}_{X\succ 0} \quad \delta_S^2(X,A) + \delta_S^2(X,B). \tag{2.5}$$

Moreover, $A\#_{1/2}B$ is equidistant from A and B.

Proof. See [22, Theorem 4.1]. We include a proof below for convenience.

Since the constraint set is open, we can simply differentiate the objective in (2.5), and set the derivative to zero to obtain the necessary condition

$$\begin{aligned}
&\left(\tfrac{X+A}{2}\right)^{-1}\tfrac{1}{2} + \left(\tfrac{X+B}{2}\right)^{-1}\tfrac{1}{2} - X^{-1} = 0,\\
\implies \quad & X^{-1} = (X+A)^{-1} + (X+B)^{-1}\\
\implies \quad & (X+A)X^{-1}(X+B) = 2X + A + B\\
\implies \quad & B = XA^{-1}X.
\end{aligned}$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

The latter is a Riccati equation whose *unique* positive solution is $X = A\#_{1/2}B$—see [4, Proposition 1.2.13][2]. Global optimality of this solution follows easily since δ_S^2 is g-convex as per Theorem 2.1. □

COROLLARY 2.3. *The unique solution to* (2.1) *is* $X^* = A\#_{1/2}I = A^{1/2}$.

2.1. Algorithm for matrix square root. We present now an iteration for solving (2.1). As for Theorem 2.2, we obtain here the optimality condition

$$X^{-1} = (X + A)^{-1} + (X + I)^{-1}. \tag{2.6}$$

Our algorithm solves (2.6) simply by running[3]

$$X_0 = \tfrac{1}{2}(A + I) \tag{2.7}$$
$$X_{k+1} \leftarrow [(X_k + A)^{-1} + (X_k + I)^{-1}]^{-1}, \qquad k = 0, 1, \ldots \tag{2.8}$$

The following facts about our algorithm and its analysis are noteworthy:

1. Towards completion of this article we discovered that iteration (2.8) has been known in matrix analysis since over three decades! Indeed, motivated by electrical resistance networks, Ando [2] analyzed exactly the same iteration as (2.8) in 1981. Our convergence proof (Theorem 2.4) is much shorter and more transparent—it not only reveals the geometry behind its convergence but also yields explicit bounds on the convergence rate, thus offering a new understanding of the classical work [2].
2. Initialization (2.7) is just one of many. Any matrix in the interval $[2(I + A^{-1})^{-1}, \frac{1}{2}(A+I)]$ is valid too; different choices can lead to faster convergence.
3. Each iteration of the algorithm involves three matrix inverses. Theoretically, this costs "just" $O(n^\omega)$ operations (where ω denotes the exponent denoting the complexity of matrix multiplication). In practice, we compute (2.8) using linear system solves; in MATLAB notation:
   ```
   R = (X+A)\I + (X+I)\I; X = R\I;
   ```
4. The gradient-descent method in [14] and the binomial method [9] do not require solving linear systems, and rely purely on matrix multiplication. But both turn out to be slower than (2.8) while also being more sensitive to ill-conditioning.
5. For ill-conditioned matrices, it is better to iterate (2.8) with αI, for a suitable scalar $\alpha > 0$; the final solution is recovered by dowscaling by $\alpha^{-1/2}$. A heuristic choice is $\alpha = \mathrm{tr}(A)/\sqrt{n}$, which seems to work well in practice (for well-conditioned matrices $\alpha = 1$ is preferable).

[2]There is a typo in the cited result; it uses A and $A^{1/2}$ where it should use A^{-1} and $A^{-1/2}$.
[3]Observe that the same algorithm also computes $A\#_{1/2}B$ if we replace I by B in (2.7) and (2.8).

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

THEOREM 2.4 (Convergence). *Let $\{X_k\}_{k\geq 0}$ be generated by (2.7)–(2.8). Let $X^* = A^{1/2}$ be the optimal solution to* (2.1)*. Then, there exists a constant $\gamma < 1$ such that $\delta_S^2(X_k, X^*) \leq \gamma^k \delta_S^2(X_0, X^*)$. Moreover, $\lim X_k = X^*$.*

Proof. Our proof is a specialization of the fixed-point theory in [18, 23]. Specifically, we prove that (2.8) is a fixed-point iteration under the *Thompson part metric*

$$\delta_T(X, Y) := \|\log(X^{-1/2} Y X^{-1/2})\|, \tag{2.9}$$

where $\|\cdot\|$ is the usual operator norm. The metric (2.9) satisfies many remarkable properties; for our purpose, we use the fact that it is complete [24], as well the following three well-known properties (e.g., [23, Proposition 4.2]):

$$\begin{aligned} \delta_T(X^{-1}, Y^{-1}) &= \delta_T(X, Y) \\ \delta_T(X + A, Y + B) &\leq \max\{\delta_T(X, Y), \delta_T(A, B)\}, \\ \delta_T(X + A, Y + A) &\leq \tfrac{\alpha}{\alpha + \lambda_{\min}(A)} \delta_T(X, Y), \quad \alpha = \max\{\|X\|, \|Y\|\}. \end{aligned} \tag{2.10}$$

Consider now the nonlinear map

$$\mathcal{G} \equiv X \mapsto \left[\tfrac{1}{2}\Big(\tfrac{X+A}{2}\Big)^{-1} + \tfrac{1}{2}\Big(\tfrac{X+I}{2}\Big)^{-1}\right]^{-1} \equiv [(X + A)^{-1} + (X + I)^{-1}]^{-1},$$

corresponding to iteration (2.8). Using properties (2.10) of the metric δ_T we have

$$\begin{aligned} \delta_T(\mathcal{G}(X), \mathcal{G}(Y)) &= \delta_T([(X + A)^{-1} + (X + I)^{-1}]^{-1}, [(Y + A)^{-1} + (Y + I)^{-1}]^{-1}) \\ &= \delta_T((X + A)^{-1} + (X + I)^{-1}, (Y + A)^{-1} + (Y + I)^{-1}) \\ &\leq \max\big\{\delta_T((X + A)^{-1}, (Y + A)^{-1}), \delta_T((X + I)^{-1}, (Y + I)^{-1})\big\} \\ &= \max\{\delta_T(X + A, Y + A), \delta_T(X + I, Y + I)\} \\ &\leq \max\{\gamma_1 \delta_T(X, Y), \gamma_2 \delta_T(X, Y)\} \\ &\qquad (\text{where } \gamma_1 = \tfrac{\alpha}{\alpha + \lambda_{\min}(A)},\ \gamma_2 = \tfrac{\alpha}{\alpha+1},\ \alpha = \max\{\|X\|, \|Y\|\}) \\ &\leq \gamma \delta_T(X, Y), \qquad \gamma < 1, \end{aligned}$$

where we can choose γ to be independent of X and Y. Thus, the map $\mathcal{G}$ is a strict contraction. Hence, from the Banach contraction theorem it follows that $\delta_T(X_k, X^*)$ converges at a linear rate given by γ, and that $X^k \to X^*$. Using operator convexity of the map X^{-1} a brief manipulation shows that $\mathcal{G}$ maps the (compact) interval $[2(I + A^{-1})^{-1}, \frac{1}{2}(A + I)]$ to itself. Thus, $\alpha \leq \frac{1}{2}\|I + A\|$; since $\frac{\alpha}{c+\alpha}$ is increasing, we can easily upper-bound $\gamma = \max(\gamma_1, \gamma_2)$ to finally obtain

$$\gamma \leq \frac{\alpha}{\alpha + \min(\lambda_{\min}, 1)} \leq \frac{1 + \|A\|}{1 + \|A\| + 2\min(\lambda_{\min}(A), 1)}.$$

This is strictly smaller than 1 if $A \succ 0$, thus yielding an explicit contraction rate. ∎

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

REMARK 1. The bound on γ above is a worst-case bound. Empirically, the value of γ is usually much smaller and the convergence rate commensurately faster.

REMARK 2. Starting with $X_0 = \frac{1}{2}(A+I)$ ensures that $X_0 \succ 0$; thus, we can use the iteration (2.8) even if A is semidefinite, as all intermediate iterates remain well-defined. This suggests why iteration (2.8) is empirically robust against ill-conditioning.

EXAMPLE 2.5. Suppose $A = 0$. Then, the map $\mathcal{G}$ is no longer contractive but still nonexpansive. Iterating (2.8) generates the sequence $\{X_k\} = \{\frac{1}{2}I, \frac{3}{8}I, \frac{33}{112}I, \ldots\}$, which converges to zero.

Example 2.5 shows that iteration (2.8) remains well-defined even for the zero matrix. This prompts us to take a closer look at computing square roots of low-rank semidefinite matrices. In particular, we extend the definition of the S-Divergence to low-rank matrices. Let A be a rank-r semidefinite matrix of size $n \times n$ where $r \leq n$. Then, define $\det_r(A) := \prod_{i=1}^{r} \lambda_i(A)$ to be product of its r positive eigenvalues. Using this, we can extend (2.2) as

$$\delta_{S,r}^2(X,Y) := \log\det_r\left(\tfrac{X+Y}{2}\right) - \tfrac{1}{2}\log\det_r(X) - \tfrac{1}{2}\log\det_r(Y) \qquad (2.11)$$

for rank-r SPD matrices X and Y. If $\operatorname{rank}(X) \neq \operatorname{rank}(Y)$, we set $\delta_{S,r}(X,Y) = +\infty$. The above definition can also be obtained as a limiting form of (2.2) by applying δ_S^2 to rank deficient X and Y by considering $X + \epsilon I$ and $Y + \epsilon I$ and letting $\epsilon \to 0$.

Although iteration (2.8) works remarkably well in practice, it is a question of future interest on how to obtain square roots for rank deficient matrices faster by exploiting (2.11) instead.

3. Numerical results. We present numerical results comparing running times and accuracy attained by: (i) YAMSR; (ii) GD; (iii) LSGD; and (iv) PN. These methods respectively refer to iteration (2.8), the fixed step-size gradient-descent procedure of [14], our line-search based implementation of gradient-descent, and the polar-Newton iteration of [9, Algorithm 6.21].

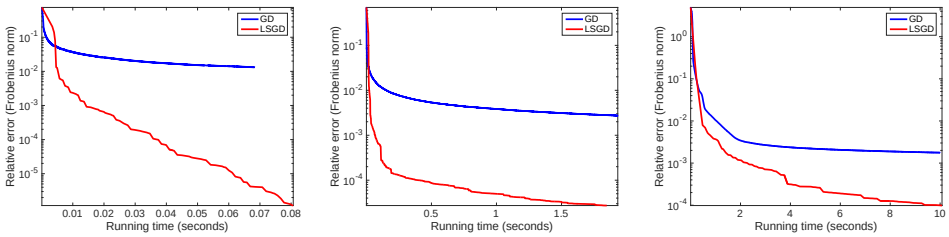
We present only one experiment with GD, because with fixed steps it vastly underperforms all other methods. In particular, if we set the step size according to the results in [14], then for matrices with large condition numbers the step size becomes smaller than machine precision! In our experiments with GD we used step sizes much larger than theoretical ones, otherwise the method makes practically no progress. More realistically, if we wish to use gradient-descent we must employ line-search. Ultimately, although substantially superior to plain GD, even LSGD turns out to be outperformed by YAMSR.

We experiment with the following matrices (generated in MATLAB):

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

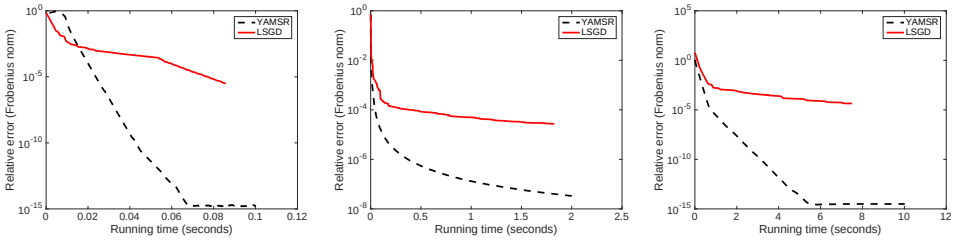
ELA
http:/repository.uwyo.edu/ela

1. $I + \beta UU'$ for a low-rank matrix U and a variable constant β. These matrices are well-conditioned.
2. Random correlation matrices (`gallery('randcorr', n)`); medium conditioned.
3. The Hilbert matrix (`hilb(n)`), a well-known ill-conditioned matrix.
4. The inverse Hilbert matrix (`invhilb(n)`). The entries of the inverse Hilbert matrix are very large integers. Extremely ill-conditioned.



50×50 matrix $I + \beta UU^T$ | 50×50 Hilbert matrix | 500×500 Covariance

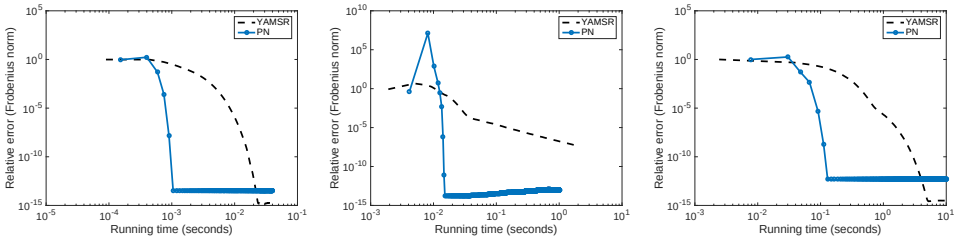
FIG. 3.1. *Running times:* GD *versus* LSGD*; this behavior is typical.*



50×50 matrix $I + \beta UU^T$ | 50×50 Hilbert matrix | 500×500 Covariance

$\kappa \approx 64$ | $\kappa \approx 2.8 \times 10^{18}$ | $\kappa \approx 832$

FIG. 3.2. *Running times:* YAMSR *vs* LSGD *(κ is condition number).*



50×50 matrix $I + \beta UU^T$ | 100×100 Hilbert matrix | 500×500 Covariance

FIG. 3.3. *Running time:* YAMSR *vs* PN*; this behavior is also typical.*

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

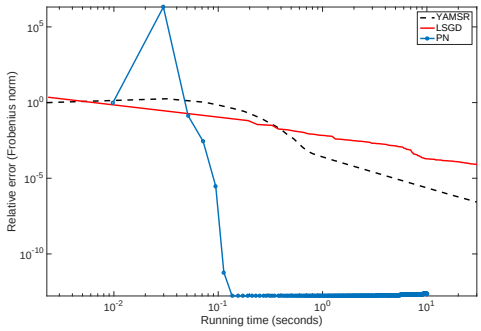


FIG. 3.4. *Running time:* YAMSR, LSGD, *and* PN *for computing square root of a* 500×500 *low-rank (rank 50) covariance matrix.*

4. Conclusions. We revisited computation of the matrix square root, and compared the recent gradient-descent procedure of [14] against the polar-Newton method of [9, Algorithm 6.21], as well as YAMSR, a first-order fixed-point algorithm that we derived using the viewpoint of geometric optimization. The experimental results show that the polar-Newton method is the clear winner for computing matrix square roots (except near the 10^{-15} accuracy level for well-conditioned matrices). Among the first-order methods YAMSR outperforms both gradient-descent as well its line-search variant across all tested settings, including singular matrices.

Acknowledgment. We thank an anonymous referee for helping making our description of [10] more precise.

REFERENCES

[1] W.N. Anderson Jr., T.D. Morley, and G.E. Trapp. Ladder networks, fixpoints, and the geometric mean. *Circuits, Systems and Signal Processing*, 2(3):259–268, 1983.
[2] T. Ando. Fixed points of certain maps on positive semidefinite operators. In: P. Butzer, B.Sz.-Nagy, and E. Görlich (editors), *Functional Analysis and Approximation*, International Series of Numerical Mathematics, Birkhäuser Basel, Berlin, 60:29–38, 1981.
[3] T. Ando, C.-K. Li, and R. Mathias. Geometric means. *Linear Algebra and its Applications*, 385:305–334, 2004.
[4] R. Bhatia. *Positive Definite Matrices*. Princeton University Press, Princeton, 2007.
[5] D.A. Bini and B. Iannazzo. Computing the Karcher mean of symmetric positive definite matrices. *Linear Algebra and its Applications*, 438(4):1700–10, 2013.
[6] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.-H. Teng. Efficient Sampling for gaussian graphical models via spectral sparsification. *Proceedings of the Conference on Learning Theory*, 364–390, 2015.
[7] A. Cherian and S. Sra. Riemannian dictionary learning and sparse coding for positive definite matrices. Preprint, 2014.

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

[8] N. Hale, N. J. Higham, and L. N. Trefethen. Computing a^α, $\log(a)$, and related matrix functions by contour integrals. *SIAM Journal on Numerical Analysis*, 46(5):2505–2523, 2008.
[9] N.J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.
[10] N.J. Higham. Newton's method for the matrix square root. *Mathematics of Computation*, 46(174):537–549, 1986.
[11] R. Hosseini and S. Sra. Matrix manifold optimization for Gaussian mixture models. *Advances in Neural Information Processing Systems (NIPS 2015)*, 2015.
[12] B. Iannazzo. The geometric mean of two matrices from a computational viewpoint. *Numerical Linear Algebra with Applications*, 23(2):208–229, 2015.
[13] B. Iannazzo and B. Meini. Palindromic matrix polynomials, matrix functions and integral representations. *Linear Algebra and its Applications*, 434(1):174–184, 2011.
[14] P. Jain, C. Jin, S.M. Kakade, and P. Netrapalli. Computing matrix squareroot via non convex local search. Preprint, arXiv:1507.05854, 2015. URL `http://arxiv.org/abs/1507.05854`.
[15] B. Jeuris, R. Vandebril, and B. Vandereycken. A survey and comparison of contemporary algorithms for computing the matrix geometric mean. *Electronic Transactions on Numerical Analysis*, 39:379–402, 2012.
[16] F. Kubo and T. Ando. Means of positive linear operators. *Mathematische Annalen*, 246:205–224, 1980.
[17] J. Lawson and Y. Lim. A general framework for extending means to higher orders. *Colloquium Mathematicum*, 113:191–221, 2008.
[18] H. Lee and Y. Lim. Invariant metrics, contractions and nonlinear matrix equations. *Nonlinearity*, 21:857–878, 2008.
[19] Y. Nakatsukasa and N.J. Higham. Backward stability of iterations for computing the polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 33(2):460–479, 2012.
[20] Y. Nakatsukasa, Z. Bai, and F. Gygi. Optimizing Halley's iteration for computing the matrix polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2700–2720, 2010.
[21] F. Nielsen and R. Bhatia (editors). *Matrix Information Geometry*. Springer, 2013.
[22] S. Sra. Positive definite matrices and the S-divergence. *Proceedings of the American Mathematical Society*, 144:2787–2797, 2016.
[23] S. Sra and R. Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM Journal on Optimization*, 25(1):713–739, 2015.
[24] A.C. Thompson. On certain contraction mappings in partially ordered vector space. *Proceedings of the American Mathematical Society*, 14:438–443, 1963.
[25] A. Wiesel. Geodesic convexity and covariance estimation. *IEEE Transactions on Signal Processing*, 60(12):6182–89, 2012.
[26] A. Wiesel. Unified framework to regularized covariance estimation in scaled Gaussian models. *IEEE Transactions on Signal Processing*, 60(1):29–38, 2012.
[27] T. Zhang. Robust subspace recovery by geodesically convex optimization. Preprint, arXiv:1206.1386, 2012.

Appendix A. Technical details.

Proof of Theorem 2.1. It suffices to prove midpoint convexity; the general case follows by continuity. Consider therefore psd matrices X_1, X_2, Y_1, Y_2. We need to show that

$$\delta_S^2(X_1 \#_{1/2} X_2, Y_1 \#_{1/2} Y_2) \leq \tfrac{1}{2}\delta_S^2(X_1, Y_1) + \tfrac{1}{2}\delta_S^2(X_2, Y_2). \tag{A.1}$$

Electronic Journal of Linear Algebra ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 31, pp. 433-443, June 2016

From the joint concavity of the operator $\#_{1/2}$ [16], we know that

$$X_1\#_{1/2}X_1 + Y_1\#_{1/2}Y_2 \preceq (X_1+Y_1)\#_{1/2}(X_2+Y_2).$$

Since $\log\det$ is a monotonic function, applying it to this inequality yields

$$\log\det[X_1\#_{1/2}X_1 + Y_1\#_{1/2}Y_2] \leq \log\det[(X_1+Y_1)\#_{1/2}(X_2+Y_2)]. \tag{A.2}$$

But we also know that $\log\det((1-t)X\#_t tY) = (1-t)\log\det(X) + t\log\det(Y)$. Thus, a brief algebraic manipulation of (A.2) combined with the definition (2.2) yields inequality (A.1) as desired. □