Received: 1 January 2023 | Revised: 13 July 2023 | Accepted: 24 July 2023

DOI: 10.1002/nla.2528

RESEARCH ARTICLE

WILEY

Low-rank updates of matrix square roots

Shany Shmueli[1] | **Petros Drineas**[2] | **Haim Avron**[1]

[1]School of Mathematical Sciences,Tel Aviv University, Tel Aviv, Israel

[2]Department of Computer Science,Purdue University, West Lafayette, Indiana, USA

Correspondence
Haim Avron, Tel Aviv University, Israel.
Email: haimav@tauex.tau.ac.il

Funding information
Israel Science Foundation, Grant/Award Number: 1272/17; National Science Foundation, Grant/Award Numbers: 10001415, 10001390; United States - Israel Binational Science Foundation, Grant/Award Number: 2017698

Abstract

Models in which the covariance matrix has the structure of a sparse matrix plus a low rank perturbation are ubiquitous in data science applications. It is often desirable for algorithms to take advantage of such structures, avoiding costly matrix computations that often require cubic time and quadratic storage. This is often accomplished by performing operations that maintain such structures, for example, matrix inversion via the Sherman–Morrison–Woodbury formula. In this article, we consider the matrix square root and inverse square root operations. Given a low rank perturbation to a matrix, we argue that a low-rank approximate correction to the (inverse) square root exists. We do so by establishing a geometric decay bound on the true correction's eigenvalues. We then proceed to frame the correction as the solution of an algebraic Riccati equation, and discuss how a low-rank solution to that equation can be computed. We analyze the approximation error incurred when approximately solving the algebraic Riccati equation, providing spectral and Frobenius norm forward and backward error bounds. Finally, we describe several applications of our algorithms, and demonstrate their utility in numerical experiments.

KEYWORDS

low rank perturbations, low rank updates, matrix functions, matrix square root

1 | INTRODUCTION

In applications, and in particular data science applications, one often encounters matrices that are low-rank perturbations of another (perhaps simpler) matrix. For example, models in which the covariance matrix has the structure of a sparse matrix plus a low rank perturbation are common. In another example, it is common for algorithms to maintain a matrix that is iteratively updated by low-rank perturbations.

It is often desirable for algorithms to take advantage of such structures, avoiding costly matrix computations that often require cubic time and quadratic storage. An indispensable tool for utilizing low-rank perturbations is the famous Sherman–Morrison–Woodbury formula, which shows that the inverse of a low-rank perturbation can be obtained using a low-rank correction of the inverse. While the usefulness of the Sherman–Morrison–Woodbury formula cannot be understated, other matrix functions also frequently appear in applications. One naturally asks the following questions. Given a matrix function f, when does a low rank perturbation of a matrix $\mathbf{A}$ correspond to a (approximately) low-rank correction of $f(\mathbf{A})$? Can we find a high quality approximate correction efficiently, that is, without computing the exact correction and truncating it using a SVD?

In this article, we answer these questions affirmatively for two important and closely related functions: the matrix square root and inverse square root. The square root operation on matrices is a fundamental operation that frequently appears in mathematical analysis. Moreover, matrix square roots and their inverse arise frequently in data science applications, for example, when sampling from a high dimensional multivariate Gaussian distribution, or when whitening data. Computing the square root (or inverse square root) of low-rank perturbations of simple matrices (e.g., diagonal matrices) appear in quite a few data science applications, for example, in the aforementioned applications when the covariance matrix follows a spiked population model. For more discussion on applications of low-rank perturbations of matrix square root, see Section 6.

In particular, we show that given a low rank perturbation $\mathbf{D}$ to a matrix $\mathbf{A}$, we can approximate $(\mathbf{A}+\mathbf{D})^{1/2}$ well by a low rank correction to $\mathbf{A}^{1/2}$. We do so by proving a geometric decay bound on the eigenvalues of $(\mathbf{A}+\mathbf{D})^{1/2}-\mathbf{A}^{1/2}$. We also provide a similar bound for the inverse square root. We then proceed to show that the exact update is a solution of an algebraic Riccati equation, and discuss how a low-rank approximate solution to that equation can be computed. This allows us to propose concrete algorithms for updating and downdating the matrix square root and matrix inverse square root. Finally, we report experiments that corroborate our theoretical results.

1.1 | Related work

Most previous work on the matrix square root focused on computing $\mathbf{A}^{1/2}\mathbf{x}$ and $\mathbf{A}^{-1/2}\mathbf{x}$ for a given vector $\mathbf{x}$ using a Krylov method, possibly with preconditioning.[1-4] The motivation for most of the aforementioned works is sampling from a multivariate Gaussian distributions. Worth mentioning is recent work by Pleiss et al.[5] which combines a Krylov subspace method with a rational approximation, and also allows preconditioning.

The problem of updating a function of a matrix after a low rank perturbation, that is, computing $f(\mathbf{A}+\mathbf{D})$ given $f(\mathbf{A})$ where $\mathbf{D}$ is low-rank, has been recently receiving attention. The Sherman–Morrison–Woodbury formula is a well known formula for updating the matrix inverse, that is, $f(x)=x^{-1}$. Bernstein and Van Loan[6] showed that that when f is a rational function of degree q, a rank one perturbation of $\mathbf{A}$ corresponds to a rank q perturbation of $f(\mathbf{A})$. This article also provides an explicit formula for the low rank perturbation. Higham[1] showed that a rank k perturbation of $\mathbf{A}=\alpha\mathbf{I}$ corresponds to a rank k perturbation of $f(\mathbf{A})$ for *any* f (see Theorem 1.35 therein). As for inexact corrections, Beckermann et al.[7] proposed a Krylov method for computing a low-rank correction of $f(\mathbf{A})$ that approximates $f(\mathbf{A}+\mathbf{D})$ well for any analytic f (however, approximation quality depends on properties of the function itself, for example, how well it is approximated by a polynomial). In follow up work, they proposed a rational Krylov method,[8] citing the matrix square root as an example of a case in which their original method might have slow convergence.

The work most similar to ours is Reference 9. In that article, the authors consider the problem of computing the square root of a matrix of the form $\alpha\mathbf{I}+\mathbf{U}\mathbf{V}^T$, as a correction of $\sqrt{\alpha}\mathbf{I}$. Due to Reference 1 (Theorem 1.35), the rank of the correction is the same as the rank of $\mathbf{U}\mathbf{V}^T$. However, the formula in Reference 1 (Theorem 1.35) requires $\mathbf{V}^T\mathbf{U}$ to be non-singular, which is not required for the square root to be defined. The authors circumvent this issue by suggesting another formula for the square root, or by using a Newton iteration. In a way, the algorithm in Reference 9 is more general than our method since it allows non-symmetric updates. However, in another way it is less general: the matrix to be perturbed must be a scaled identity matrix. Moreover, we also suggest a method for updating the inverse square root.

2 | PRELIMINARIES

2.1 | Notation and basic definitions

We denote scalars using Greek letters or using $x, y, \ldots$. Vectors are denoted by $\mathbf{x}, \mathbf{y}, \ldots$ and matrices by $\mathbf{A}, \mathbf{B}, \ldots$. The $n \times n$ identity matrix is denoted $\mathbf{I}_n$. We use the convention that vectors are column-vectors.

Given a symmetric positive-semidefinite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, another matrix $\mathbf{B} \in \mathbb{R}^{n\times n}$ is a *square root* of $\mathbf{A}$ if $\mathbf{B}^2=\mathbf{A}$. There is a unique square root of $\mathbf{A}$ that is also positive semi-definite, which is called the *principal square root* and we denote it by $\mathbf{A}^{1/2}$.

Given two matrices $\mathbf{F}$ and $\mathbf{G}$, the $(\mathbf{F},\mathbf{G})$-displacement rank of $\mathbf{A}$ is defined as the rank of $\mathbf{F}\mathbf{A}-\mathbf{A}\mathbf{G}$. Displacement structures and displacement rank are closely connected to the Sylvester equation.

2.2 | Low-rank algebraic Riccati equation

Consider the following equation in $\mathbf{X} \in \mathbb{R}^{n\times n}$,

$$\mathbf{E}\mathbf{X} + \mathbf{X}\mathbf{E} + \alpha\mathbf{X}^2 = \mathbf{G}^T\mathbf{G}, \tag{1}$$

where $\mathbf{E} \in \mathbb{R}^{n\times n}$ is a symmetric full-rank matrix, $\mathbf{G} \in \mathbb{R}^{k\times n}$ where $k \ll n$, and $\alpha = \pm 1$. Our algorithms are based on approximately solving Equation (1) with a positive semidefinite low-rank $\mathbf{X}$.

If $\alpha = +1$, Equation (1) is an instance of the *algebraic Riccati equation*. There is a rich literature on algorithms for finding low rank solutions for the algebraic Riccati equation. We note the survey due to Benner and Saak,[10] and the book by Bini et al.[11] In our experiment, we use a recently proposed meta-scheme for approximately solving a slightly more general version of Equation (1) based on Riemannian optimization.[12] When applied to Equation (1) their scheme assumes the ability to take products of $\mathbf{E}$ by a vector, and to solve linear equations where the matrix is equal to $\mathbf{E}^2$ + low-rank, which is easily achievable via the Sherman–Morrison–Woodbury formula if we have access to an oracle that multiplies $\mathbf{E}^{-1}$ by a vector. Under the assumption that each rank update in Reference 12 requires $O(1)$ trust-region iterations, and that the target maximum rank of $\mathbf{X}$ is r, the overall cost of the scheme in Reference 12 is $O((T_{\mathbf{E}} + T_{\mathbf{E}^{-1}})r^2 + nr^4)$ where $T_{\mathbf{E}}$ and $T_{\mathbf{E}^{-1}}$ is the cost of multiplying $\mathbf{E}$ and $\mathbf{E}^{-1}$ by a vector (respectively). In most of our applications $\mathbf{E}$ is diagonal, so the cost reduces to $O(nr^4)$.

The method described in Reference 12 handles only the case of $\alpha = +1$. However, it can be generalized to the case of $\alpha = -1$. We give details in Appendix A.

In our algorithms, we denote the process of solving Equation (1) via the notation

$$\mathbf{U} \leftarrow \text{RiccatiLRSolver}(\mathbf{E}, \mathbf{G}, \alpha, r),$$

where r is the target maximum rank, and $\mathbf{U} \in \mathbb{R}^{n\times r}$ is a symmetric factor of the solution $\mathbf{X}$ (i.e., $\mathbf{X} = \mathbf{U}\mathbf{U}^T$). In the complexity analyses we assume that this process takes $O((T_{\mathbf{E}} + T_{\mathbf{E}^{-1}})r^2 + nr^4)$, as justified by the discussion above. However, we stress that our algorithm can use any algorithm for finding low-rank approximate solution to the algebraic Riccati equation, for example, we have also successfully used the M.E.S.S solver[1] for this purpose (experiments not reported). We use the algorithm from Reference 12 in our discussion and experiments due to the algorithm's clear complexity cost. We remark that our algorithm actually uses only the case $\alpha = +1$, but for some discussions it is useful to consider also the ability to solve for $\alpha = -1$.

2.3 | Problem statement

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a a symmetric positive semidefinite matrix. Suppose we are given $\mathbf{A}^{1/2}$, perhaps implicitly (i.e., as a function that maps a vector $\mathbf{x}$ to $\mathbf{A}^{1/2}\mathbf{x}$). Given a perturbation $\mathbf{D} \in \mathbb{R}^{n\times n}$ of rank $k \ll n$, our goal is to approximate $(\mathbf{A} + \mathbf{D})^{1/2}$ using a low-rank correction of $\mathbf{A}^{1/2}$. That is, to find a $\tilde{\Delta}$ of rank r such that $(\mathbf{A} + \mathbf{D})^{1/2} \approx \mathbf{A}^{1/2} + \tilde{\Delta}$. The rank r of $\tilde{\Delta}$ should be treated as a parameter, and should optimally be $O(k)$. We show in Section 3 that we can expect to find a good approximation with $r \ll n$.

We make two additional assumptions on $\mathbf{D}$. First, we assume that it is either positive semidefinite or negative semidefinite. Indefinite perturbations can be handled by splitting the update into two semidefinite perturbations and applying our algorithms sequentially. Second, we assume that $\mathbf{D}$ is given in a symmetric factorized form. We can combine the last two assumptions in a single assumption by assuming we are given a $\mathbf{Z} \in \mathbb{R}^{n\times k}$ such that $\mathbf{D} = \alpha\mathbf{Z}\mathbf{Z}^T$ where $\alpha = \pm 1$. We refer to the case of $\alpha = +1$ as *updating the square root*, and $\alpha = -1$ as *downdating the square root*.

In the case of updating the square root, we are guaranteed that $\mathbf{A} + \mathbf{Z}\mathbf{Z}^T$ is positive definite for any $\mathbf{Z}$, but this does not necessarily holds for downdating. Thus, for downdates we further assume that $\mathbf{A} - \mathbf{Z}\mathbf{Z}^T$ is positive definite. The following Lemma gives an easy way to test this condition in cases we also have access to the inverse of $\mathbf{A}^{1/2}$ or $\mathbf{A}$.

Lemma 1. *Suppose that $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric positive definite, and $\mathbf{Z} \in \mathbb{R}^{n\times k}$ for $k \le n$. Then $\mathbf{A} - \mathbf{Z}\mathbf{Z}^T$ is positive semidefinite if and only if $\mathbf{I}_k - \mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z}$ is positive semi definite.*

Proof. $\mathbf{A} - \mathbf{Z}\mathbf{Z}^T \succeq 0$ if and only if $\mathbf{A} \succeq \mathbf{Z}\mathbf{Z}^T$. Multiplying by $\mathbf{A}^{-1/2}$ on both sides, we see that this holds if and only if $\mathbf{A}^{-1/2}\mathbf{Z}\mathbf{Z}^T\mathbf{A}^{-1/2} \preceq \mathbf{I}_n$. The last inequality holds if and only if all the eigenvalues of $\mathbf{A}^{-1/2}\mathbf{Z}\mathbf{Z}^T\mathbf{A}^{-1/2}$ are smaller or equal to 1. However, the non-zero eigenvalues of that matrix are equal to the eigenvalues of $\mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z}$, so $\mathbf{A} - \mathbf{Z}\mathbf{Z}^T \succeq 0$ if and only if $\mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z} \preceq \mathbf{I}_k$, which is equivalent to $\mathbf{I}_k - \mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z}$ being positive definite. ∎

The correction $\tilde{\Delta}$ should be returned in factorized form. For reasons that will become apparent in our algorithm, the correction will be positive semidefinite when $\mathbf{D}$ is positive semidefinite, and negative semidefinite when $\mathbf{D}$ is negative semidefinite. Thus, we require our algorithm to return a $\mathbf{U} \in \mathbb{R}^{n\times r}$ such that $(\mathbf{A} + \alpha\mathbf{D})^{1/2} \approx \mathbf{A}^{1/2} + \alpha\mathbf{U}\mathbf{U}^T$.

The discussion so far was for updating or downdating the square root. We also aim at correcting the inverse of the square root, that is, $\mathbf{A}^{-1/2}$. Again, we will require $\mathbf{D}$ to be either positive semidefinite or negative semidefinite, and $\tilde{\Delta}$ to be factorized as well and definite. However, for updating the inverse of the square root, the definiteness of $\tilde{\Delta}$ will be opposite to the one of $\mathbf{D}$.

We can capture the distinction between updating the square root and the inverse square root with an additional parameter $\beta = \pm 1$, where we wish to update $\mathbf{A}^{\beta/2}$. Putting it all together, we arrive at the following problem:

Problem 1. Given implicit access to $\mathbf{A}^{1/2}$ and/or $\mathbf{A}^{-1/2}$, $\mathbf{Z} \in \mathbb{R}^{n\times k}$, $\alpha = \pm 1$, $\beta = \pm 1$ and target rank r, return a $\mathbf{U} \in \mathbb{R}^{n\times r}$ such that

$$(\mathbf{A} + \alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2} \approx \mathbf{A}^{\beta/2} + \alpha\beta\mathbf{U}\mathbf{U}^T.$$

3 | DECAY BOUNDS FOR SQUARE ROOTS CORRECTIONS

Given a matrix $\mathbf{A}$ and a low-rank perturbation $\mathbf{D}$, our goal is to find a low-rank correction $\tilde{\Delta}$ to $\mathbf{A}^{\beta/2}$. However, one can ask whether such a correction even exists? Let Δ denote the exact correction, that is,

$$\Delta := (\mathbf{A} + \mathbf{D})^{\beta/2} - \mathbf{A}^{\beta/2}.$$

In this section, we show that the eigenvalues of Δ exhibit a geometric decay. Thus, by applying eigenvalue thresholding to Δ we can obtain a low-rank approximate correction $\tilde{\Delta}$ (however, our algorithm uses a different method for finding $\tilde{\Delta}$).

3.1 | Decay bound for square root corrections

We first consider the case that $\beta = 1$, so we are perturbing the square root.

Let us denote $\mathbf{B} := \mathbf{A} + \mathbf{D}$. The following is a known identity:

$$\mathbf{A}^{1/2}\Delta + \Delta\mathbf{B}^{1/2} = \mathbf{B} - \mathbf{A}.$$

Indeed, since $\mathbf{B} = (\mathbf{B}^{1/2})^2 = (\mathbf{A}^{1/2} + \Delta)^2$ we have

$$\begin{aligned}\mathbf{B} - \mathbf{A} &= (\mathbf{A}^{1/2} + \Delta)^2 - \mathbf{A}\\ &= \mathbf{A} + \mathbf{A}^{1/2}\Delta + \Delta(\mathbf{A}^{1/2} + \Delta) - \mathbf{A}\\ &= \mathbf{A}^{1/2}\Delta + \Delta(\mathbf{A}^{1/2} + \Delta)\\ &= \mathbf{A}^{1/2}\Delta + \Delta\mathbf{B}^{1/2}.\end{aligned}$$

In our case, that is when $\mathbf{B} = \mathbf{A} + \mathbf{D}$, we see that Δ upholds the following Sylvester equation:

$$\mathbf{A}^{1/2}\Delta + \Delta\mathbf{B}^{1/2} = \mathbf{D}. \tag{2}$$

Since the rank of $\mathbf{D}$ is k, we see that Δ has a $(\mathbf{A}^{1/2}, -\mathbf{B}^{1/2})$-displacement rank of k. Beckermann and Townsend[13] recently developed singular value decay bounds for matrices with displacement structure. Using their results, we can prove the following bound.

Theorem 1. *Suppose that both* $\mathbf{A}$ *and* $\mathbf{B} = \mathbf{A} + \mathbf{D}$ *are symmetric positive definite matrices, and that* $\mathbf{D}$ *is of rank* k. *Let* $\Delta = \mathbf{B}^{1/2} - \mathbf{A}^{1/2}$. *Then for* $j \geq 1$, *the singular values of* Δ *satisfy the following bound*

$$\sigma_{j+kl}(\Delta) \leq 4\left[\exp\left(\frac{\pi^2}{2\log(4\hat{\kappa})}\right)\right]^{-2l}\sigma_j(\Delta),$$

where

$$\hat{\kappa} = \frac{2(\sqrt{\|\mathbf{A}\|_2 + \|\mathbf{D}\|_2} + \sqrt{\lambda_{\min}(\mathbf{A})}/2)}{\sqrt{\lambda_{\min}(\mathbf{A})}}.$$

Proof. Let $\mu = \sqrt{\lambda_{\min}(\mathbf{A})}$ and $\delta = \sqrt{\|\mathbf{A}\|_2 + \|\mathbf{D}\|_2}$. The matrix Δ upholds the following Sylvester equation

$$(\mathbf{A}^{1/2} - (\mu/2)\mathbf{I}_n)\Delta + \Delta(\mathbf{B}^{1/2} + (\mu/2)\mathbf{I}_n) = \mathbf{D}.$$

Thus, the results in Reference 13 show that we can bound

$$\sigma_{j+kl}(\Delta) \leq Z_l(E, F)\sigma_j(\Delta), \tag{3}$$

where E is any set that contains the spectrum of $\mathbf{A}^{1/2} - (\mu/2)\mathbf{I}_n$, F is any set that contains the spectrum of $-(\mathbf{B}^{1/2} + (\mu/2)\mathbf{I}_n)$, and $Z_l(E, F)$ is the Zolotarev number.

Let $a = \mu/2$. Since μ is the minimal eigenvalue of $\mathbf{A}^{1/2}$, all the eigenvalues of $\mathbf{A}^{1/2} - (\mu/2)\mathbf{I}_n$ are bigger than a or equal to it. Since $\mathbf{B}$ is by assumption positive definite, all the eigenvalues of $-(\mathbf{B}^{1/2} + (\mu/2)\mathbf{I}_n)$ are smaller than $-a$ or equal to it. Let $b = \delta + \mu/2$. Obviously, all the eigenvalues of $\mathbf{A}^{1/2} - (\mu/2)\mathbf{I}_n$ are smaller than b. Furthermore, since $\|\mathbf{B}\|_2 \leq \|\mathbf{A}\|_2 + \|\mathbf{D}\|_2$, all the eigenvalues of $-(\mathbf{B}^{1/2} + (\mu/2)\mathbf{I}_n)$ are bigger than or equal to $-b$. Thus, we can take $E = [a, b]$ and $F = [-b, -a]$. In Reference 13 it is also shown that

$$Z_l([a, b], [-b, -a]) \leq 4\left[\exp\left(\frac{\pi^2}{2\log(4b/a)}\right)\right]^{-2l},$$

plugging that into Equation (3) gives the desired bound. ∎

The theorem bounds the singular values. However, if $\mathbf{D}$ is positive semidefinite, then Δ is also positive semidefinite, and the bound is actually on the eigenvalues. Figure 1 illustrates the bound versus actual decay of the eigenvalues on two simple test cases. In both examples, $\mathbf{A} \in \mathbb{R}^{100\times100}$ is a diagonal matrix. In the left graph, the diagonal entries are sampled uniformly from $U(0, 1)$. In the right graph, diagonal entries are logarithmically spaced between 10^{-3} and 10^3. The perturbation is $\mathbf{D} = \mathbf{z}\mathbf{z}^T$, where $\mathbf{z}$ is a normalized Gaussian vector.

We remark the previous to the aforementioned theoretical results of Beckermann and Townsend,[13] it has been empirically observed that if the righthand side of a Sylvester equation is low rank, then the solution is well approximated using a low rank matrix.[14]

3.2 | Decay bound for inverse square root corrections

Observe that

$$-\mathbf{A}^{-1}\mathbf{D}\mathbf{B}^{-1} = \mathbf{B}^{-1} - \mathbf{A}^{-1} = \mathbf{A}^{-1/2}\Delta + \Delta\mathbf{B}^{-1/2}. \tag{4}$$

Since $\mathbf{D}$ is rank k, so the matrix $-\mathbf{A}^{-1}\mathbf{D}\mathbf{B}^{-1}$ is of rank at most k. Thus, similarly to Theorem 1, by observing that $\|\mathbf{B}^{-1}\|_2 \leq \|\mathbf{A}^{-1}\|_2(1 + \|\mathbf{D}\|_2\|\mathbf{B}^{-1}\|_2)$ (which follows from $\mathbf{B}^{-1} = \mathbf{A}^{-1} - \mathbf{B}^{-1}\mathbf{D}\mathbf{A}^{-1}$), we can prove the following bound on the singular

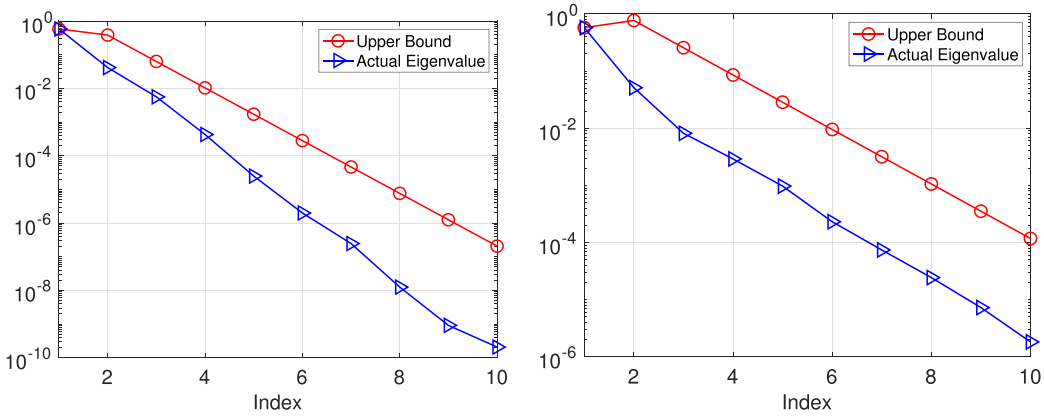

FIGURE 1 Illustration of the eigenvalue decay bound of Theorem 1 versus the actual decay observed on two simple examples—on the left we sampled A entries uniformly and on the right we used logspace sampling.

values of Δ:

$$\sigma_{j+kl}(\Delta) \leq 4\left[\exp\left(\frac{\pi^2}{2\log(4\hat{\kappa})}\right)\right]^{-2l}\sigma_j(\Delta),$$

where

$$\hat{\kappa} = \frac{2(\sqrt{\lambda_{\min}(\mathbf{A})^{-1}(1+\|\mathbf{D}\|_2\lambda_{\min}(\mathbf{B})^{-1})} + \sqrt{\lambda_{\max}(\mathbf{A})^{-1}}/2)}{\sqrt{\lambda_{\max}(\mathbf{A})^{-1}}}.$$

We omit the proof since it is almost identical to the proof of Theorem 1.

4 | EQUATION FOR SQUARE ROOTS CORRECTIONS AND ERROR ANALYSIS

Our algorithms are based on writing Δ as a solution of an equation, and then finding a low-rank approximate solution $\tilde{\Delta}$. Seemingly, Equations (2) and (4) are the equations we need. However, these equations contain the unknown $\mathbf{B}^{\beta/2}$ so they are not useful for us algorithmically. We derive a different equation instead. In particular, we write Δ as the solution of an algebraic Riccati equation, that is, in the form of Equation (1).

We can combine Equations (2) and (4) into a single equation:

$$\mathbf{A}^{\beta/2}\Delta + \Delta\mathbf{B}^{\beta/2} = \mathbf{B}^{\beta} - \mathbf{A}^{\beta}.$$

Recalling that $\mathbf{B}^{\beta/2} = \mathbf{A}^{\beta/2} + \Delta$, and plugging it into the last equation we get

$$\mathbf{A}^{\beta/2}\Delta + \Delta\mathbf{A}^{\beta/2} + \Delta^2 = \mathbf{B}^{\beta} - \mathbf{A}^{\beta}. \tag{5}$$

In order for the equation to fit Equation (1) we must write the right side as a positive semi-definite factorized matrix. The first step is finding a matrix $\mathbf{V} \in \mathbb{R}^{n\times k}$ such that

$$\mathbf{B}^{\beta} - \mathbf{A}^{\beta} = \alpha\beta\mathbf{V}\mathbf{V}^T.$$

When $\beta = 1$, and recalling that in Problem 1 we have $\mathbf{B} - \mathbf{A} = \alpha\mathbf{Z}\mathbf{Z}^T$, we can take $\mathbf{V} = \mathbf{Z}$. When $\beta = -1$, we obtain $\mathbf{V}$ using the Sherman–Morrison–Woodbury formula. Indeed,

$$\mathbf{B}^{-1} - \mathbf{A}^{-1} = -\alpha\mathbf{A}^{-1}\mathbf{Z}(\mathbf{I} + \alpha\mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{A}^{-1}.$$

So we take $\mathbf{V} = \mathbf{A}^{-1}\mathbf{Z}(\mathbf{I} + \alpha\mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z})^{-1/2}$ (note that if $\alpha = -1$ the condition that $\mathbf{A} - \mathbf{Z}\mathbf{Z}^T$ is positive definite ensures that $\mathbf{I} + \alpha\mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z}$ is positive definite and the inverse square root exists).

We now have the equation

$$\mathbf{A}^{\beta/2}\Delta + \Delta\mathbf{A}^{\beta/2} + \Delta^2 = \alpha\beta\mathbf{V}\mathbf{V}^T.$$

We do an additional change of variables to make the right-hand side positive definite even if $\alpha \neq \beta$. Let $\mathbf{C} = \alpha\beta\Delta$ (so $\mathbf{B}^{1/2} = \mathbf{A}^{1/2} + \alpha\beta\mathbf{C}$ since $\alpha = \pm 1$ and $\beta = \pm 1$). By multiplying the last equation on both sides by $\alpha\beta$ we obtain the equation:

$$\mathbf{A}^{\beta/2}\mathbf{C} + \mathbf{C}\mathbf{A}^{\beta/2} + \alpha\beta\mathbf{C}^2 = \mathbf{V}\mathbf{V}^T. \tag{6}$$

Except $\mathbf{C}$, all other quantities of the last equation are known, and solving Equation (6) using a low-rank positive semidefinite $\mathbf{C}$ of the form $\mathbf{C} = \mathbf{U}\mathbf{U}^T$ forms the basis of our algorithm (see next section). However, all our algorithms solve Equation (6) approximately, since they output low-rank solutions, while the exact solution tends to be full-rank.

We now analyze how errors in solving Equation (6) translate to errors in approximating $\mathbf{B}^{\beta/2}$. First, let us define the residual of an approximate solution:

$$\mathbf{R}(\tilde{\mathbf{C}}) := \mathbf{V}\mathbf{V}^T - \mathbf{A}^{\beta/2}\tilde{\mathbf{C}} - \tilde{\mathbf{C}}\mathbf{A}^{\beta/2} - \alpha\beta\tilde{\mathbf{C}}^2.$$

We start with a backward error bound, that is, showing that if the residual has a small norm, then $\mathbf{A}^{\beta/2} + \alpha\beta\tilde{\mathbf{C}}$ is the square root of a matrix that is close to $(\mathbf{A} + \alpha\mathbf{Z}\mathbf{Z}^T)^\beta$.

Lemma 2. *We have*

$$\|(\mathbf{A} + \alpha\mathbf{Z}\mathbf{Z}^T)^\beta - (\mathbf{A}^{\beta/2} + \alpha\beta\tilde{\mathbf{C}})^2\|_F = \|\mathbf{R}(\tilde{\mathbf{C}})\|_F.$$

Proof. We have defined $\mathbf{V}$ so that $(\mathbf{A} + \alpha\mathbf{Z}\mathbf{Z}^T)^\beta = \mathbf{A}^\beta + \alpha\beta\mathbf{V}\mathbf{V}^T$, so

$$\begin{aligned}
\|(\mathbf{A} + \alpha\mathbf{Z}\mathbf{Z}^T)^\beta - (\mathbf{A}^{\beta/2} + \alpha\beta\tilde{\mathbf{C}})^2\|_F &= \|\mathbf{A}^\beta + \alpha\beta\mathbf{V}\mathbf{V}^T - (\mathbf{A}^{\beta/2} + \alpha\beta\tilde{\mathbf{C}})^2\|_F \\
&= \|\mathbf{A}^\beta + \alpha\beta\mathbf{V}\mathbf{V}^T - \mathbf{A}^\beta - \alpha\beta\mathbf{A}^{\beta/2}\tilde{\mathbf{C}} - \alpha\beta\tilde{\mathbf{C}}\mathbf{A}^{\beta/2} - \tilde{\mathbf{C}}^2\|_F \\
&= \|\alpha\beta\mathbf{R}(\tilde{\mathbf{C}})\|_F \\
&= \|\mathbf{R}(\tilde{\mathbf{C}})\|_F.
\end{aligned}$$

■

In order to get a bound on the forward error in terms of the backward error, we need the following perturbation bound for the matrix square root:

Lemma 3 (Schmitt[15(Lemma 2.2)]). *Suppose that* $\mathbf{Re}(\mathbf{A}_j) \succeq \mu_j^2\mathbf{I}$, $\mu_j > 0$, $j = 1, 2$. *Then, both* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *have square roots satisfying* $\mathbf{Re}\left(\mathbf{A}_j^{1/2}\right) \succeq \mu_j\mathbf{I}$ *for j = 1,2, and*

$$\|\mathbf{A}_2^{1/2} - \mathbf{A}_1^{1/2}\|_2 \leq \frac{1}{\mu_1 + \mu_2}\|\mathbf{A}_2 - \mathbf{A}_1\|_2.$$

Lemma 4. *Let* $\mathbf{W}$ *and* $\mathbf{H}$ *be two symmetric positive definite matrices. The following bounds hold*:

$$\|\mathbf{W}^{\beta/2} - \mathbf{H}\|_F \leq (n^{1/2}\|\mathbf{W}^\beta - \mathbf{H}^2\|_F)^{1/2},$$

$$\|\mathbf{W}^{\beta/2} - \mathbf{H}\|_2 \leq \min\left(\frac{\|\mathbf{W}^\beta - \mathbf{H}^2\|_2}{\sqrt{\lambda_{\min}(\mathbf{W}^\beta)}}, (n^{1/2}\|\mathbf{W}^\beta - \mathbf{H}^2\|_F)^{1/2}\right).$$

Proof. The bound $\|\mathbf{W}^{\beta/2} - \mathbf{H}\|_2 \leq \|\mathbf{W}^\beta - \mathbf{H}^2\|_2/\sqrt{\lambda_{\min}(\mathbf{W}^\beta)}$ follows immediately from Lemma 3. The Frobenius norm bound follows from Wihler inequality the pth root of positive semidefinite matrices: for any two $n \times n$ positive semidefinite matrix $\mathbf{X}$ and $\mathbf{Y}$ and $p > 1$ we have $\|\mathbf{X}^{1/p} - \mathbf{Y}^{1/p}\|_F^p \leq n^{(p-1)/2}\|\mathbf{X} - \mathbf{Y}\|_F$.[16] We apply this inequality to $\mathbf{X} = \mathbf{W}^\beta$ and $\mathbf{Y} = \mathbf{H}^2$. ■

Finally, we obtain the following bound:

Corollary 1. *Suppose that both* $\mathbf{A}$ *and* $\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T$ *are positive definite where* $\alpha=\pm1$. *Let* $\tilde{\mathbf{C}}$ *be a positive semidefinite matrix. Assume that* $\mathbf{A}^{\beta/2}+\alpha\beta\tilde{\mathbf{C}}$ *is positive definite. The following bounds holds*:

$$\|(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}-(\mathbf{A}^{\beta/2}+\alpha\beta\tilde{\mathbf{C}})\|_F \le \left(n^{1/2}\|\mathbf{R}(\tilde{\mathbf{C}})\|_F\right)^{1/2},$$

$$\|(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}-(\mathbf{A}^{\beta/2}+\alpha\beta\tilde{\mathbf{C}})\|_2 \le \min\left\{\frac{\|\mathbf{R}(\tilde{\mathbf{C}})\|_F}{\sqrt{\lambda_{\min}((\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta})}}, \left(n^{1/2}\|\mathbf{R}(\tilde{\mathbf{C}})\|_F\right)^{1/2}\right\}.$$

Proposition 1. *Suppose that both* $\mathbf{A}$ *and* $\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T$ *are positive definite where* $\alpha=\pm1, \beta=\pm1$. *The matrix* $\alpha\beta((\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}-\mathbf{A}^{\beta/2})$ *is a solution to Equation (6). Conversely, if* $\mathbf{C}$ *is a positive definite solution of Equation (6) for which* $\mathbf{A}^{\beta/2}+\alpha\beta\mathbf{C}$ *is positive definite as well, then* $\mathbf{A}^{\beta/2}+\alpha\beta\mathbf{C}=(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}$.

Proof. Let $\mathbf{C}=\alpha\beta((\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}-\mathbf{A}^{\beta/2})$. By substation we have $\mathbf{A}^{\beta/2}\mathbf{C}+\mathbf{C}\mathbf{A}^{\beta/2}+\alpha\beta\mathbf{C}^2=\alpha\beta(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta}-\alpha\beta\mathbf{A}^{\beta}$. Recall that we defined $\mathbf{V}$ such that $(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta}=\mathbf{A}^{\beta}+\alpha\beta\mathbf{V}\mathbf{V}^T$ so we find that $\mathbf{A}^{\beta/2}\mathbf{C}+\mathbf{C}\mathbf{A}^{\beta/2}+\alpha\beta\mathbf{C}^2=\mathbf{V}\mathbf{V}^T$ and Equation (6) holds.

Conversely, if $\mathbf{C}$ is a positive definite solution of Equation (6) then $\mathbf{R}(\mathbf{C})=0$. Since $\mathbf{A}^{\beta/2}+\alpha\beta\mathbf{C}$ is positive definite, Corollary 1 ensures that $\|(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}-(\mathbf{A}^{\beta/2}+\alpha\beta\tilde{\mathbf{C}})\|_F\le 0$. This can only happen if $(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}-(\mathbf{A}^{\beta/2}+\alpha\beta\tilde{\mathbf{C}})=0$, that is, $\mathbf{A}^{\beta/2}+\alpha\beta\mathbf{C}=(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^T)^{\beta/2}$. ∎

5 | ALGORITHMS

In this section, we describe our algorithms for solving Problem 1. As alluded earlier, our algorithms are based on using a Riccati low-rank solver as encapsulated by RicattiLRSolver. However, in some combinations of α and β there is a challenge: while the returned $\tilde{\mathbf{C}}$ is guaranteed to be positive definite, there is no guarantee that $\mathbf{A}^{\beta/2}+\alpha\beta\tilde{\mathbf{C}}$ is positive definite as well. Such guarantee is necessary in order to apply Corollary 1. Thus, we split our algorithm to various cases based on the combination of α and β. Our proposed algorithms are summarized in pseudo-code form in Algorithm 1.

Algorithm 1. Algorithms for updating/downdating square root and inverse square root

Inputs: $\mathbf{A}^{1/2}\in\mathbb{R}^{n\times n}$ and/or $\mathbf{A}^{-1/2}\in\mathbb{R}^{n\times n}$ implicitly, $\alpha=\pm1$, $\beta=\pm1$, $\mathbf{Z}\in\mathbb{R}^{n\times k}$, target rank r.
Output: $\mathbf{U}\in\mathbb{R}^{n\times r}$ such that
$(\mathbf{A}+\alpha\mathbf{Z}\mathbf{Z}^{\mathrm{T}})^{\beta/2}\approx\mathbf{A}^{\beta/2}+\alpha\beta\mathbf{U}\mathbf{U}^{\mathrm{T}}$.

$\underline{\alpha=+1,\beta=+1}$: (only $\mathbf{A}^{1/2}$ required)
$\mathbf{U}\leftarrow$ RiccatiLRSolver$(\mathbf{A}^{1/2},\mathbf{Z}^{\mathrm{T}},+1,r)$

$\underline{\alpha=-1,\beta=-1}$: (only $\mathbf{A}^{-1/2}$ required)
$\mathbf{G}\leftarrow\mathbf{A}^{-1/2}\mathbf{Z}$
Verify $\mathbf{I}_k-\mathbf{G}^{\mathrm{T}}\mathbf{G}$ is positive definite (o/w return error)
$\mathbf{V}\leftarrow\mathbf{A}^{-1/2}\mathbf{G}(\mathbf{I}_k-\mathbf{G}^{\mathrm{T}}\mathbf{G})^{-1/2}$
$\mathbf{U}\leftarrow$ RiccatiLRSolver$(\mathbf{A}^{-1/2},\mathbf{V}^{\mathrm{T}},+1,r)$

$\underline{\alpha=-1,\beta=+1}$:
Execute the $\alpha=-1,\beta=-1$ case to obtain $\mathbf{U}_1$.
$\mathbf{U}\leftarrow\mathbf{A}^{1/2}\mathbf{U}_1(\mathbf{I}_r+\mathbf{U}_1^{\mathrm{T}}\mathbf{A}^{1/2}\mathbf{U}_1)^{-1/2}$

$\underline{\alpha=+1,\beta=-1}$:
Execute the $\alpha=+1,\beta=+1$ case to obtain $\mathbf{U}_1$.
$\mathbf{U}\leftarrow\mathbf{A}^{-1/2}\mathbf{U}_1(\mathbf{I}_r+\mathbf{U}_1^{\mathrm{T}}\mathbf{A}^{-1/2}\mathbf{U}_1)^{-1/2}$

5.1 | Updating ($\alpha = 1$) the square root ($\beta = 1$)

This is the simplest case: we simply call $\mathbf{U} \leftarrow \text{RiccatiLRSolver}(\mathbf{A}^{1/2}, \mathbf{Z}^T, +1, r)$ and return $\mathbf{U}$.

5.2 | Downdating ($\alpha = -1$) the inverse square root ($\beta = -1$)

We first compute $\mathbf{V} = \mathbf{A}^{-1}\mathbf{Z}(\mathbf{I}_k - \mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z})^{-1/2}$. Along the way we can verify that $\mathbf{I} - \mathbf{Z}^T\mathbf{A}^{-1}\mathbf{Z}$ is positive definite, which is required for $\mathbf{A} - \mathbf{Z}\mathbf{Z}^T$ to be positive semidefinite, and our algorithm to work. We now call $\mathbf{U} \leftarrow \text{RiccatiLRSolver}(\mathbf{A}^{-1/2}, \mathbf{V}^T, +1, r)$ and return $\mathbf{U}$.

5.3 | Downdating ($\alpha = -1$) the square root ($\beta = 1$)

Seemingly, we could simply call $\mathbf{U} \leftarrow \text{RiccatiLRSolver}(\mathbf{A}^{1/2}, \mathbf{Z}^T, -1, r)$ and return $\mathbf{U}$. However, there is no guarantee that $\mathbf{A}^{1/2} - \mathbf{U}\mathbf{U}^T$ is positive definite, and Corollary 1 no longer guarantees that we have an approximation to the principal square root.

If we want to approximate the principal square root, we can first solve for downdating the inverse square root ($\alpha = -1, \beta = -1$), obtaining $\mathbf{U}_1$ such that $(\mathbf{A} - \mathbf{Z}\mathbf{Z}^T)^{-1/2} \approx \mathbf{A}^{-1/2} + \mathbf{U}_1\mathbf{U}_1^T$. We now use the Sherman–Morrison–Woodbury formula to note that

$$(\mathbf{A}^{-1/2} + \mathbf{U}_1\mathbf{U}_1^T)^{-1} = \mathbf{A}^{1/2} - \mathbf{A}^{1/2}\mathbf{U}_1(\mathbf{I}_r + \mathbf{U}_1^T\mathbf{A}^{1/2}\mathbf{U}_1)^{-1}\mathbf{U}_1^T\mathbf{A}^{1/2},$$

so we return $\mathbf{U} = \mathbf{A}^{1/2}\mathbf{U}_1(\mathbf{I}_r + \mathbf{U}_1^T\mathbf{A}^{1/2}\mathbf{U}_1)^{-1/2}$.

5.4 | Updating ($\alpha = 1$) the inverse square root ($\beta = -1$)

Again, calling the Riccati solver directly might return a corrected matrix which it not necessarily positive definite, and it will not be a good approximation to the principal square root. To approximate the principal square root, we first solve the updating problem for the square root $(\mathbf{A} + \mathbf{Z}\mathbf{Z}^T)^{1/2} \approx \mathbf{A}^{1/2} + \mathbf{U}_1\mathbf{U}_1^T$ ($\alpha = +1, \beta = +1$), and then use the Sherman–Morrison–Woodbury formula to find a $\mathbf{U}$ such that $(\mathbf{A}^{1/2} + \mathbf{U}_1\mathbf{U}_1^T)^{-1} = \mathbf{A}^{-1/2} - \mathbf{U}\mathbf{U}^T$. We omit the details and simply refer the reader to the pseudo code description in Algorithm 1.

5.5 | Costs

The main cost of the algorithms is in solving the Riccati equation. Even when the Sherman–Morrison–Woodbury formula is needed to ensure positive definiteness of the correction, its cost of $O(nk^2)$ is subsumed by the cost of solving the Riccati equation. Overall, under our assumptions on the cost of solving the Riccati equation, the overall cost of the algorithms is $O((T_{\mathbf{A}^{1/2}} + T_{\mathbf{A}^{-1/2}})r^2 + nr^4)$ where $T_{\mathbf{A}^{1/2}}$ and $T_{\mathbf{A}^{-1/2}}$ are the costs of the taking products of $\mathbf{A}^{1/2}$ and $\mathbf{A}^{-1/2}$ (respectively) with a vector. In many of the applications we discuss in the next section $\mathbf{A}$ is diagonal, in which case the cost of the algorithms reduces to $O(nr^4)$.

6 | APPLICATIONS

6.1 | ZCA whitening of high dimensional data

Whitening transformations are designed to transform a random vector (or samples of that random vector) with a known covariance matrix into a new random vector whose covariance is the identity matrix. Suppose that $\mathbf{x} \in \mathbb{R}^p$ is a random vector with covariance matrix $\Sigma \in \mathbb{R}^{p\times p}$. Let $\mathbf{W}$ be any matrix such that $\mathbf{W}^T\mathbf{W} = \Sigma^{-1}$; such a matrix is called a *whitening matrix*. Then the covariance matrix of the random vector $\mathbf{z} = \mathbf{W}\mathbf{x}$ is the identity, so the random variable has been *whitened*.

There are several possible choices for $\mathbf{W}$, leading to different whitening transformations. ZCA whitening is the whitening transformation defined by $\mathbf{W} = \Sigma^{-1/2}$.

In practice, the ZCA transformation is learned from data. Given samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ an estimate $\hat{\Sigma}$ of Σ is formed, and $\hat{\mathbf{W}} = \hat{\Sigma}^{-1/2}$ is used for the ZCA whitening matrix. A common choice is to use the sample covariance matrix $\mathbf{S}_n = n^{-1}\mathbf{X}_c^T\mathbf{X}_c$ for $\hat{\Sigma}$ where $\mathbf{X}_c$ is the data matrix whose rows are $\mathbf{x}_1, \ldots, \mathbf{x}_n$ after centering (subtraction of the mean).

However, it is well appreciated in the statistical literature that when the random vectors are high dimensional, that is, when p is of the same order as n (or much larger), then the sample covariance $\mathbf{S}_n$ is a poor estimate of Σ. Indeed, one can easily see that if $p > n$ then $\mathbf{S}_n$ is not even invertible so the ZCA transformation is not even defined. In high dimensional settings it is common to adopt the *spiked covariance model* of Johnstone.[17] In the spiked covariance model it is assumed that the covariance matrix has the following form:

$$\Sigma = \sigma^2\mathbf{I}_p + \mathbf{Z}\mathbf{Z}^T, \tag{7}$$

for some $\mathbf{Z} \in \mathbb{R}^{p\times k}$ (k is a parameter).

Suppose that we have formed an estimate $\hat{\Sigma}$ of Σ with the same structure as in Equation (7), and we want to transform the samples using ZCA whitening. Explicitly computing the square root of $\hat{\Sigma}$ requires $O(p^3)$, which is prohibitive when p is large, and additional $O(p^2n)$ is required for applying the transformation to the data. Instead, we can use our algorithm to find a $\mathbf{U} \in \mathbb{R}^{n\times r}$ with $r = O(k)$ such that

$$\hat{\Sigma}^{-1/2} \approx \sigma^{-1}\mathbf{I} - \mathbf{U}\mathbf{U}^T.$$

We can then apply the ZCA transformation in $O(npk)$.

6.2 | Updating/downdating polar decomposition and ZCA transformed data

Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ where $n \geq d$, a polar decomposition of it is

$$\mathbf{X} = \mathbf{U}\mathbf{P},$$

where $\mathbf{U}$ has orthonormal columns and $\mathbf{P}$ is symmetric positive semidefinite. The matrix $\mathbf{P}$ is always unique, and is given by $\mathbf{P} = (\mathbf{X}^T\mathbf{X})^{1/2}$. If $\mathbf{X}$ has full rank, then $\mathbf{P}$ is positive definite, and $\mathbf{U} = \mathbf{X}\mathbf{P}^{-1}$. The polar decomposition can be computed using a reduced SVD, so the cost of computing a polar decomposition is $O(nd^2)$. There are quite a few uses for the polar decomposition.[18]

We now consider the following updating/downdating problem. Let us denote the rows of $\mathbf{X}$ by $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, that is, row j of $\mathbf{X}$ is $\mathbf{x}_j^T$. Suppose we already have a polar decomposition $\mathbf{X} = \mathbf{U}\mathbf{P}$ of $\mathbf{X}$. The *downdating problem* is: compute the polar decomposition of a matrix $\mathbf{X}_-$ obtained by removing one row from $\mathbf{X}$. The*updating problem* is: compute the polar decomposition of $\mathbf{X}_+$, a matrix obtained by adding a single row to $\mathbf{X}$.

We describe an algorithm for downdating a polar decomposition. The updating algorithm is almost the same. Without loss of generality, assume we remove the last row from $\mathbf{X}$:

$$\mathbf{X}_-^T\mathbf{X}_- = \mathbf{X}^T\mathbf{X} - \mathbf{x}_n\mathbf{x}_n^T.$$

This is a rank-1 perturbation of $\mathbf{X}^T\mathbf{X}$. The $\mathbf{P}$-factor of $\mathbf{X}_-$, which we denote by $\mathbf{P}_-$, is obtained by computing the square root of $\mathbf{X}_-^T\mathbf{X}_-$, and we already have the square root $\mathbf{P}$ for the unperturbed matrix $\mathbf{X}^T\mathbf{X}$. So we can use the algorithm described in Section 5 to find a matrix $\mathbf{U} \in \mathbb{R}^{d\times k}$ for some small r (a parameter; e.g., $k = 4$) such that

$$\mathbf{P}_- \approx \tilde{\mathbf{P}}_- := \mathbf{P} - \mathbf{U}\mathbf{U}^T.$$

Assume now that $\mathbf{X}_-$ is full rank as well. We now have

$$\mathbf{U}_- \approx \tilde{\mathbf{U}}_- := \mathbf{X}_-\tilde{\mathbf{P}}_-^{-1}.$$

Using the Sherman–Morrison–Woodbury formula:

$$\tilde{\mathbf{P}}_{-}^{-1} = \mathbf{P}^{-1} + \mathbf{P}^{-1}\mathbf{U}(\mathbf{I}_r - \mathbf{U}^T\mathbf{P}^{-1}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{P}^{-1},$$

so

$$\tilde{\mathbf{U}}_{-} = \mathbf{X}_{-}\mathbf{P}^{-1}(\mathbf{I}_r + \mathbf{U}(\mathbf{I}_k - \mathbf{U}^T\mathbf{P}^{-1}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{P}^{-1}).$$

Now notice that $\mathbf{X}_{-}\mathbf{P}^{-1}$ is just the first $n-1$ rows of $\mathbf{X}\mathbf{P}^{-1} = \mathbf{U}$ so we do not need to recompute it. Multiplying $\mathbf{X}_{-}\mathbf{P}^{-1}$ by $(\mathbf{I}_d + \mathbf{U}(\mathbf{I}_r - \mathbf{U}^T\mathbf{P}^{-1}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{P}^{-1})$ can be done, utilizing the low rank structure of that matrix, using $O(ndr)$ operations. If $r \ll n$ this is a big reduction in complexity over $O(nd^2)$.

The polar decomposition is closely connected to ZCA whitening. Suppose that $\mathbf{X}$ is a data matrix whose rows are sampled from a zero mean random vector (or, alternatively, $\mathbf{X}$ has been centered). The $\mathbf{U}$-factor is equal, up to scaling, to the ZCA transformed data, while the inverse of the $\mathbf{P}$-factor is, up to scaling, the ZCA whitening matrix itself. Using the ability to update the inverse square root, the procedure for updating/downdating the polar decomposition can be adjusted to update/downdate ZCA. Updating/downdating ZCA can be useful if you want to transform data that arrives over time while the covariance matrix itself changes slowly. That is, data point $\mathbf{x}_j$ is sampled with covariance Σ_j. If we assume the covariance changes slowly, we can keep an approximate ZCA of the data by taking the sample covariance over a sliding window. To do so efficiently, we can use the proposed ZCA updating/downdating procedure to first remove outdated data (a downdate operation), and then add the newly arrived data (and update operation).

6.3 | Sampling from a multivariate normal distribution with perturbed precision matrix

Consider a random vector $\mathbf{x} \in \mathbb{R}^n$ following a multivariate normal distribution with precision matrix $\mathbf{Q}$, that is, $\mathbf{x} \sim N(\mu, \mathbf{Q}^{-1})$. Suppose we want to sample $\mathbf{x}$. This can be accomplished by sampling a vector $\mathbf{z}$ from the standard multivariate normal distribution (i.e., $\mathbf{z} \sim N(0, \mathbf{I}_n)$) and then computing the sample $\mathbf{x} = \mu + \mathbf{Q}^{-1/2}\mathbf{z}$.

In certain cases the matrix $\mathbf{Q}$ has the structure of a low-rank perturbation of a fixed precision matrix, that is, $\mathbf{Q} = \mathbf{Q}_0 + \mathbf{Z}\mathbf{Z}^T$. Assuming we already have computed the inverse square root of $\mathbf{Q}_0$, we can use our algorithms to compute a $\mathbf{U}$ such that $\mathbf{Q}^{-1/2} \approx \mathbf{Q}_0^{-1/2} - \mathbf{U}\mathbf{U}^T$. We can then sample efficiently from $\mathbf{x}$.

Such cases can occur in Gibbs Samplers for Bayesian inference on spatially structured data. An example is the image reconstruction task discussed in Reference 19. The computational bottleneck in the algorithm proposed in Reference 19 is sampling from a conditional Gaussian distribution whose precision matrix has the structure $\mathbf{Q} = \gamma_{\text{prior}}\mathbf{L} + \gamma_{\text{obs}}\mathbf{Z}\mathbf{Z}^T$ where $\mathbf{L}$ is a fixed discrete Laplace operator that encodes prior smoothness assumptions on the image, while $\mathbf{Z}^T$ encodes how the high-resolution images are blurred and downsampled to yield low-resolution images.

6.4 | Preconditioned second-order optimization

Recently introduced by Gupta et al.,[20] SHAMPOO is a preconditioned second-order optimization method for solving problems in which the parameter space is naturally organized as a $m \times n$ matrix or higher order tensor. Here, we consider the matrix-shaped case. In this case, at the core, SHAMPOO performs update steps of the form

$$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\mathbf{L}_t^{-1/4}\mathbf{G}_t\mathbf{R}_t^{-1/4}, \tag{8}$$

where η is the learning rate, $\{\mathbf{W}_t\}$ are the parameters at time t, $\{\mathbf{G}_t\}$ are the gradients at time t, and

$$\mathbf{L}_t := \epsilon\mathbf{I}_m + \sum_{s=1}^{t}\mathbf{G}_s\mathbf{G}_s^T \quad \mathbf{R}_t := \epsilon\mathbf{I}_n + \sum_{s=1}^{t}\mathbf{G}_s^T\mathbf{G}_s.$$

If $m \gg nt$ then it is better to avoid holding $\mathbf{L}_t^{-1/4}$ explicitly (which is $m \times m$), and simply hold an implicit representation of both $\mathbf{L}_t^{-1/4}$ and $\mathbf{L}_t^{-1/2}$ as diagonal plus low-rank matrices. In each iteration we can update both by applying our algorithm

twice. Since $n \ll m$, we can store $\mathbf{R}_t$ explicitly, and compute $\mathbf{R}_t^{-1/4}$ in each iteration. If $n \gg mt$ we can reverse roles, implicitly keeping $\mathbf{R}_t$ and explicitly keeping $\mathbf{L}_t$. Even if both m and n are of comparable size, then in some cases $\mathbf{G}_t$ is of low-rank, and again we can track $\mathbf{L}_t$ and $\mathbf{R}_t$ using our algorithm.

We stress that our method has an advantage over Fasi et al.,[9] when applied to SHAMPOO. Fasi et al.,[9] can only handle perturbations of the identity, so when applied to compute the square root of $\mathbf{L}_{t+1}$ it cannot use the square root of $\mathbf{L}_t$ (which is available from the previous iteration). Our algorithm, on the other hand, can use the fact that $\mathbf{L}_{t+1} = \mathbf{L}_t + \mathbf{G}_t\mathbf{G}_t^T$ for a low rank update of the previous iteration. If the matrices are explicitly held, Fasi et al.,[9] will have a cost per iteration that grows linearly with iteration count, while with our algorithm the cost will stay constant.

6.5 | Faster generalized least squares with a spiked weight matrix

Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$. In generalized least squares we wish to find the minimizer

$$\min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{X}\mathbf{w} - \mathbf{b}\|_{\mathbf{W}}, \tag{9}$$

where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is some symmetric positive definite weight matrix, and $\|\mathbf{z}\|_{\mathbf{W}} := \sqrt{\mathbf{z}^T\mathbf{W}\mathbf{z}}$. In this section, we focus on the cases that $\mathbf{W}$ can be written as a diagonal plus a definite low rank perturbation $\mathbf{W} = \mathbf{D} + \alpha\mathbf{Z}\mathbf{Z}^T$ where $\mathbf{D} \in \mathbb{R}^{n\times n}$ is diagonal, and $\mathbf{Z} \in \mathbb{R}^{n\times k}$.

A statistical motivation for this problem is generalized linear regression with a spiked covariance matrix. Assume that the rows of $\mathbf{X}$ correspond to data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, the entries $y_1, \ldots, y_n$ of $\mathbf{y}$ are responses. We now assume that the responses follow the model $y_i = \mathbf{x}_i^T\mathbf{w}^\star + \epsilon_i$ where the vector of noise elements $\epsilon_1, \ldots, \epsilon_m$ is distributed according to $\mathcal{N}(0, \mathbf{C})$ for some covariance matrix $\mathbf{C} = \mathbf{D} + \alpha\mathbf{Z}\mathbf{Z}^T$. The optimal unbiased estimate of $\mathbf{w}^\star$ is obtained by solving Equation (9) with $\mathbf{W} = \mathbf{C}^{-1}$.

One can easily see that Equation (9) is equivalent to

$$\min_{\mathbf{w}\in\mathbb{R}^d} \|\mathbf{W}^{1/2}\mathbf{X}\mathbf{w} - \mathbf{W}^{1/2}\mathbf{y}\|_2. \tag{10}$$

Once we have efficiently computed $\mathbf{W}^{1/2}\mathbf{X}$ and $\mathbf{W}^{1/2}\mathbf{y}$, we can leverage faster, sketching based, least squares algorithms.[21,22] Using the algorithms from Section 5 we can compute a $\mathbf{U} \in \mathbb{R}^{n\times r}$ such that $\mathbf{W}^{1/2} = \mathbf{D}^{-1/2} - \alpha\mathbf{U}\mathbf{U}^T$ with $r = O(k)$. We can them compute $\mathbf{W}^{1/2}\mathbf{X}$ and $\mathbf{W}^{1/2}\mathbf{y}$ efficiently.

7 | EXPERIMENTS

We report experiments exploring the ability of our algorithm to find low-rank corrections to the square root or inverse square root of a perturbed matrix. In our experiments, we focus on the quality of the corrections found. We do not report running time since the code we used for the algebraic Riccati solver (downloaded from the homepage of Mishra and Vandereycken[12]) is not optimized to take advantage of the structures present in the input matrices for the specific algebraic Riccati equations our algorithm solves.

7.1 | Synthetic experiments

We first test our algorithm on randomly generated matrices, and compare them to the approximations obtained using the algorithm of Beckermann et al.,[7] the approximation obtained using the algorithm of Beckermann et al.,[8] and the optimal correction obtained by zeroing out the smallest eigenvalues of the exact correction. The matrix $\mathbf{A} \in \mathbb{R}^{100\times 100}$ is a diagonal matrix, whose diagonal is either sampled uniformly from $U(0, 1)$, or whose entries are logarithmically spaced between 10^{-3} and 10^3. The perturbation is $\mathbf{D} = \mathbf{z}\mathbf{z}^T$, where $\mathbf{z}$ is a normalized Gaussian vector. We consider both updates ($\alpha = +1$) and downdates ($\alpha = -1$). In case of downdates, we multiply $\mathbf{z}$ by 0.1 to ensure positive definiteness after the downdate. We consider both the square root ($\beta = +1$) and inverse square root ($\beta = -1$). We plot the relative error $\|(\mathbf{A} + \alpha\mathbf{z}\mathbf{z}^T)^{\beta/2} - (\mathbf{A}^{\beta/2} + \alpha\beta\mathbf{U}\mathbf{U}^T)\|_F / \|(\mathbf{A} + \alpha\mathbf{z}\mathbf{z}^T)^{\beta/2}\|_F$ as a function of the rank of the update (#columns in $\mathbf{U}$).

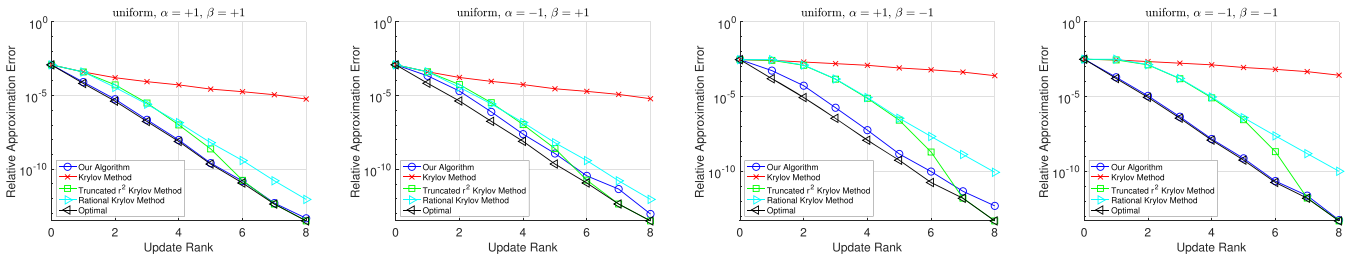

FIGURE 2 Comparison of our algorithm to approximations obtained using Beckermann et al.[7] and to the optimal approximation, on randomly generated matrices of the form $\mathbf{D} + \mathbf{z}\mathbf{z}^T$ where $\mathbf{D}$ is diagonal with uniformly sampled entries from $N(0, 1)$.

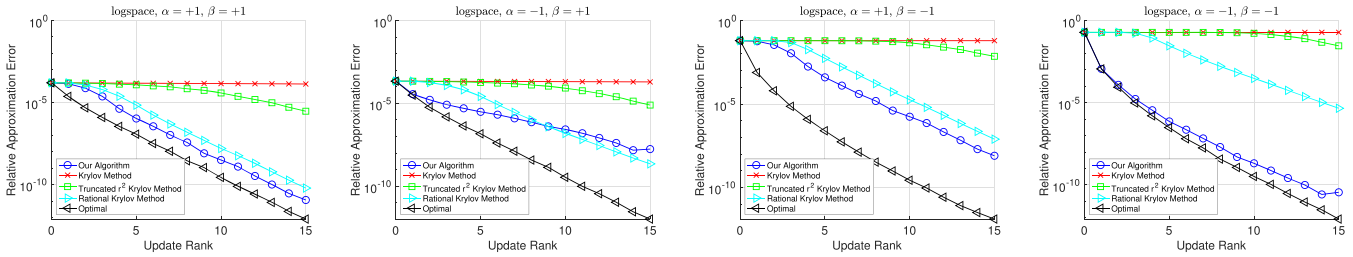

FIGURE 3 Comparison of our algorithm to approximations obtained using Beckermann et al.[7] and to the optimal approximation, on randomly generated matrices of the form $\mathbf{D} + \mathbf{z}\mathbf{z}^T$ where $\mathbf{D}$ is diagonal with logarithmically spaced entries.

We consider the algorithm of Beckermann et al.[7] applied in two different ways. The first, which is labeled in the graphs as "Krylov Method," simply runs the algorithm of algorithm of Beckermann et al.[7] for r iterations (r is the target rank), to obtain a rank r perturbation. In the second, which is labeled in the graphs as "Truncated r^2 Krylov Method," runs the algorithm of algorithm of Beckermann et al.[7] for r^2 iterations, but then computes the best rank r approximation to the correction (which is of rank r^2). This algorithm has the same asymptotic cost for diagonal matrices as our algorithm.

We implemented the algorithm of Beckermann et al.[8] using the Rational Krylov Toolbox for MATLAB.[2] The poles are obtained using that toolbox as well. The algorithm is labeled in the graphs as "Rational Krylov Method."

Figures 2 and 3 show the result for all different combinations. Our algorithm is clearly able to find much better approximations than the Krylov method of Beckermann et al.[7]

7.2 | Matrices arising from second-order optimization

Our next set of experiments simulates the use of our algorithm to track $\mathbf{L}_t^{-1/4}$ or $\mathbf{R}_t^{-1/4}$ in SHAMPOO (see Section 6.4). We obtain and preprocess the data in a similar way to Reference 9, but the experiment itself is different. We downloaded two test matrices available from the LINGVO framework for TENSORFLOW[23] and are available on GitHub.[3] These matrices are obtained by accumulating updates with $\alpha = 0$. Unfortunately, the provided test matrices are only the final accumulated matrix, and do not contain the discrete updates themselves, so we need to extract updates that accumulate to the final matrix. We do so in a similar fashion to the one used by Fasi et al.[9]: we compute an eigendecomposition, and keep dominant factors that are bigger than 0.1. This yields a rank 82 approximation to the first matrix, and a rank 221 approximation to the second matrix.

For the experiment, we split the low rank approximation into discrete updates of rank 5. So we now have a sequence of $\{\mathbf{G}_s\}$, each $\mathbf{G}_s$ having five columns. Our goal is to efficiently track $\mathbf{L}_t^{-1/4}$ where $\mathbf{L}_t = \alpha\mathbf{I} + \sum_{s=1}^{t}\mathbf{G}_s\mathbf{G}_s^T = \mathbf{L}_{t-1} + \mathbf{G}_t\mathbf{G}_t^T$ where we set $\alpha = 0.001$. We use our algorithm to form two sets of updates, $\{\mathbf{U}_s\}$ and $\{\mathbf{W}_s\}$, each with five columns, such that $\mathbf{L}_t^{-1/2} \approx \mathbf{L}_{t-1}^{-1/2} + \mathbf{U}_t\mathbf{U}_t^T \approx \alpha^{-1/2}\mathbf{I} + \sum_{s=1}^{t}\mathbf{U}_s\mathbf{U}_s^T$ and $\mathbf{L}_t^{-1/4} \approx \mathbf{L}_{t-1}^{-1/4} + \mathbf{W}_t\mathbf{W}_t^T \approx \alpha^{-1/4}\mathbf{I} + \sum_{s=1}^{t}\mathbf{W}_s\mathbf{W}_s^T$. Note that in iteration t, we consider the latest perturbation to be of $\mathbf{L}_{t-1}$ and the approximation of $\mathbf{L}_{t-1}^{-1/2}$. This saves time (since the perturbation rank does not grow) and storage (we do not need to keep previous updates). We plot in Figure 4 the distance between

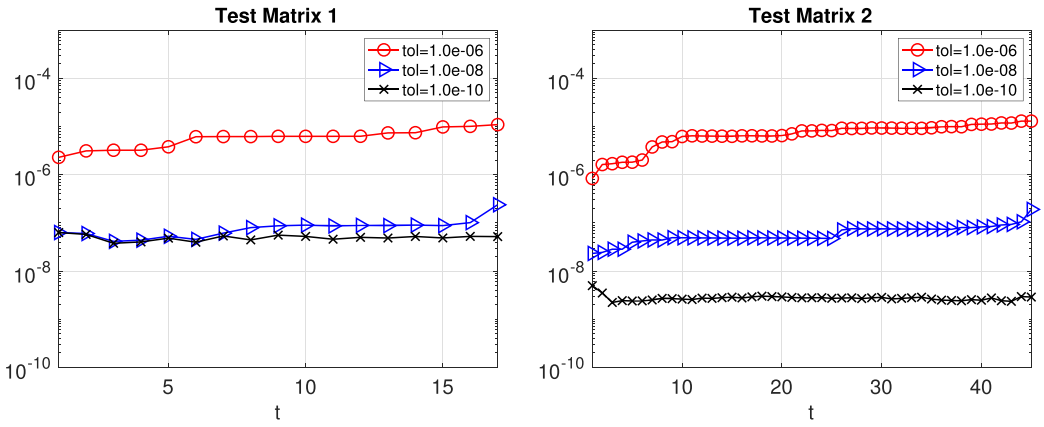

FIGURE 4 Simulating the use of our algorithm to track $\mathbf{L}_t^{-1/4}$ or $\mathbf{R}_t^{-1/4}$ in SHAMPOO, on two test matrices from the LINGVO framework.

$\mathbf{L}_t^{-1/4}$ and its approximation, as it evolves over time. We do so for three different tolerances in the internal Riccati low-rank solver. We see that our algorithm is able to track $\mathbf{L}_t^{-1/4}$ well over time, without the errors blowing-up.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their helpful comments. Haim Avron and Shany Shmueli were partially supported by the Israel Science Foundation (Grant no. 1272/17) and by the US-Israel Binational Science Foundation (Grant no. 2017698). Petros Drineas was partially supported by NSF 10001415 and NSF 10001390.

CONFLICT OF INTEREST STATEMENT

This study does not have any conflicts to disclose.

DATA AVAILABILITY STATEMENT

Research data are not shared.

ENDNOTES

[1]https://www.mpi-magdeburg.mpg.de/projects/mess
[2]http://guettel.com/rktoolbox/guide/html/index.html
[3]https://github.com/tensorflow/lingvo/tree/master/lingvo/core/testdata

REFERENCES

1. Higham NJ. Functions of matrices. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2008.
2. Aune E, Eidsvik J, Pokern Y. Iterative numerical methods for sampling from high dimensional Gaussian distributions. Stat Comput. 2013;23(4):501–21.
3. Chow E, Saad Y. Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions. SIAM J Sci Comput. 2014;36(2):A588–A608.
4. Frommer A, Güttel S, Schweitzer M. Efficient and stable Arnoldi restarts for matrix functions based on quadrature. SIAM J Matrix Anal Appl. 2014;35(2):661–83.
5. Pleiss G, Jankowiak M, Eriksson D, Damle A, Gardner J. Fast matrix square roots with applications to Gaussian processes and Bayesian optimization. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in neural information processing systems. Volume 33. Red Hook, NY: Curran Associates, Inc.; 2020. p. 22268–81.
6. Bernstein DS, Van Loan CF. Rational matrix functions and rank-1 updates. SIAM J Matrix Anal Appl. 2000;22(1):145–54.
7. Beckermann B, Kressner D, Schweitzer M. Low-rank updates of matrix functions. SIAM J Matrix Anal Appl. 2018;39(1):539–65.
8. Beckermann B, Cortinovis A, Kressner D, Schweitzer M. Low-rank updates of matrix functions II: rational Krylov methods. SIAM J Numer Anal. 2021;59(3):1325–47.
9. Fasi M, Higham NJ, Liu X. Computing the square root of a low-rank perturbation of the scaled identity matrix. MIMS ePrint. 2022. p. 1.
10. Benner P, Saak J. Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey. GAMM Mitt. 2013;36(1):32–52.
11. Bini DA, Iannazzo B, Meini B. Numerical solution of algebraic Riccati equations. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2011.
12. Mishra B, Vandereycken B. A Riemannian approach to low-rank algebraic Riccati equations. arXiv preprint arXiv: 1312.4883, 2014.

13. Beckermann B, Townsend A. On the singular values of matrices with displacement structure. SIAM J Matrix Anal Appl. 2017;38(4):1227–48.
14. Benner P, Kürschner P. Computing real low-rank solutions of Sylvester equations by the factored ADI method. Comput Math Appl. 2014;67(9):1656–72.
15. Schmitt BA. Perturbation bounds for matrix square roots and Pythagorean sums. Linear Algebra Appl. 1992;174:215–27.
16. Wihler T. On the Hölder continuity of matrix functions for normal matrices. J Inequalities Pure Appl Math. 2009;10:10.
17. Johnstone IM. On the distribution of the largest eigenvalue in principal components analysis. Ann Stat. 2001;29(2):295–327.
18. Higham NJ. Computing the polar decomposition, with applications. SIAM J Sci Stat Comput. 1986;7(4):1160–74.
19. Bardsley JM. MCMC-based image reconstruction with uncertainty quantification. SIAM J Sci Comput. 2012;34(3):A1316–32.
20. Gupta V, Koren T, Singer Y. Shampoo: preconditioned stochastic tensor optimization. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. Volume 80. New York: PMLR; 2018. p. 1842–50.
21. Woodruff DP. Sketching as a tool for numerical linear algebra. Found Trends Theor Comput Sci. 2014;10(1–2):1–157.
22. Drineas P, Mahoney MW. RandNLA: randomized numerical linear algebra. Commun ACM. 2016;59(6):80–90.
23. Shen J, Nguyen P, Wu Y, Chen Z, Chen MX, Jia Y, et al. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. arXiv preprint arXiv: abs/1902.08295, 2019.

How to cite this article: Shmueli S, Drineas P, Avron H. Low-rank updates of matrix square roots. Numer Linear Algebra Appl. 2023;e2528. https://doi.org/10.1002/nla.2528

APPENDIX A. SOLVING EQUATION (1) FOR $\alpha = -1$

If $\alpha = +1$, Equation (1) is an instance of the *algebraic Riccati equation*, for which Mishra and Vandereycken[12] proposed an algorithm for finding an approximate low rank solution. That algorithm can be adjusted to the $\alpha = -1$, by making several small adjustments to the various *Euclidean* components (the Riemannian ones are obtained by converting the Euclidean components to Riemannian ones). We frame the expressions with an $\alpha = \pm 1$ to cover both cases concurrently.

- Optimization problem: the new optimization problem is:

$$\min_{rank\mathbf{X}=k,\Delta\succeq 0} 0.25\|\mathbf{EX}+\mathbf{XE}+\alpha\mathbf{XBB}^T\mathbf{X}^T-\mathbf{G}^T\mathbf{G}\|_F^2. \tag{A1}$$

- Gradient expression: the method in Reference 12 keeps $\mathbf{X}$ in factorized low-rank form $\mathbf{X}=\mathbf{YY}^T$. Let

$$\mathbf{S}(\mathbf{Y}) := \mathbf{EYY}^T+\mathbf{YY}^T\mathbf{E}+\alpha\mathbf{YY}^T\mathbf{BB}^T\mathbf{YY}^T-\mathbf{G}^T\mathbf{G}.$$

  The cost function in Equation (A1) is $F(\mathbf{Y}) := 0.25\|\mathbf{S}(\mathbf{Y})\|_F^2$. Simple calculations show that the gradient of $\mathbf{F}$ is:

$$\nabla F(\mathbf{Y}) = \mathbf{E}^T\mathbf{S}(\mathbf{Y})\mathbf{Y}+\mathbf{S}(\mathbf{Y})\mathbf{EY}+\alpha\mathbf{S}(\mathbf{Y})\mathbf{Y}^T\mathbf{BB}^T\mathbf{Y}+\alpha\mathbf{BB}^T\mathbf{YY}^T\mathbf{S}(\mathbf{Y}).$$

- Hessian expression: using a similar technique as Reference 12, we calculate directional derivative in direction $\mathbf{w}$ by computing $\lim \frac{1}{\epsilon}(\nabla F(\mathbf{Y}+\epsilon\mathbf{W})-\nabla F(\mathbf{Y}))$. In the limit, $\mathbf{S}(\mathbf{Y})$ and $\mathbf{S}(\mathbf{Y}+\epsilon\mathbf{W})$ include α and so are slightly different from the ones used in Reference 12. Nevertheless, the expression for the Euclidean Hessian is:

$$\begin{aligned}&\mathbf{E}^T\mathbf{S}(\mathbf{Y})\mathbf{Y}+\mathbf{E}^T\mathbf{S}(\mathbf{Y})\mathbf{W}+\mathbf{S}(\mathbf{Y})\mathbf{EW}+\mathbf{S}(\mathbf{Y})\mathbf{E}^T\mathbf{Y}\\&\quad+\alpha\big(\mathbf{S}(\mathbf{Y})\mathbf{YY}^T\mathbf{BB}^T\mathbf{Y}+\mathbf{S}(\mathbf{Y})\mathbf{W}^T\mathbf{BB}^T\mathbf{Y}+\mathbf{S}(\mathbf{Y})\mathbf{WY}^T\mathbf{BB}^T\mathbf{Y}+\mathbf{S}(\mathbf{Y})\mathbf{YY}^T\mathbf{BB}^T\mathbf{W}\big)\\&\quad+\alpha\big(\mathbf{BB}^T\mathbf{YY}^T\mathbf{S}(\mathbf{Y})\mathbf{W}+\mathbf{BB}^T\mathbf{YY}^T\mathbf{S}(\mathbf{Y})\mathbf{Y}+\mathbf{BB}^T\mathbf{YW}^T\mathbf{S}(\mathbf{Y})\mathbf{Y}+\mathbf{BB}^T\mathbf{WY}^T\mathbf{S}(\mathbf{Y})\mathbf{Y}\big).\end{aligned}$$