

**A Riemannian optimization approach
for computing low-rank solutions of
Lyapunov equations**

*Bart Vandereycken
Stefan Vandewalle*

Report TW 544, July 2009



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations

Bart Vandereycken
Stefan Vandewalle

Report TW 544, July 2009

Department of Computer Science, K.U.Leuven

Abstract

We propose a new framework based on optimization on manifolds to approximate the solution of a Lyapunov matrix equation by a low-rank matrix. The method minimizes the error on the Riemannian manifold of symmetric positive semi-definite matrices of fixed rank. We detail how objects from differential geometry, like the Riemannian gradient and Hessian, can be efficiently computed for this manifold. As minimization algorithm we use the Riemannian Trust-Region method of [Found. Comput. Math., 7 (2007), pp. 303–330] based on a second-order model of the objective function on the manifold. Together with an efficient preconditioner this method can find low-rank solutions with very little memory. We illustrate our results with numerical examples.

Keywords : Lyapunov matrix equations, low-rank approximations, numerical optimization on manifolds, Riemannian manifold, large-scale equations
MSC : 65F10, 65N22, 93B40, 65K05, 90C26, 58C05

A RIEMANNIAN OPTIMIZATION APPROACH FOR COMPUTING LOW-RANK SOLUTIONS OF LYAPUNOV EQUATIONS

BART VANDEREYCKEN[†] AND STEFAN VANDEWALLE[†]

Abstract. We propose a new framework based on optimization on manifolds to approximate the solution of a Lyapunov matrix equation by a low-rank matrix. The method minimizes the error on the Riemannian manifold of symmetric positive semi-definite matrices of fixed rank. We detail how objects from differential geometry, like the Riemannian gradient and Hessian, can be efficiently computed for this manifold. As minimization algorithm we use the Riemannian Trust-Region method of [Found. Comput. Math., 7 (2007), pp. 303–330] based on a second-order model of the objective function on the manifold. Together with an efficient preconditioner this method can find low-rank solutions with very little memory. We illustrate our results with numerical examples.

Key words. Lyapunov matrix equations, low-rank approximations, numerical optimization on manifolds, Riemannian manifold, large-scale equations

AMS subject classifications. 65F10, 65N22, 93B40, 65K05, 90C26, 58C05

1. Introduction. The subject of this paper is the problem of approximating solutions of large-scale matrix equations by iterative methods. We will focus on the generalized Lyapunov equation

$$AXM + MXA = C \tag{1.1}$$

with given symmetric matrices $A, M, C \in \mathbb{R}^{n \times n}$. Furthermore, we assume that A and M are positive definite and that the generalized eigenvalues of the pencil $A - \lambda M$ are strictly positive. Equation (1.1) is a linear matrix equation in $X \in \mathbb{R}^{n \times n}$ and, by our assumptions, its solution X is symmetric and unique, see [13] or [35].

Lyapunov equations are of significant importance in control theory [7], model reduction [33] and stochastic analysis of dynamical systems [39]; see [4] for a general overview. A recurring pattern is that the solution X of the Lyapunov equation (1.1) can be associated with a Gramian of the linear time-invariant system

$$M\dot{x}(t) = Ax(t) + Bu, \quad x(0) = x_0$$

with state x and input u . These Gramians capture useful information about the energy of a system such as the controllability, observability and covariance matrices. In general, this matrix X is dense even if the system is modelled by sparse matrices.

Large-scale matrix equations arise naturally when the system is modelled by a system of partial differential equations (PDE). Semi-discretization in space by, e.g., the finite element method (FEM) results in a Lyapunov equation with a sparse system matrix A and a sparse mass matrix M . The dimension n of these matrices is usually very large. A major challenge when solving matrix equations for such a large-scale problem is that the dense matrix X has n^2 entries. Storing this matrix when $n \gg 1000$ will be problematic, let alone solving the Lyapunov equation itself.

1.1. Low-rank approximations for large-scale problems. Direct methods for solving a Lyapunov equation, such as the Bartels–Stewart algorithm [6], compute the $n \times n$ matrix X in $O(n^3)$ floating point operations. This makes them suitable

[†]Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium (bart.vandereycken@cs.kuleuven.be, stefan.vandewalle@cs.kuleuven.be).

only for small-scale problems up to $n \simeq 1000$. Even optimal solvers, like multigrid, will be out of reach, since their $O(n^2)$ complexity is still too large in practice. The origin of this problem lies in the fact that these matrix equations are defined on the tensor product space $\mathbb{R}^n \times \mathbb{R}^n$, whereas the physical system, modelled by A and M , is defined on \mathbb{R}^n . This is termed the curse of dimensionality and it requires special consideration regarding discretizations and solvers.

A popular algebraic technique for alleviating this is approximating the solution by a low-rank matrix of rank $k \ll n$. In this way the number of unknowns is reduced to $O(kn)$. If one can compute this low-rank approximation in $O(k^c n)$ with c small, then solving such a large-scale matrix becomes feasible. There already exist a significant number of low-rank Lyapunov solvers using this principle. Some are based on either the ADI and Smith method, see [36], [31] and [21]; on Krylov subspace techniques, see [38], [25], [26] and [40]; or on low-rank arithmetic, see [20]. Our method is based on Riemannian optimization.

Clearly, low-rank approximations are not always suitable. Although it is reasonable to expect that the quality of the approximation will improve if the rank grows, this rank can be very large, maybe too large to be of any practical use. Indeed, consider the equation $AX + XA = 2A$ with solution $X = I$, the identity matrix. Any low-rank approximation will unavoidably be very poor. However, there are a significant number of applications where the solution does exhibit the so called *low-rank property*: the eigenvalues of the matrix X have an exponential decay and the accuracy of the best low-rank approximation increases rapidly with growing rank. This has been studied in [37], [41], [5] and [19] for the case of $M = I$ and a low-rank matrix C in (1.1). There, bounds have been proposed that depend on the spectrum of A and, to some extent, explain the low-rank phenomenon. If we consider, e.g., a matrix A with condition number κ , these bounds suggest that we can approximate X with a relative accuracy of ϵ using a rank $k = O(\kappa \log(1/\epsilon))$, see [19, Remark 1].

The low-rank solvers cited above (except for [20]) are all based on a clever reformulation of a well-known iterative method, e.g. the ADI method, to a low-rank setting. In each step i , the algorithm can be reformulated exactly to work on a factor Y_i of the iterate $X_i = Y_i Y_i^\top$ and with each iteration, the rank of the approximation X_i will grow. If convergence is fast, the approximation will have low rank. In this case, these solvers can be very efficient, since they are very cheap per iteration. If convergence is slow however, there is not much that can be done, except to keep on iterating and possibly truncating iterates along the way, see [21]. A major problem is that there seems to be little room to incorporate preconditioning compatible with the structure of a Lyapunov equation, although the method in [40] can be viewed as preconditioning with A^{-1} . Another problem is that these algorithms need to form the product of A with a square root of C . For general matrices, this square root can be costly to compute. Therefore, the algorithms are usually only applied to the case where $C = bb^\top$ as this gives a trivial square root b .

The solver [20] is based on low-rank arithmetic and circumvents the problem of slow convergence by doing a geometric multigrid iteration with low-rank matrices. In each iteration, the rank of the iterate will temporarily grow, only to be truncated back to low rank. It has the benefit that it can combine an optimal and fast solver with a low-rank solution, provided that this low-rank arithmetic does not destroy convergence. However, this can only be seen a posteriori and the algorithm seems rather sensitive to the rank chosen at each level of the multigrid algorithm. Furthermore, geometric multigrid is usually used with complicated smoothers and acting as

a preconditioner, or exchanged in favor of algebraic multigrid. It is far from clear how to perform this in low-rank arithmetic.

1.2. Contributions and outline. The major contribution of this paper is the introduction of a new geometric framework for computing low-rank approximations to solutions of matrix equations. The method is based on optimizing an objective function on the Riemannian manifold of symmetric positive semi-definite matrices of fixed rank. We focus on the stable generalized Lyapunov equation and show that this Riemannian optimization approach can lead to an efficient and scalable solver. The results on the geometric properties of the manifold are kept general in the expectation that the geometric framework can be applied also to other matrix equations, e.g. Riccati, and to similar matrix manifolds, e.g. non-symmetric fixed rank.

We begin in §3 by discussing the objective function in our approach, and by showing that it represents the energy norm of the error made by the low-rank approximation. After a brief introduction to Riemannian optimization in §4, we derive some properties of the manifold essential for the Riemannian Trust-Region (RTR) method in §5. These include the tangent space, the Riemannian gradient and Hessian, and the retraction. Special attention is given to the efficient implementation and application of these geometric objects.

In §6 we will show how the previous results can be applied to the Lyapunov equation. First, a second-order model is derived, after which we discuss the efficient implementation of the RTR method. Due to the large-scale nature of the problem, we devote §7 to the problem of finding a suitable preconditioner. Again, most attention is given to the efficient computation of this preconditioner. Finally, in §8 we present some numerical results and compare our approach with other low-rank Lyapunov solvers.

2. Notational conventions and basic principles of the method.

2.1. Notations. We will now summarize the notational conventions that will be used in the remainder of this text. The space of $n \times k$ real matrices is denoted by $\mathbb{R}^{n \times k}$. The symbol $\mathbb{R}_*^{n \times k}$ is used to denote the restriction to full rank matrices. With $Y_\perp \in \mathbb{R}^{n \times n-k}$ we mean the normalized orthogonal complement of $Y \in \mathbb{R}^{n \times k}$, $k < n$ such that $Y_\perp^\top Y = 0$ and $Y_\perp^\top Y_\perp = I_k$. With $A \succ 0$ we denote that A is symmetric positive definite (s.p.d.) and likewise, $A \succeq 0$ means that A is symmetric positive semi-definite (s.p.s.d.). The set of eigenvalues of a matrix A is denoted by $\lambda(A)$. Similarly, $\lambda(A, M)$ denotes the set of generalized eigenvalues of the pencil $A - \lambda M$.

The directional derivative of a function f at x in the direction of ξ is denoted by $Df(x)[\xi]$. Derivatives w.r.t. the parametrization parameter t of a curve $\gamma(t)$ are denoted by dots, so $\dot{\gamma}(t) := d\gamma/dt$ and $\ddot{\gamma}(t) := d^2\gamma/dt^2$. We do not assume anything specific about the parametrization except for sufficient differentiability.

We use the following spaces for $n \times n$ matrices: the orthogonal group O_n , the symmetric matrices S_n^{sym} and the skew-symmetric matrices S_n^{skew} . In order to differentiate between abstract elements and concrete matrices, we use the following notation: x, y, z are abstract elements of the manifold \mathcal{M} , while big capitals X, Y, Z are matrices, as stored on a computer. The statement $x = YY^\top$ means that the low-rank s.p.s.d. matrix x is implemented as a factorization with matrix Y . The inner product at an element x of a Riemannian manifold is denoted by $\langle \xi, \nu \rangle_x$, with ξ and ν tangent vectors of x . With $\text{tr}(\cdot)$ we mean the trace of a matrix.

2.2. The new method. The method we propose finds a low-rank approximation of X by minimizing the objective function

$$f : \mathcal{M} \rightarrow \mathbb{R}, \quad X \mapsto \text{tr}(XAXM) - \text{tr}(XC) \quad (2.1)$$

over the manifold \mathcal{M} of s.p.s.d. matrices of rank k ,

$$\mathcal{M} = \{X : X \in \mathbb{S}_n^{\text{sym}}, X \succeq 0, \text{rank}(X) = k\}. \quad (2.2)$$

This results in the optimization problem

$$\min_x f(x) \quad \text{subject to } x \in \mathcal{M}. \quad (2.3)$$

We solve (2.3) by a robust second-order method in the framework of Riemannian optimization, namely the Riemannian Trust-Region method (RTR) from [1]. This optimization algorithm exploits the fact that \mathcal{M} is a smooth manifold. We will show how geometric objects from Riemannian geometry can be efficiently computed for \mathcal{M} and that f is related to a norm of the error made by the low-rank approximation. As a consequence, the minimizer of (2.3) will be locally the best low-rank solution in this norm.

Our method is inspired by the existing low-rank solvers and it addresses some of their drawbacks. In particular, we solve for the best low-rank solution in an energy norm, similar to Krylov subspace methods. Yet, in our approach the rank can be kept fixed and we can find a better approximation without the need to increase the rank. This resembles a kind of low-rank arithmetic but performed in the precise setting of differential geometry. Furthermore, the optimization method is robust regarding saddle points and local maxima and it enables us to use preconditioning. In addition, the algorithm is quite general and can be adapted to other matrix equations and other data-sparse techniques, as long as they fit into the Riemannian optimization framework. We remark that in contrast to most low-rank solvers, the actual performance of the algorithm is less dependent on the approximation quality of the low-rank matrix. Whether or not the low-rank approximation is good enough, will depend on the application.

3. The objective function. Matrix equation (1.1) can be written as a normal linear equation by means of vectorization. Let $\text{vec}(\cdot)$ denote the operator that makes a vector from a matrix by column-wise stacking and let \otimes denote the Kronecker product, then we have that (1.1) is equivalent to

$$\mathcal{L} \text{vec}(X) = \text{vec}(C), \quad \text{with } \mathcal{L} = A \otimes M + M \otimes A. \quad (3.1)$$

This is easy to see once one observes that $\text{vec}(ABC) = (C^\top \otimes A) \text{vec}(B)$ for square matrices A, B, C . Note that the $\text{vec}(\cdot)$ operator defines an isomorphism between the Hilbert spaces $\mathbb{R}^{n \times n}$ and \mathbb{R}^{n^2} where the inner products relate to each other by

$$\langle X, Y \rangle_{\mathbb{R}^{n \times n}} = \langle \text{vec}(X), \text{vec}(Y) \rangle_{\mathbb{R}^{n^2}} \iff \text{tr}(X^\top Y) = \text{vec}(X)^\top \text{vec}(Y).$$

See [29] and [24] for more on these properties. Further on, we will need the \mathcal{L} -norm

$$\|\cdot\|_{\mathcal{L}} = \sqrt{\langle \cdot, \cdot \rangle_{\mathcal{L}}}, \quad \text{with } \langle u, v \rangle_{\mathcal{L}} = \langle u, \mathcal{L}v \rangle.$$

In order for this to make sense, \mathcal{L} as defined in (3.1), should be symmetric and positive definite. We show that this is always the case by our assumptions on A and M , as detailed in the introduction.

PROPOSITION 3.1. *Suppose $A, M \succ 0$ and $\lambda(A, M) > 0$, then $\mathcal{L} \succ 0$.*

Proof. From the symmetry of A and M , we have that $\mathcal{L} = \mathcal{L}^\top$. To prove that \mathcal{L} is positive, we need to show that $\lambda(A \otimes M + M \otimes A) > 0$. Since $M \succ 0$, there exists a full rank matrix G such that $M = GG^\top$. By Sylvester's Law of Inertia [24], the congruence relation with the full rank matrix $(G \otimes G)^{-1} = G^{-1} \otimes G^{-1}$ does not change the number of positive eigenvalues. We get

$$\begin{aligned} \lambda(\mathcal{L}) > 0 &\iff \lambda((G \otimes G)^{-1}(A \otimes GG^\top + GG^\top \otimes A)(G \otimes G)^{-\top}) > 0, \\ &\iff \lambda(G^{-1}AG^{-\top} \otimes I + I \otimes G^{-1}AG^{-\top}) > 0, \\ &\iff \lambda(G^{-1}AG^{-\top}) > 0 \iff \lambda(A, M) > 0. \end{aligned}$$

The second last equivalence follows from the property that $\lambda(B \otimes I + I \otimes B)$ consists of the set of all numbers $\mu_i + \mu_j$, where μ_i and μ_j are eigenvalues of B , see [28]. \square

Now we can work out the \mathcal{L} -norm of the error $E = X - X_*$ of $X \in \mathcal{M}$, with X_* the true solution of (1.1). Since X is symmetric, E is symmetric too and using the relations from above, we get

$$\begin{aligned} \|\text{vec}(E)\|_{\mathcal{L}}^2 &= \text{vec}(E)^\top (A \otimes M + M \otimes A) \text{vec}(E), \\ &= \text{vec}(E)^\top \text{vec}(MEA) + \text{vec}(E)^\top \text{vec}(AEM), \\ &= 2 \text{tr}(EMEA), \end{aligned}$$

where we used the identity $\text{tr}(FG) = \text{tr}(GF)$. Substituting E by its definition, we continue

$$\begin{aligned} \|\text{vec}(E)\|_{\mathcal{L}}^2 &= 2 \text{tr}[(X - X_*)M(X - X_*)A], \\ &= 2 \text{tr}(XMXA) - 2 \text{tr}(XMX_*A + XAX_*M) + 2 \text{tr}(X_*MX_*A), \\ &= 2 \text{tr}(XMXA) - 2 \text{tr}(XC) + 2 \text{tr}(X_*MX_*A), \\ &= 2f(X) + 2 \text{tr}(X_*MX_*A). \end{aligned}$$

Since $\text{tr}(X_*MX_*A)$ is a constant, minimizing $f(X)$, as defined in (3.1), amounts to minimizing the \mathcal{L} -norm of the error of X .

The optimization problem (2.3) is formulated on the set of s.p.s.d. matrices of rank k . One may wonder whether this is a restriction in comparison to optimizing over the set of s.p.s.d. matrices with rank less than or equal to k , i.e.,

$$\min_x f(x) \quad \text{s.t. } x \in \{X : X \in S_n^{\text{sym}}, X \succeq 0, \text{rank}(X) \leq k\}. \quad (3.2)$$

Intuitively, we can expect that at least one of the minimizers of problem (3.2) will not be of rank lower than k if the rank of the exact solution X_* is at least k . This can be proved rigorously as follows.

PROPOSITION 3.2. *Let $A, M \succ 0$, $\lambda(A, M) > 0$ and $\text{rank}(X_*) \geq k$, with X_* the exact solution of (1.1), then the minimizer of (3.2) has rank k .*

Proof. The proof mimics the proof of a similar result in [23, Prop. 2.4], but is based using the \mathcal{L} -norm instead of the Euclidean norm. Let $g(X) := \frac{1}{2} \|\text{vec}(X - X_*)\|_{\mathcal{L}}^2 = \text{tr}[(X - X_*)M(X - X_*)A]$. Since $g(X) = f(X) + c$, $c \in \mathbb{R}$, replacing f with g in (3.2) does not change the minimizers. Suppose \hat{X} is such a minimizer and $\text{rank}(\hat{X}) = l < k$. Then the rank one perturbation $\hat{X} + \epsilon bb^\top$, with $\epsilon \geq 0$ and arbitrary $b \in \mathbb{R}^{n \times 1}$ satisfying $\text{tr}(bb^\top Mbb^\top A) = 1$ will not have a function value lower than $g(\hat{X})$. This gives

$$g(\hat{X} + \epsilon bb^\top) = g(\hat{X}) - 2\epsilon \text{tr}[(X_* - \hat{X})Abb^\top M] + \epsilon^2 \geq g(\hat{X}).$$

So for all ϵ and b we have that $2 \operatorname{tr}[(X_* - \hat{X})Abb^\top M] \leq \epsilon$ and this means that $\operatorname{tr}[(X_* - \hat{X})Abb^\top M] = 0$. Since $\mathcal{L} \succ 0$ and b arbitrary, we can conclude $X_* = \hat{X}$. This contradicts the assumption that $\operatorname{rank}(\hat{X}) = l < k$. \square

Hence it suffices to restrict the optimization to \mathcal{M} .

4. Riemannian optimization. Having defined the objective function, we need to decide on the numerical method to solve problem (2.3). It may be tempting to use standard techniques for constrained optimization to solve (2.3), but the presence of the low-rank constraint makes this very difficult. The condition that x is an s.p.s.d. matrix of fixed rank cannot easily be formulated as a set of (in)equalities suitable for off the shelf constrained optimization codes. Furthermore, any algorithm that generates infeasible points that are not of low rank requires n^2 variables. Since the manifold is introduced to reduce the dimension, it is clear that using arbitrary points in $\mathbb{R}^{n \times n}$ will never yield the desired $O(nk^c)$ complexity.

We address this problem by exploiting the fact that the set \mathcal{M} is a *smooth manifold*, which enables us to use the framework of *Riemannian optimization*, see e.g., [22], [15] and [3]. Riemannian optimizers abandon the concept of a flat, Euclidean space $\mathbb{R}^{n \times n}$ and formulate the problem on a curved, Riemannian manifold \mathcal{M} instead. In exchange, we can eliminate the low-rank constraint and get an unconstrained optimization problem that, by construction, will only use feasible points. Although optimizing on a smooth manifold is more complicated than optimizing on $\mathbb{R}^{n \times n}$, there are general techniques available; see [3] for an overview in case of matrix manifolds.

In this paper, we will use the Riemannian Trust-Region (RTR) method of [1] which is a matrix-free and robust second-order method suitable for large-scale optimization. Simply put, the method is a generalization of the classic unconstrained Trust-Region (TR) method to Riemannian manifolds. Each iteration consists of two phases: first, approximating the solution of the so-called trust-region subproblem, followed by the computation of a new iterate. The algorithm is summarized below:

- 1: **for** $k = 1, 2, \dots$ **do**
- 2: Approximately minimize the trust-region subproblem

$$\min_{\xi \in T_x \mathcal{M}, \langle \xi, \xi \rangle_x \leq \Delta^2} m_x(\xi) := f(x) + \langle \operatorname{grad} f(x), \xi \rangle_x + \frac{1}{2} \langle \operatorname{Hess} f(x)[\xi], \xi \rangle_x \quad (4.1)$$

- 3: Construct the new iterate $x \leftarrow R_x(\xi)$ and update the trust-region radius Δ depending on the quality of the new iterate x .
- 4: **end for**

The algorithm computes a series of approximations $x \in \mathcal{M}$ by using a series of second-order models $m_x : T_x \mathcal{M} \rightarrow \mathbb{R}$ associated with every x . These models are each defined on the tangent space of x , denoted by $T_x \mathcal{M}$, and are based on the Riemannian gradient and Hessian. The optimized tangent vector ξ that solves (4.1), is then used to get a new iterate. To do this we need a mapping $R_x : T_x \mathcal{M} \rightarrow \mathcal{M}$, called the retraction, that maps tangent vectors $\xi \in T_x \mathcal{M}$ to the manifold.

Under quite general conditions on the manifold and cost function, the RTR method has the same favorable convergence properties as the classic TR method, see [1]. The main differences with a classic TR approach are the successive lifting-and-retracting to and from the tangent space by means of the retraction operator R_x , and the use of concepts from Riemannian geometry to define m_x .

REMARK 4.1. A different approach is to parametrize $x = YY^\top$ with $Y \in \mathbb{R}_*^{n \times k}$ and minimize $f(YY^\top)$ over $\mathbb{R}_*^{n \times k}$. This has been used successfully in the context of

low-rank SDP solvers, [11]. While this effectively lowers the dimension of the search space, it suffers from non-local minimizers which can cause problems for second-order methods like Newton's method. Indeed, every $Z = YQ$, with $Q \in O_k$ is also a minimizer. A possible solution to obtain isolated minimizers, see [2] and [27], is to optimize over the quotient space $\mathbb{R}_*^{n \times k}/O_k$. This approach is in fact closely related to optimizing on \mathcal{M} since these two manifolds are diffeomorphic.

5. The manifold. This section is devoted to the Riemannian geometry of \mathcal{M} . Manifold \mathcal{M} has already been studied in [22, Ch. 5] and [23], where the fact that \mathcal{M} is a smooth manifold is proved.

THEOREM 5.1 ([22, Ch. 5],[23, Prop. 2.1]). *The set \mathcal{M} is a smooth embedded submanifold of $\mathbb{R}^{n \times n}$ with dimension $nk - \frac{1}{2}k(k-1)$. The tangent space of \mathcal{M} at an element x is*

$$T_x \mathcal{M} = \{\Delta x + x \Delta^\top : \Delta \in \mathbb{R}^{n \times n}\}. \quad (5.1)$$

In order to be able to use this manifold in our Riemannian optimization algorithm, we shall require some additional insights into its structure. First, for computational reasons, the description of the tangent space by (5.1) is not suitable since $\Delta \in \mathbb{R}^{n \times n}$. In the next subsection, we will therefore derive a more efficient representation. Next, after having specified the metric, we will show that the Riemannian gradient and Hessian can be computed by means of orthogonal projections onto the tangent space. Finally, we will construct the retraction mapping $R_x : T_x \mathcal{M} \rightarrow \mathcal{M}$ and outline some of its properties. In addition, we will construct a second-order expansion of this retraction that can be used to compute the Riemannian Hessian analytically.

5.1. The tangent space. Before deriving the tangent space, we need an efficient representation of the elements $x \in \mathcal{M}$. Since every s.p.s.d. matrix of rank k can be factorized as the product of an $n \times k$ -matrix with its transpose, we make the obvious choice of $x = YY^\top$ with $Y \in \mathbb{R}_*^{n \times k}$. Although this representation is not unique nor optimal since $\dim \mathcal{M} < nk$, this is not a problem: uniqueness of Y will never be required in our algorithm, and, moreover, the representation is not far from optimal anyway.

The dimension of the tangent space is, by definition, the dimension of the manifold. However, expression (5.1) for $T_x \mathcal{M}$ is clearly an over-parametrization. A minimal parametrization is given by the following proposition.

PROPOSITION 5.2. *The tangent space of \mathcal{M} at $x = YY^\top$ is given by*

$$T_x \mathcal{M} = \left\{ [Y \quad Y_\perp] \begin{bmatrix} S & N^\top \\ N & 0 \end{bmatrix} \begin{bmatrix} Y^\top \\ Y_\perp^\top \end{bmatrix} : S \in \mathbb{S}_k^{sym}, N \in \mathbb{R}^{n-k \times k} \right\}. \quad (5.2)$$

Proof. The right-hand side of (5.2) has the correct number of degrees of freedom, it is a linear space and by taking $\Delta = (YS/2 + Y_\perp N)(Y^\top Y)^{-1}Y^\top$, it is included in $T_x \mathcal{M}$ of (5.1). \square

This representation shows that we can parametrize every tangent vector with the matrices S and N and this representation is optimal. Note that an algorithm will not require explicit computation and storage of Y_\perp , which would be unacceptably expensive. It will require only products of the form $Y_p := Y_\perp N \in \mathbb{R}^{n \times k}$, see also §6.3.

5.2. The Riemannian metric and gradient. Until now we have not mentioned the choice of the metric. This was deliberate because the tangent space itself does not depend on the metric. The metric is important however if one wants to optimize an objective function, e.g., to define the gradient. The aim is to smoothly assign at each point x an inner product $\langle \xi, \psi \rangle_x$ for all tangent vectors $\xi, \psi \in T_x \mathcal{M}$. This will turn \mathcal{M} into a Riemannian manifold.

Because \mathcal{M} is an embedded submanifold of $\mathbb{R}^{n \times n}$ and $T_x \mathcal{M} \subset \mathbb{R}^{n \times n}$, the choice of the classic Euclidean inner product

$$\langle \xi, \psi \rangle_x := \text{tr}(\xi^\top \psi) \quad (5.3)$$

as the *Riemannian metric* is obvious. Since this metric does not depend on x , we will drop the subscript x in the notations. The induced norm will be denoted by $\|\xi\| := \sqrt{\langle \xi, \xi \rangle}$. Furthermore, orthogonality will always mean orthogonal w.r.t. (5.3).

Using the metric, we can define the normal space $N_x \mathcal{M}$ as the complementary subspace, orthogonal to $T_x \mathcal{M}$

$$N_x \mathcal{M} := \left\{ [Y \quad Y_\perp] \begin{bmatrix} C & -M^\top \\ M & K \end{bmatrix} \begin{bmatrix} Y^\top \\ Y_\perp^\top \end{bmatrix} : C \in \mathbb{S}_k^{\text{skew}}, M \in \mathbb{R}^{n-k \times k}, K \in \mathbb{R}^{n-k \times n-k} \right\}.$$

Counting dimensions and verifying that $\langle \xi, \nu \rangle = 0$ for all $\xi \in T_x \mathcal{M}$ and $\nu \in N_x \mathcal{M}$, it is easy to see that $T_x \mathcal{M} \oplus N_x \mathcal{M} = \mathbb{R}^{n \times n}$. Additionally, we will denote the orthogonal projection along $N_x \mathcal{M}$ of a matrix $Z \in \mathbb{R}^{n \times n}$ onto the tangent space at x by $P_x(Z)$.

The following orthogonal projectors will be convenient later on:

$$P_x^s : \mathbb{R}^{n \times n} \rightarrow \{Y S Y^\top : S \in \mathbb{S}_k^{\text{sym}}\}, \quad (5.4)$$

$$P_x^p : \mathbb{R}^{n \times n} \rightarrow \{Y_\perp N Y^\top + Y N^\top Y_\perp^\top : N \in \mathbb{R}^{n-k \times k}\}. \quad (5.5)$$

Observe that $\text{range}(P_x^s) \perp \text{range}(P_x^p)$ and that $P_x = P_x^s + P_x^p$ which follows from $\text{range}(P_x^s) \cup \text{range}(P_x^p) = T_x \mathcal{M} = \text{range}(P_x)$. Considering Prop. (5.2), we see that every tangent vector $\xi \in T_x \mathcal{M}$ can be decomposed into two mutually orthogonal parts $\xi = \xi^s + \xi^p$ with $\xi^s := P_x^s(\xi)$ and $\xi^p := P_x^p(\xi)$. Using the standard orthogonal projectors onto the column span of Y and Y^\perp , namely $P_Y = Y(Y^\top Y)^{-1}Y^\top$ and $P_Y^\perp = I - P_Y$, we can further specify the projectors (5.4)–(5.5) as

$$P_x^s : Z \mapsto P_Y \frac{Z + Z^\top}{2} P_Y, \quad (5.6)$$

$$P_x^p : Z \mapsto P_Y^\perp \frac{Z + Z^\top}{2} P_Y + P_Y \frac{Z + Z^\top}{2} P_Y^\perp, \quad (5.7)$$

$$P_x : Z \mapsto P_x^s(Z) + P_x^p(Z). \quad (5.8)$$

Since these projectors are linear operators, they can be represented as matrices. We can do this by vectorizing $\mathbb{R}^{n \times n}$ as in §3, so that the projectors have \mathbb{R}^{n^2} as domain. Applying the $\text{vec}(\cdot)$ operator to the expressions (5.6)–(5.7), we obtain the $n^2 \times n^2$ matrices

$$P_x^s = \frac{1}{2}(P_Y \otimes P_Y)(I + \Pi), \quad (5.9)$$

$$P_x^p = \frac{1}{2}(P_Y \otimes P_Y^\perp + P_Y^\perp \otimes P_Y)(I + \Pi), \quad (5.10)$$

$$P_x = P_x^s + P_x^p. \quad (5.11)$$

Matrix Π is the symmetric permutation matrix, known in [44] as the *perfect shuffle* $S_{n,n}$, that satisfies $\text{vec}(A^\top) = \Pi \text{vec}(A)$. To verify the symmetry of the projection matrices, we can use the property that Π allows one to switch Kronecker products as follows: $\Pi(A \otimes B)\Pi = B \otimes A$, for square A, B of equal size.

We will use the *Riemannian gradient* of an objective function f defined on \mathcal{M} in our optimization algorithm. This gradient, denoted as $\text{grad } f$, has the well-known interpretation of the direction of steepest ascent, but restricted to \mathcal{M} ,

$$\frac{\text{grad } f(x)}{\|\text{grad } f(x)\|} = \arg \max_{\xi \in T_x \mathcal{M}, \|\xi\|=1} \text{D } f(x)[\xi].$$

In addition, the gradient can be identified from the relation

$$\langle \text{grad } f(x), \xi \rangle = \text{D } f(x)[\xi], \quad \text{for all } \xi \in T_x \mathcal{M}. \quad (5.12)$$

With the aid of the following theorem, the Riemannian gradient can be computed also by the orthogonal projection P_x of the gradient in the embedding space.

THEOREM 5.3 ([3, Ch. 3.6]). *Suppose a function $f : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ has matrix G_x as Euclidean gradient in point $x \in \mathcal{M}$, then the Riemannian gradient of $f : \mathcal{M} \rightarrow \mathbb{R}$ is given by $\text{grad } f(x) = P_x(G_x)$.*

5.3. The Riemannian Hessian. The RTR method uses a second-order model that is based on the Hessian. In case of a Riemannian manifold, the *Riemannian Hessian* of a function f at $x \in \mathcal{M}$ is the unique symmetric and linear operator $\text{Hess } f(x) : T_x \mathcal{M} \rightarrow T_x \mathcal{M}$ that satisfies [15]

$$\langle \text{Hess } f(x)[\xi], \xi \rangle = \left. \frac{d^2}{dt^2} \right|_{t=0} f(\gamma_\xi(t)), \quad (5.13)$$

where $\gamma_\xi(t)$ is a geodesic with $\gamma_\xi(0) = x$ and $\dot{\gamma}_\xi(0) = \xi$. In differential geometry, this Hessian has a rigorous meaning by aid of the Levi-Civita connection. We will not explore this further, except from noting that it is a standard result that this connection is unique [10, Th. 3.3].

Since \mathcal{M} is embedded into $\mathbb{R}^{n \times n}$, the Hessian can be explicitly constructed by taking a classic directional derivative of the Riemannian gradient, followed by an orthogonal projection onto $T_x \mathcal{M}$, see [3, Prop. 5.3.2]. This gives

$$\text{Hess } f(x)[\eta] = P_x(\text{D grad } f(x)[\eta]).$$

Due to the presence of the projectors it may be rather difficult to derive the Hessian in this way. In §6.1 we will therefore use another approach that involves a second-order approximation of the exponential mapping.

5.4. The retraction. As explained in §4, we need a mapping, called the retraction R_x , to map updates in the tangent space onto the manifold. There is considerable freedom in this, but a well-chosen retraction is crucial for the performance of a Riemannian optimization algorithm. A generic choice for a retraction $R_x(\xi)$ is the so-called exponential mapping [10, Def. VII.6.3]. It is defined as the image of the geodesic $\gamma_\xi(t)$ at $t = 1$ and it always exists in a neighborhood $B_x \subset T_x \mathcal{M}$. However, the use of the exponential mapping is computationally expensive and not really necessary. Almost any smooth mapping that maps tangent vectors rigidly onto the manifold suffices and will not hamper convergence. These retractions have to fulfill some properties.

DEFINITION 5.4 ([3, Def. 4.1.1]). A first-order retraction on \mathcal{M} is a smooth mapping R from the tangent bundle $T\mathcal{M}$ onto \mathcal{M} with the following properties. Let R_x be the restriction of R to $T_x\mathcal{M}$, then

1. $R_x(0) = x$,
2. Local rigidity: For every tangent vector $\xi \in T_x\mathcal{M}$, the curve $\gamma_\xi : t \mapsto R_x(t\xi)$ satisfies $\dot{\gamma}_\xi(0) = \xi$.

As the presence of “first-order” suggests, these retractions approximate the exponential mapping up to first order. The next step is a *second-order* retraction where the second-order derivative must be interpreted in the sense of §5.3.

DEFINITION 5.5 ([3, Prop. 5.5.5]). A second-order retraction on \mathcal{M} is a first-order retraction which satisfies in addition the zero initial acceleration condition:

$$P_x \left(\left. \frac{d^2}{dt^2} \right|_{t=0} R_x(t\xi) \right) = 0 \quad \text{for all } \xi \in T_x\mathcal{M}.$$

As far as convergence of Riemannian optimization methods goes, first-order accuracy is sufficient [3, Ch. 4], but second-order retractions enjoy a very nice property: the Riemannian Hessian of a cost function f coincides with the Euclidean Hessian of the lifted cost function $\hat{f}_x := f \circ R_x$.

THEOREM 5.6 ([3, Prop. 5.5.5]). Let R_x be a second-order retraction on \mathcal{M} , then

$$\text{Hess } f(x) = \text{Hess}(f \circ R_x)(0) \quad \text{for all } x \in \mathcal{M}.$$

A popular retraction is simply the projection onto \mathcal{M} ,

$$R_x^P(\xi) := P_{\mathcal{M}}(x + \xi) \tag{5.14}$$

where operator $P_{\mathcal{M}}$ selects the nearest element to \mathcal{M} in the Frobenius norm, i.e.,

$$P_{\mathcal{M}} : \mathbb{R}^{n \times n} \rightarrow \mathcal{M}, \quad X \mapsto \arg \min_{z \in \mathcal{M}} \|X - z\|.$$

It is used both in the general context of retraction based Riemannian optimization (see [3] and the references therein) and in the specific context of low-rank solvers for matrix equations, see [21] and [20]. It will come as no surprise that this choice owes greatly to the fact that we minimize the error after retraction and stay as close to the manifold as possible. However, there is a caveat: since we project onto a non-convex set, the projection is not always well-defined, except in a (possibly very small) neighborhood of x . The following proposition gives a characterization of this projection and shows that the retraction is not defined if the matrix to project has not enough positive eigenvalues.

THEOREM 5.7 ([22, Corr. 2.3]). Let $A \in S_n^{\text{sym}}$ have n_+ positive and n_- negative eigenvalues. Let its eigenvalue decomposition be $A = V \text{diag}(\lambda_1, \dots, \lambda_n) V^\top$ with $\lambda_1 \geq \dots \geq \lambda_{n_+} > 0 > \lambda_{n-n_++1} \geq \dots \geq \lambda_n$. The best s.p.s.d. approximation of rank k in the Frobenius norm exists if and only if $k \geq n_+$. One such minimizer is given by

$$P_{\mathcal{M}}(A) = V \text{diag}(\lambda_1, \dots, \lambda_k, 0, \dots, 0) V^\top.$$

We shall use this theorem to compute the retraction provided $k \geq n_+$. Although in theory, the retraction has to be defined only in a neighborhood of each tangent

space, in practice, it is desirable to have a retraction defined on the whole tangent bundle $T\mathcal{M}$. However, this is not the case for the retraction (5.14) but we can give sufficient conditions for the existence of the retraction.

PROPOSITION 5.8. *Suppose $n \geq 2k$. Retraction $R_x^P(\xi)$ defined by (5.14) exists if*

- (i) *the rank of $P_x^p(\xi)$ is $2k$, or*
- (ii) *$P_x^s(x + \xi)$ has k strictly positive eigenvalues.*

Condition (ii) can always be satisfied for ξ small enough.

Proof. Elaboration of $x + \xi$ with $x = YY^\top$ and $\xi \in T_x\mathcal{M}$ as in Prop. (5.2), gives

$$x + \xi = \begin{bmatrix} Y & Y_\perp \end{bmatrix} T \begin{bmatrix} Y^\top \\ Y_\perp^\top \end{bmatrix} \quad \text{with } T = \begin{bmatrix} I + S & N^\top \\ N & 0 \end{bmatrix}, \quad S \in \mathbb{S}_k^{\text{sym}}, \quad N \in \mathbb{R}^{n-k \times k}.$$

Th. 5.7 guarantees that the retraction exists if the $n \times n$ matrix T has at least k positive eigenvalues. By [34, Th. 16.6], we have that $\text{inertia}(T) = \text{inertia}(Z^\top(I+S)Z) + (p, p, 0)$ with $p = \text{rank}(N)$ and Z a basis for the null space of N . Since $n \geq 2k$, p equals the column rank of N and Z will have $k - p \leq k$ columns. Suppose condition (i) is true, then N will be full rank, so $p = k$, and T will have at least k strictly positive eigenvalues. If condition (ii) is satisfied, then all k eigenvalues of $I + S$ are strictly positive. Now all $k - p$ eigenvalues of $Z^\top(I + S)Z$ are strictly positive eigenvalues as well and T will have $k - p + p = k$ strictly positive eigenvalues. Since $I + S \succ 0$ for S small enough, the result follows. \square

We should remark that in practice we have observed that the retraction almost always exists, because condition (i) is almost never violated.

PROPOSITION 5.9. *Retraction R_x^P defined by (5.14) is a second-order retraction*

Proof. It is clear that (5.14) defines a smooth mapping $T_x\mathcal{M} \rightarrow \mathcal{M}$ since it only involves smooth matrix operations. The property $R_x^P(0) = x$ is trivially satisfied.

First-order: The local rigidity condition follows from Lemma 2.1 in [30] for projections on general manifolds, which states that $\text{grad} P_{\mathcal{M}}(x) = P_x$. The existence in a neighborhood of each x follows from Prop. 5.8 for ξ small enough.

Second-order: Slight variation of [32, Th. 4.9]. We claim that $R_x^P(t\xi) - \gamma_\xi(t) = O(t^3)$ with $\gamma_\xi(t)$ the geodesic with foot x and direction ξ . Observing that $\gamma_\xi(t) \in \mathcal{M}$ for all t and expanding $\gamma_\xi(t)$ in series, we have the following error term

$$\begin{aligned} R_x^P(t\xi) - \gamma_x(t) &= R_x^P(t\xi) - P_{\mathcal{M}} \gamma_x(t) \\ &= R_x^P(t\xi) - P_{\mathcal{M}}[x + t\dot{\gamma}_x(0) + \frac{1}{2}t^2\ddot{\gamma}_x(0) + O(t^3)]. \end{aligned}$$

Since $\dot{\gamma}_\xi(0) = \xi$ and $\ddot{\gamma}_\xi(t)$ belongs to the normal space at $\gamma_\xi(t)$, the first order terms cancel and the second order term is projected out by $P_{\mathcal{M}}$. Since all geodesics $\gamma_\xi(t)$ are locally the image of the exponential mapping in x [10, Th. VII.6.9], the retraction R_x^P is of second order. \square

Since R_x^P is a second-order retraction, we could derive the Riemannian Hessian of the objective function f by computing the Euclidean Hessian of $f \circ R_x^P$, see Th. 5.6. However, in order to derive an analytical expression of the Hessian, as we will do in §6.1, it is convenient to have a second-order expansion of the retraction as well. Retraction R_x^P is not readily available in series, but we can construct an equivalent expansion by carefully inspecting $x + \xi$ if we split ξ into $P_x^s(\xi) + P_x^p(\xi)$.

PROPOSITION 5.10. *For any $x \in \mathcal{M}$, with $x^\dagger \in \mathcal{M}$ its pseudo-inverse [17], the mapping $R_x^{(2)} : T_x\mathcal{M} \rightarrow \mathcal{M}$, given by*

$$R_x^{(2)} : \xi \mapsto wx^\dagger w^\top, \quad \text{with } w = x + \frac{1}{2}\xi^s + \xi^p - \frac{1}{8}\xi^s x^\dagger \xi^s - \frac{1}{2}\xi^p x^\dagger \xi^s, \quad (5.15)$$

where $\xi^s = P_x^s(\xi)$ and $\xi^p = P_x^p(\xi)$, is a second-order retraction on \mathcal{M} .

Proof. First, we show that $R_x^{(2)}$ is a retraction, i.e. a mapping $B_x \rightarrow \mathcal{M}$ in a neighborhood $B_x \subset T_x\mathcal{M}$. Choose $x = YY^\top$, then $x^\dagger = Y(Y^\top Y)^{-2}Y^\top$. This allows us to write the components of the tangent vector $\xi = \xi^s + \xi^p$ as $\xi^s = YSY^\top$ and $\xi^p = Y_\perp NY^\top + YN^\top Y_\perp^\top$. Elaborating the term w in (5.15) and using the relations $\xi^p x^\dagger = Y_\perp NY^\top x^\dagger$, $\xi^s x^\dagger \xi^s = YS^2 Y^\top$ and $\xi^p x^\dagger \xi^s = Y_\perp NSY^\top$ we arrive at

$$w = (Y + \frac{1}{2}YS + Y_\perp N - \frac{1}{8}YS^2 - \frac{1}{2}Y_\perp NS)Y^\top = ZY^\top$$

where we introduced the matrix $Z \in \mathbb{R}^{n \times k}$. Since $Y^\top x^\dagger Y = I_k$, we see that $R_x^{(2)}$ can be written as $R_x^{(2)} : \xi \mapsto wx^\dagger w^\top = ZZ^\top$. Thus, the image of $R_x^{(2)}$ consists of s.p.s.d. matrices of rank less than or equal to k . Furthermore, there will always be a neighborhood of $T_x\mathcal{M}$ that results in matrices Z of full rank k (take S small enough). Next, we prove that $R_x^{(2)}$ is of second-order. Expanding $wx^\dagger w$ fully up to second-order terms in ξ^s and ξ^p and using the substitutions $xx^\dagger \xi^s = \xi^s xx^\dagger = \xi^s$ and $xx^\dagger \xi^p + \xi^p xx^\dagger = \xi^p$ we see that many of the second-order terms cancel. Finally, we obtain

$$R_x^{(2)}(\xi) = x + \xi^s + \xi^p + \xi^p x^\dagger \xi^p + O(\|\xi\|^3).$$

From this $R_x^{(2)}(0) = x$ and local rigidity (first-order) are obvious. Since $\xi^p x^\dagger \xi^p = Y_\perp N^2 Y_\perp^\top \in N_x \mathcal{M}$, zero acceleration (second-order) is proved. \square

6. The Riemannian Trust-Region method. We will use the trust-region algorithm adapted for Riemannian manifolds from [1], as listed in Algorithm 1. Except for the model definition, which we will explain in §6.1, only the step calculation is different from a classic trust-region method. Since we are dealing with large-scale problems, a numerically exact minimization of the trust-region subproblem (6.1) using direct methods is not feasible. There are several solutions to lower the numerical burden of which the preconditioned truncated conjugate-gradient (tCG) [43], [42] and truncated generalized Lanczos [18] methods are some of the most popular. Both can easily be adapted to Riemannian manifolds since they only need evaluations of inner products and matrix-vector multiplications with the Hessian. We will use tCG since in our numerical tests, truncated Lanczos did not perform better than tCG. The updating strategies can be chosen as in the usual framework, see [34], [14].

6.1. Second-order model. The RTR method needs a second-order model of f around x . A convenient choice is to use a quadratic model m_x of the lifted objective function $\widehat{f}_x := f \circ R_x$, see [3, Ch. 7]. When R_x is a second-order retraction, this model can be based on the Riemannian gradient and Hessian, i.e.,

$$m_x : T_x \mathcal{M} \rightarrow \mathbb{R}, \xi \mapsto f(x) + \langle \text{grad } f(x), \xi \rangle + \frac{1}{2} \langle \text{Hess } f(x)[\xi], \xi \rangle.$$

Making use of the properties (5.12) and (5.13), it is not difficult to show that $m_x(\xi)$ is indeed accurate up to second order, i.e., $|m_x(\xi) - f(R_x(\xi))| = O(\|\xi\|^3)$.

Thanks to Th. 5.6 we can build this model by taking a classic Taylor expansion

Algorithm 1 Riemannian Trust-Region (RTR) [1] with TR strategy from [34]

Require: $\bar{\Delta} > 0, \Delta_1 \in (0, \bar{\Delta})$

1: **for** $k = 1, 2, \dots$ **do**

2: **Model definition:** define the second-order model

$$m_k : T_{x_k} \mathcal{M} \rightarrow \mathbb{R}, \xi \mapsto f(x_k) + \langle \text{grad } f(x_k), \xi \rangle + \frac{1}{2} \langle \text{Hess } f(x_k)[\xi], \xi \rangle$$

3: **Step calculation:** compute η_k by (approximately) solving

$$\eta_k = \arg \min m_k(\xi) \quad \text{s.t. } \|\xi\| \leq \Delta_k \quad (6.1)$$

4: **Acceptance of trial point:** compute $\rho_k = (\hat{f}(0) - \hat{f}_{x_k}(\eta_k)) / (m_k(0) - m_k(\eta_k))$

5: **if** $\rho_k \geq 0.05$ **then**

6: accept step and set $x_{k+1} = R_{x_k}^P(\eta_k)$

7: **else**

8: reject step and set $x_{k+1} = x_k$

9: **end if**

10: **Trust-Region radius update:** set

$$\Delta_{k+1} = \begin{cases} \min(2\Delta_k, \bar{\Delta}) & \text{if } \rho_k \geq 0.75 \text{ and } \|\eta_k\| = \Delta_k, \\ 0.25 \|\eta_k\| & \text{if } \rho_k \leq 0.25, \\ \Delta_k & \text{otherwise.} \end{cases}$$

11: **end for**

of $\hat{f}_x := f_x \circ R_x^{(2)}$ with $R_x^{(2)}$ the second-order expansion (5.15). Let $\xi^p = P_x^p(\xi)$, then

$$\begin{aligned} \hat{f}_x(\xi) &= f(R_x^{(2)}(\xi)) \\ &= f(x + \xi + \xi^p x^\dagger \xi^p + O(\xi^3)) \\ &= \text{tr} [(x + \xi + \xi^p x^\dagger \xi^p + O(\xi^3))A(x + \xi + \xi^p x^\dagger \xi^p + O(\xi^3))M] \\ &\quad - \text{tr} [(x + \xi + \xi^p x^\dagger \xi^p + O(\xi^3))C] \\ &= f(x) + \text{tr}(\xi R) + \text{tr}(\xi A \xi M - \xi^p R \xi^p x^\dagger) + O(\|\xi\|^3) \end{aligned}$$

where R is the residual of x , i.e., $R := AxM + MxA - C$. In the truncated expression we can easily recognize the terms that contribute to the gradient and the Hessian, namely

$$\langle \xi, \text{grad } f(x) \rangle = \text{tr}[\xi R] \quad \text{and} \quad \langle \xi, \text{Hess } f(x)[\xi] \rangle = 2 \text{tr}[\xi A \xi M - \xi^p R \xi^p x^\dagger].$$

Next, we need to manipulate these expressions to obtain the gradient as a tangent vector of $T_x \mathcal{M}$ and the Hessian as a linear and symmetric mapping of $T_x \mathcal{M} \rightarrow T_x \mathcal{M}$. This can be done by judiciously inserting some projectors, e.g. $P_x(\xi) = \xi$.

The computation of the gradient is almost trivial since $R = R^\top$

$$\langle \xi, \text{grad } f(x) \rangle = \text{tr}(\xi R) = \langle \xi, R \rangle = \langle P_x(\xi), R \rangle = \langle \xi, P_x(R) \rangle$$

and so we obtain

$$\text{grad } f(x) = P_x(AxM + MxA - C). \quad (6.2)$$

We recover the gradient as the orthogonal projection onto $T_x\mathcal{M}$ of the gradient in the full space, as it should be according to Th. 5.3.

As for the Hessian, we need additionally $\xi^p = P_x^p(\xi)$,

$$\begin{aligned} \langle \xi, \text{Hess } f(x)[\xi] \rangle &= 2 \text{tr}(\xi A \xi M - \xi^p R \xi^p x^\dagger) \\ &= 2 \langle \xi, A \xi M \rangle - 2 \langle \xi^p, R \xi^p x^\dagger \rangle \\ &= 2 \langle \xi, P_x(A \xi M) \rangle - 2 \langle P_x^p(\xi), R P_x^p(\xi) x^\dagger \rangle \\ &= \langle \xi, 2 P_x(A \xi M) - 2 P_x^p(R P_x^p(\xi) x^\dagger) \rangle \end{aligned}$$

and so

$$\begin{aligned} \text{Hess } f(x)[\xi] &= 2 P_x(A \xi M) - 2 P_x^p(R P_x^p(\xi) x^\dagger) \\ &= P_x(A \xi M + M \xi A) - P_x^p(R P_x^p(\xi) x^\dagger + x^\dagger P_x^p(\xi) R). \end{aligned} \quad (6.3)$$

From looking at this expression, it may not be clear whether all properties of a Hessian are satisfied. Luckily, we can get a much more familiar representation of the Hessian after vectorization. Recalling the derivation of the matrices (5.9)–(5.11), we apply the $\text{vec}(\cdot)$ operator to (6.3),

$$\begin{aligned} \text{vec}(\text{Hess } f(x)[\xi]) &= \text{vec}(P_x(A \xi M + M \xi A) - P_x^p(R P_x^p(\xi) x^\dagger + x^\dagger P_x^p(\xi) R)) \\ &= P_x \text{vec}(A \xi M + M \xi A) - P_x^p \text{vec}(R P_x^p(\xi) x^\dagger + x^\dagger P_x^p(\xi) R) \\ &= P_x(A \otimes M + M \otimes A) \text{vec}(P_x \xi) - P_x^p(R \otimes x^\dagger + x^\dagger \otimes R) \text{vec}(P_x^p \xi) \\ &= [P_x(A \otimes M + M \otimes A) P_x - P_x^p(R \otimes x^\dagger + x^\dagger \otimes R) P_x^p] \text{vec}(\xi). \end{aligned}$$

Finally, the Hessian is given by the matrix

$$\mathcal{H}_x := P_x(A \otimes M + M \otimes A) P_x - P_x^p(x^\dagger \otimes R + R \otimes x^\dagger) P_x^p. \quad (6.4)$$

This matrix is clearly a linear and symmetric operator and due to the presence of P_x and P_x^p , it has $T_x\mathcal{M}$ as its domain and range.

If we compare this Hessian to the Hessian of the full space, $\mathcal{L} = A \otimes M + M \otimes A$, we see that besides the expected projector P_x there is a “correction term” due to curvature of the low-rank constraint. Furthermore, this term can make the Hessian indefinite and renders the optimization problem non-convex (also in the Riemannian sense). This shows the need for a robust modification of Newton’s method and motivated the TR approach.

The correction term in \mathcal{H}_x is necessary to have a correct second-order model, as we illustrate in Fig. 6.1. There we have plotted the maximum relative error of two models in function of the norm of the tangent vector for 1000 random vectors. The first model, denoted in black, uses \mathcal{H}_x as the Hessian and the second model, in white, uses $P_x \mathcal{L} P_x$. In addition to the second-order retraction $R_x^{(2)}$, shown with \circ , we have also used the projection based retraction R_x^P , shown with \diamond . It is clearly visible that only the model with \mathcal{H}_x as the Hessian gives rise to a second-order model. From the figure, we can also see that R_x^P does not deteriorate the second-order accuracy. Indeed, this is to be expected since this retraction is also of second-order.

6.2. Final algorithm. In principle, the previous optimization formulated in Algorithm 1 suffices to find a low-rank approximation if the rank is known in advance.

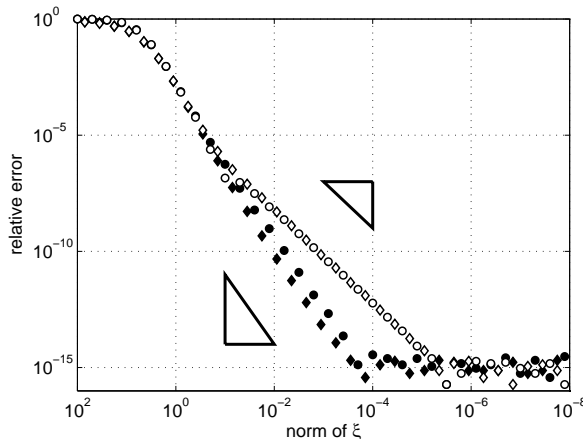


FIG. 6.1. Relative error of the linear and quadratic models. The triangles indicate the second and third order convergence of the error.

In practice however, this rank is unknown since one is usually interested in an approximation that is better than a certain tolerance. We will take the relative residual in the Frobenius norm, i.e. $\|AXM + MXA - C\| / \|C\|$, as tolerance.

We therefore propose Algorithm 2 that computes a series of local optimizers to problem (2.3) with increasing rank k until the tolerance is satisfied. In order to ensure a monotonic decrease of the cost function, and thus the error, we can reuse the previous solution Y_i and append a zero column. Since the zero column does not increase the rank of $Y_{i+1} = [Y_i \ 0]$, the Hessian would be singular due to $x^\dagger \notin \mathcal{M}$ in (6.3) and so we cannot use Y_{i+1} as initial guess for Algorithm 1. Instead, we perform one steepest descent step to obtain a full rank matrix Y_{i+1} . After that, we can find a minimizer with Algorithm 1.

Due to numerical cancellation, the residual should not be computed based on the expression $\text{tr}(R_i R_i)$ with $R_i = AY_i Y_i^\top M + MY_i Y_i^\top A - C$. Instead, we follow a similar approach as in [36]. Suppose $x_i = Y_i Y_i^\top$ and $C = bb^\top$ and we have computed the reduced QR factorization $[AY_i \ MY_i \ b] = Q_i R_i$. Then we can express the relative residual as

$$\begin{aligned} r_i &= \|Ax_i M + Mx_i A + C\| / \|C\| = \left\| [AY_i \ MY_i \ b] [MY_i \ AY_i \ b]^\top \right\| / \|bb^\top\| \\ &= \left\| R_i \begin{bmatrix} 0 & I_k & 0 \\ I_k & 0 & 0 \\ 0 & 0 & I_l \end{bmatrix} R_i^\top \right\| / \|b^\top b\|. \end{aligned} \quad (6.5)$$

The computational cost of r_i stays very moderate since the rank is small and we only need to compute it once the minimizer is found. This is in contrast to [36] where the computation of the residual is the most expensive step of the whole algorithm.

6.3. Implementation aspects. We shall assume that the matrices A and M have a fast matrix vector product at the cost of $O(n)$. This is a very reasonable assumption when we are dealing with a discretized PDE where A represents a sparse system matrix and M is a sparse mass matrix. Likewise, we assume an $O(n)$ matrix vector product for the matrix C . Normally C is not a sparse matrix but it is *data-sparse*. In many control applications, C is the outer product of the input or output

Algorithm 2 Final algorithm: Lyap-RTR

Require: Initial guess $x_1 = Y_1 Y_1^\top$ with $Y_1 \in \mathbb{R}_*^{n \times k_1}$, residual tolerance τ

- 1: **for** $i = 1, 2, \dots$ **do**
 - 2: **Find** x_i **as a minimizer of** (2.3):
 perform Algorithm 1 with \mathcal{M} the manifold of rank k_i s.p.s.d. matrices
 - 3: **Compute the relative residual of** x_i : calculate r_i based on (6.5)
 - 4: **if** $r_i \leq \tau$ **then**
 - 5: **Solution found:** return x_i and quit
 - 6: **else**
 - 7: **Increase rank:** $k_{i+1} = k_i + 1$
 - 8: **Compute initial guess** $x_{i+1} = Y_{i+1} Y_{i+1}^\top$:
 perform one step of steepest descent on $Y_{i+1} = [Y_i \ 0]$
 - 9: **end if**
 - 10: **end for**
-

matrices, e.g. $C = cc^\top$ with $c \in \mathbb{R}^{n \times l}$, $l \ll n$.

Factorization. Instead of using the factorization YY^\top , we store x as VDV^\top with $V \in \mathbb{R}_*^{n \times k}$ orthonormal and $D \in \mathbb{R}_*^{k \times k}$ diagonal, i.e., as a truncated eigenvalue decomposition. This is only slightly more costly but has the advantage that the projector P_Y can be applied as $P_V = VV^\top$. The orthogonal projection onto $T_x \mathcal{M}$, given by (5.8), becomes

$$P_x(Z) = VV^\top \frac{Z + Z^\top}{2} VV^\top + (I - VV^\top) \frac{Z + Z^\top}{2} VV^\top + VV^\top \frac{Z + Z^\top}{2} (I - VV^\top).$$

Tangent vectors. We will also use this factorization for the tangent vectors. Suppose $x = YY^\top = VDV^\top$, then $Y = VM$ for some $M \in \mathbb{R}_*^{k \times k}$. According to Prop. 5.2, a tangent vector $\xi \in T_x \mathcal{M}$ in $x = YY^\top$ is parametrized as

$$\xi = YUY^\top + Y_\perp NY^\top + YN^\top Y_\perp^\top, \quad U \in \mathbb{S}_k^{\text{sym}}, N \in \mathbb{R}^{n-k \times k}.$$

Since $Y = VM$, the same ξ can be stored as

$$\xi = VSV^\top + ZV^\top + VZ^\top, \quad S = MUM^\top \in \mathbb{S}_k^{\text{sym}}, Z = Y_\perp NM^\top \in \mathbb{R}^{n \times k}.$$

So we only need to compute and store the matrices S and Z .

Objective function. Evaluation of $f(x)$ in $x = VDV^\top$ can be done efficiently since

$$f(x) = \text{tr}(xAxM) - \text{tr}(xC) = \text{tr}[(V^\top AV)D(V^\top MV)D] - \text{tr}[(V^\top CV)D].$$

The vectors AV , MV and CV will be useful later on so we store them after each calculation of $f(x)$.

Gradient. The gradient at $x = VDV^\top$ is given by the projection of the residual $R = AVDV^\top M + MVDV^\top A - C$ onto $T_x \mathcal{M}$, see eq. (6.2). With the previous explained projectors, this becomes

$$\text{grad } f(x) = VV^\top R V V^\top + (I - VV^\top) R V V^\top + VV^\top R (I - VV^\top).$$

After some manipulations and rearranging the terms for efficiency, we obtain that $\text{grad } f(x)$ equals the tangent vector $VSV^\top + ZV^\top + VZ^\top$ with

$$\begin{aligned} T &= (AV)D(V^\top MV) + (MV)D(V^\top AV) - (CV), \\ S &= V^\top T, \\ Z &= T - VS. \end{aligned}$$

Hessian. The Hessian at x evaluated for $\xi = VS_\xi V^\top + Z_\xi V^\top + VZ_\xi^\top$ is given by eq. (6.3) where $x^\dagger = VD^{-1}V^\top$. After similar but slightly more tedious manipulations as for the gradient, we get that $\text{Hess } f(x)[\xi] = VSV^\top + ZV^\top + VZ^\top$ with

$$\begin{aligned} T_1 &= (AV)S_\xi(V^\top MV) + (MV)S_\xi(V^\top AV) \\ &\quad + (AV)(Z_\xi^\top MV) + (MV)(Z_\xi^\top AV) + (AZ_\xi)(V^\top MV) + (MZ_\xi)(V^\top AV), \\ T_2 &= (AV)D(V^\top MZ_\xi) + (MV)D(V^\top AZ_\xi) - (CZ_\xi), \\ S &= V^\top T_1, \\ Z &= T_1 - V(V^\top T_1) + (T_2 - V(V^\top T_2))D^{-1}. \end{aligned}$$

The dominating costs are the matrix vector products AZ_ξ , MZ_ξ and CZ_ξ .

Retraction. The projection-based retraction $R_x^P(\xi)$ can be computed with an eigenvalue decomposition of $x + \xi$, see Theorem 5.7. In general, computing this decomposition with a direct method implies an $O(n^3)$ cost. Luckily, we can exploit the fact that we only need to retract from the tangent space. Suppose we need to retract $\xi = VSV^\top + ZV^\top + VZ^\top$ in the point $x = VDV^\top$, then $x + \xi$ can be written as

$$x + \xi = \begin{bmatrix} V & V_p \end{bmatrix} \begin{bmatrix} D + S & R^\top \\ R & 0 \end{bmatrix} \begin{bmatrix} V^\top \\ V_p^\top \end{bmatrix}$$

with $Z = V_p R$ a reduced QR factorization. Observe that because $V^\top Z = 0$, we have that $V^\top V_p = 0$ and thus $\begin{bmatrix} V & V_p \end{bmatrix}$ is orthonormal. Since the Frobenius norm is unitary invariant, it suffices to compute the eigenvalue decomposition of a small $2k \times 2k$ matrix to project $x + \xi$:

$$R_x^P(\xi) = P_{\mathcal{M}}(x + \xi) = \begin{bmatrix} V & V_p \end{bmatrix} P_{\widetilde{\mathcal{M}}} \left(\begin{bmatrix} D + S & R^\top \\ R & 0 \end{bmatrix} \right) \begin{bmatrix} V^\top \\ V_p^\top \end{bmatrix}.$$

Here $\widetilde{\mathcal{M}}$ is the manifold of s.p.s.d. matrices of size $2k$ and rank k . This brings the dominating costs of this retraction to $O(k^2n)$ for the reduced QR and $O(k^3)$ for the eigenvalue decomposition.

7. Preconditioning. The computationally most expensive step of the RTR method is the solution of the TR subproblems (6.1). Since these problems are possibly very large, we solve them iteratively with the truncated CG (tCG) algorithm [43], [42]. In each outer step of the RTR method, the second-order model is minimized with a classic matrix-free CG algorithm. This results in a number of inner iterations to solve (6.1) up to a certain tolerance while still guaranteeing super-linear convergence of RTR. In addition, tCG employs two extra stopping criteria: the algorithm terminates if CG would use a search direction of negative or zero curvature, or if the new iterate would violate the TR bound. See [1, Alg. 2] for specific details.

Like classic CG, tCG lends itself excellent for preconditioning since a well chosen preconditioner will have a good influence on the conditioning of each TR subproblem. The effect is that the number of inner iterations will be drastically lowered. It is however not directly obvious how we can define a matrix-free preconditioner that is symmetric and positive definite in all points x . In this section, we will derive such a preconditioner.

TABLE 7.1

Effect of preconditioning: dependence on n for problems with a fixed rank $k = 8$.

precond.	n	15^2	20^2	25^2	30^2	35^2	40^2	45^2
none	n_{outer}	56	60	65	57	77	72	60
	$\sum n_{\text{inner}}$	314	370	410	498	539	536	695
	$\max n_{\text{inner}}$	43	47	69	89	81	98	126
$P_x \mathcal{L} P_x$	n_{outer}	58	69	68	65	68	68	69
	$\sum n_{\text{inner}}$	67	73	73	68	74	73	83
	$\max n_{\text{inner}}$	4	3	3	3	4	3	4

TABLE 7.2

Effect of preconditioning: dependence on k for problems with a fixed size $n = 30^2$.

precond.	k	1	2	3	4	5	6	7	8
none	n_{outer}	10	19	21	33	44	50	53	57
	$\sum n_{\text{inner}}$	193	273	256	283	297	427	433	498
	$\max n_{\text{inner}}$	58	79	76	76	83	94	80	89
$P_x \mathcal{L} P_x$	n_{outer}	7	14	34	48	54	60	60	65
	$\sum n_{\text{inner}}$	7	16	34	50	57	63	70	68
	$\max n_{\text{inner}}$	2	2	2	3	3	3	3	3

7.1. Projected Euclidean Hessian. From §6.1 we know that the Riemannian Hessian of $f(x)$,

$$\mathcal{H}_x := P_x(A \otimes M + M \otimes A)P_x - P_x^p(x^\dagger \otimes R + R \otimes x^\dagger)P_x^p,$$

consists of two parts, namely a projection of the Euclidean Hessian $\mathcal{L} = A \otimes M + M \otimes A$ and a term involving the residual. This projected Hessian $P_x \mathcal{L} P_x$ should make a good candidate for a preconditioner, since most of the poor conditioning of the TR subproblems can be attributed to \mathcal{L} , e.g. the PDE. Moreover, thanks to $\mathcal{L} \succ 0$, $P_x \mathcal{L} P_x$ is always symmetric and positive definite on $T_x \mathcal{M}$. In order to see how effective this preconditioner is, we have solved a series of (small) Lyapunov equations resulting from a discretized Laplace equation on a square.

First, we check the dependence on the size n of the system. The condition number of \mathcal{H}_x (and \mathcal{L}) will grow with n so we can expect more tCG iterations as the subproblems become larger. In Table 7.1 we see the number of outer RTR iterations and the total number of inner tCG iterations to solve for a rank $k = 8$ approximation. We have included the maximum number of CG iterations as well since the last subproblems need to be solved up to high accuracy. It is clearly visible that the preconditioner helps to reduce the number inner iterations to a number almost independent of the size of the system. The preconditioner does especially a good job for the last subproblems which can be solved in less than 5 iterations.

On the other hand, we can expect that the efficiency of the preconditioner worsens if the rank k grows because we did not take the second part of the Riemannian Hessian into account. This dependence on k is visible in Table 7.2 where we kept the size fixed to $n = 30^2$. Even though we see an increase in the total number of inner iterations with growing rank, there is still a significant reduction thanks to preconditioning.

7.2. Applying the preconditioner. It is obvious from the tables that preconditioning with $P_x \mathcal{L} P_x$ greatly reduces the total number of inner iterations. However,

the preconditioner will only be effective if it can be computed sufficiently fast, ideally at a cost of $O(n)$. We will show that this is possible for $M = I$ and when $(A + \lambda I)x = b$ with $\lambda > 0$ can be solved for x in $O(n)$.

Applying the preconditioner in $x = VDV^\top$ means solving $\xi \in T_x\mathcal{M}$ such that

$$(P_x \mathcal{L} P_x)(\xi) = \eta, \quad \eta \in T_x\mathcal{M}. \quad (7.1)$$

First, we write (7.1) in matrix form using the projectors (5.6) and (5.7):

$$P_V(A\xi M + M\xi A)P_V + P_V^\perp(A\xi M + M\xi A)P_V + P_V(A\xi M + M\xi A)P_V^\perp = \eta,$$

which decomposes into

$$P_V(A\xi M + M\xi A)P_V = P_V\eta P_V \quad \text{and} \quad P_V^\perp(A\xi M + M\xi A)P_V = P_V^\perp\eta P_V. \quad (7.2)$$

From now on, let $M = I$. With the factorizations as explained in §6.3, we can take the following matrix representations for the tangent vectors of $x = VDV^\top$:

$$\xi = VS_\xi V^\top + Z_\xi V^\top + VZ_\xi^\top \quad \text{and} \quad \eta = VS_\eta V^\top + Z_\eta V^\top + VZ_\eta^\top.$$

System (7.2) can then be written as

$$V^\top AVS_\xi + S_\xi V^\top AV + V^\top AZ_\xi + Z_\xi^\top AV = S_\eta, \quad (7.3)$$

$$P_V^\perp(AVS_\xi + AZ_\xi + Z_\xi V^\top AV) = Z_\xi, \quad \text{s.t.} \quad V^\top Z_\xi = 0. \quad (7.4)$$

where $S_\xi \in \mathbb{S}_k^{\text{sym}}$ and $Z_\xi \in \mathbb{R}^{n \times k}$ are the unknown matrices. By taking the eigenvalue decomposition $V^\top AV = Q\Lambda Q^\top$, the previous system is equivalent to

$$\Lambda \tilde{S}_\xi + \tilde{S}_\xi \Lambda + \tilde{V}^\top A \tilde{Z}_\xi + \tilde{Z}_\xi^\top A \tilde{V} = \tilde{S}_\eta, \quad (7.5)$$

$$P_{\tilde{V}}^\perp(A\tilde{V}\tilde{S}_\xi + A\tilde{Z}_\xi + \tilde{Z}_\xi^\top A) = \tilde{Z}_\xi, \quad \text{s.t.} \quad \tilde{V}^\top \tilde{Z}_\xi = 0. \quad (7.6)$$

where $\tilde{S}_\xi = Q^\top S_\xi Q \in \mathbb{S}_k^{\text{sym}}$ and $\tilde{Z}_\xi = Z_\xi Q \in \mathbb{R}^{n \times k}$ are transformed unknown matrices and $\tilde{V} = VQ$.

We will now eliminate \tilde{Z}_ξ from (7.6) and substitute it into (7.5). Since $\Lambda = \text{diag}(\lambda_i)$, we can solve in (7.6) for each column $\tilde{Z}_\xi(:, i)$ independently (we use the notation $(:, i)$ to denote the i -th column):

$$P_{\tilde{V}}^\perp(A + \lambda_i I)\tilde{Z}_\xi(:, i) = \tilde{Z}_\eta(:, i) - P_{\tilde{V}}^\perp A \tilde{V} \tilde{S}_\xi(:, i), \quad \text{s.t.} \quad \tilde{V}^\top \tilde{Z}_\xi(:, i) = 0. \quad (7.7)$$

By writing out the projector $P_{\tilde{V}}^\perp$, it is straightforward to see that this system is equivalent to the following saddle-point problem

$$\begin{bmatrix} A + \lambda_i I & \tilde{V} \\ \tilde{V}^\top & 0 \end{bmatrix} \begin{bmatrix} \tilde{Z}_\xi(:, i) \\ n \end{bmatrix} = \begin{bmatrix} \tilde{Z}_\eta(:, i) - P_{\tilde{V}}^\perp A \tilde{V} \tilde{S}_\xi(:, i) \\ 0 \end{bmatrix} \quad (7.8)$$

with $n \in \mathbb{R}^k$. This saddle-point problem can be efficiently solved by exploiting the sparsity of A but we will postpone the details to §7.3. For now, we can formally write (7.7) as

$$\tilde{Z}_\xi(:, i) = \mathcal{T}_i^{-1}(\tilde{Z}_\eta(:, i)) - \mathcal{T}_i^{-1}(P_{\tilde{V}}^\perp A \tilde{V}) \tilde{S}_\xi(:, i), \quad (7.9)$$

where $\mathcal{T}_i^{-1}(B)$ indicates solving the i -th saddle-point problem, corresponding to (7.8), with right-hand side B . Now plugging (7.9) into (7.5), we obtain

$$\Lambda \tilde{S}_\xi + \tilde{S}_\xi \Lambda + [v_1 - w_1 \quad \cdots \quad v_k - w_k] + [v_1 - w_1 \quad \cdots \quad v_k - w_k]^\top = \tilde{S}_\eta$$

with $v_i = \tilde{V}^\top AT_i^{-1}(\tilde{Z}_\eta(:, i))$ and $w_i = \tilde{V}^\top AT_i^{-1}(P_V^\perp A \tilde{V}) \tilde{S}_\xi(:, i)$. Let $K_i = \lambda_i I_k - \tilde{V}^\top AT_i^{-1}(P_V^\perp A \tilde{V})$, then we can isolate \tilde{S}_ξ as

$$\begin{bmatrix} K_1 \tilde{S}_\xi(:, 1) & \cdots & K_k \tilde{S}_\xi(:, k) \end{bmatrix} + \begin{bmatrix} \tilde{S}_\xi(1, :) K_1^\top \\ \vdots \\ \tilde{S}_\xi(k, :) K_k^\top \end{bmatrix} = R, \quad (7.10)$$

with known right-hand side matrix $R = \tilde{S}_\eta - [v_1 \quad \cdots \quad v_k] - [v_1 \quad \cdots \quad v_k]^\top$. Equation (7.10) is a linear equation in \tilde{S}_ξ with a special block structure. By vectorizing as in §3, it is straightforward to see that the first part of (7.10) satisfies

$$\text{vec} \begin{bmatrix} K_1 \tilde{S}_\xi(:, 1) & \cdots & K_k \tilde{S}_\xi(:, k) \end{bmatrix} = \begin{bmatrix} K_1 & & \\ & \ddots & \\ & & K_k \end{bmatrix} \begin{bmatrix} \tilde{S}_\xi(:, 1) \\ \vdots \\ \tilde{S}_\xi(:, k) \end{bmatrix} = \mathcal{K} \text{vec}(\tilde{S}_\xi),$$

where \mathcal{K} denotes the k^2 -by- k^2 block-diagonal matrix $\text{diag}(K_i)$. For the second part, we can use $\text{vec}(X) = \Pi \text{vec}(X^\top)$ with Π the perfect-shuffle matrix to obtain

$$\text{vec} \begin{bmatrix} \tilde{S}_\xi(1, :) L_1^\top \\ \vdots \\ \tilde{S}_\xi(k, :) L_k^\top \end{bmatrix} = \Pi \text{vec} \begin{bmatrix} L_1 \tilde{S}_\xi^\top(:, 1) & \cdots & L_k \tilde{S}_\xi^\top(:, k) \end{bmatrix} = \Pi \mathcal{K} \text{vec}(\tilde{S}_\xi^\top).$$

Finally, the whole equation (7.10) can be written as a linear system of size k^2

$$\mathcal{K} \text{vec}(\tilde{S}_\xi) + \Pi \mathcal{K} \text{vec}(\tilde{S}_\xi^\top) = \text{vec}(R) \iff (\mathcal{K} + \Pi \mathcal{K} \Pi) \text{vec}(\tilde{S}_\xi) = \text{vec}(R). \quad (7.11)$$

After solving (7.11) for \tilde{S}_ξ we obtain \tilde{Z}_ξ from (7.9). Undoing the transformations by Q , we get $S_\xi = Q \tilde{S}_\xi Q^\top$ and $Z_\xi = \tilde{Z}_\xi Q^\top$, and thus ξ , such that (7.1) is satisfied.

In case $M \neq I$, we simply approximate M by I and use the previous techniques. Although this is a very crude approximation of M , the obtained preconditioner seems to work quite well in the numerical experiments. The reason is that for generalized Lyapunov equations M usually represents the Galerkin mass matrix of a FEM discretization. For quasi-uniform meshes with shape regular elements this mass matrix is essentially a scaled identity matrix, see [16, Ch. 1.6].

7.3. Cost. There are two dominating costs for applying the preconditioner, namely solving the saddle-point problems (7.8) and solving the linear system (7.11).

Regarding (7.8), there is vast amount of literature for solving large and sparse saddle-point problems of this kind, see [9] for an overview. The solution technique of the previous section requires solving $\mathcal{T}_i(X) = B$ for two different right-hand sides, or equivalently, $B = \begin{bmatrix} \tilde{Z}_\eta(:, i) & P_V^\perp A \tilde{V} \end{bmatrix} \in \mathbb{R}^{n \times k+1}$, see (7.9). In our case, k is rather

small so we can solve $\mathcal{T}_i(X) = B$ by eliminating the (negative) Schur complement $S_i = \tilde{V}^\top (A + \lambda_i I)^{-1} \tilde{V}$, see [9, Sec. 5]. This gives

$$\begin{aligned} N &= S_i^{-1}(\tilde{V}^\top (A + \lambda_i I)^{-1} B), \\ X &= (A + \lambda_i I)^{-1} B - (A + \lambda_i I)^{-1} \tilde{V} N. \end{aligned}$$

For each \mathcal{T}_i , applying S_i^{-1} means an $O(k^3)$ cost for the Cholesky factorization of the dense matrix S_i and for the forward and back substitution to solve for $N \in \mathbb{R}^{k \times k+1}$. In addition, we need to apply $(A + \lambda_i I)^{-1}$ to B and \tilde{V} . Assuming an optimal solver for the sparse s.p.d. matrix $A + \lambda_i I$, this implies a cost of $O(nk)$. In total, we get a cost of $O(nk^2) + O(k^4)$ to solve all k saddle-point problems. Usually $k \ll n$ and the $O(nk^2)$ cost dominates.

Since in every inner iteration of RTR, the iterate x stays fixed, the $n \times k$ matrices $(A + \lambda_i I)^{-1} \tilde{V}$ and $(A + \lambda_i I)^{-1} P_{\tilde{V}}^\perp A \tilde{V}$ remain the same and can be re-used. If one is willing to cache these results for all k shifts, there is a significant speedup possible. The downside is that the memory requirements grow from $O(nk)$ to $O(nk^2)$. For more details, see the next section on numerical results

Equation (7.11) is a linear and symmetric system of size k^2 . Solving the vectorized form by a dense factorization results in an $O(k^6)$ cost which is prohibitively large, even for small k . However, in practice the equation can be solved much faster with an iterative method like CG. In all problems, we have observed convergence in only $O(\log k)$ steps. Together with (7.10) as matrix-vector product with cost $O(k^3)$, this results in an empirical $O(k^3 \log k)$ cost for solving (7.11).

8. Numerical results. In this section we illustrate the performance of our Riemannian optimization approach with some numerical experiments. The computations were done with MATLAB R2008a on a Pentium Xeon 2.33 GHz with $\epsilon_{mach} \simeq 2 \cdot 10^{-16}$. The code is available at <http://www.cs.kuleuven.be/~bartvde>.

8.1. Exact solution. We would like to compare the local minimizer of Algorithm 1 with an exact global minimizer. To the best of our knowledge there is no method that computes such a global minimizer for the general problem. However, if we choose $A = M = I$ as Lyapunov equation, equation (1.1) reduces to $2X = C$ and the optimization problem is equivalent to the best rank k s.p.s.d. approximation of $\frac{1}{2}C$. If $C \succ 0$, the global minimizer exists for all ranks and it can be easily computed based on an eigenvalue decomposition of C .

As a performance test we minimized 100 random problems. Each problem consists of finding approximations of rank $k = 1, 2, \dots, 30$ for 30 random matrices, generated by MATLAB using `randsym(n, 10/n, 1/n, 1)` with `n=floor(logspace(1.8, 3, 30))`. This results in sparse matrices of size n , with density $10/n$ and a condition number of n . There was no specific reason for choosing this kind of random matrices besides that they have an exponential decay of the eigenvalues. This decay also occurs with solutions of Lyapunov equations for real problems, cfr. the low-rank property. The total number of Lyapunov equations matrices was 90 000. The reason for solving such a large amount of problems is to assess whether the non-convexity of the optimization problem poses a problem in practice.

In Fig. 8.1 we see for each rank and size the maximum and mean of the 100 relative errors. We used the RTR algorithm with and without a preconditioner but this did not have an influence on the error of the obtained approximations. Furthermore, in order to get the smallest error we did not use a convergence criterion on the norm of

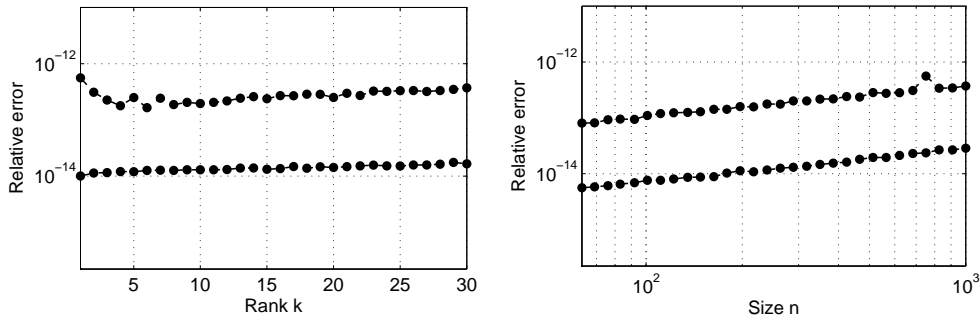


FIG. 8.1. Comparison with the exact solution. The mean and maximum of the relative error for 100 random problems solved by RTR as a function of rank (left) and of size (right).

the gradient, i.e. we let the algorithm run until it could make no further progress. All test problems were minimized with a relative error smaller than 10^{-12} and a majority performed significantly better. This indicates that the non-convexity of the problem does not pose practical problems. Comparing the left and the right frame, we see that the precision worsens with the size of the problem, while the rank seems to have slightly less influence on the error.

8.2. Quality of the low-rank solutions. In Fig. 8.2 we investigated the quality of the solutions from Algorithm 1 compared to the best rank- k approximations of the exact solution. The generalized Lyapunov equation was the simplified steel cooling benchmark from [8] of size $n = 1357$ with a rank one matrix C . Since RTR minimizes the error in the energy norm, we should expect that the best rank- k approximations always have a better accuracy measured in the Frobenius norm. This is verified in the left panel of Fig. 8.2. In addition we see that the difference stays rather small and behaves uniformly in the rank. In other words, the RTR approximations are nearly as good as the best rank- k approximations. Surprisingly, the errors of the residual of the RTR approximations are a little better than the best rank- k approximations, as seen in the right panel of Fig. 8.2.

Next, we performed the same comparison with the K-PIK algorithm from [40]. Each step of K-PIK appends a new column to the factor Y of the solution $x = YY^T$ and thus the rank will increase with every step. Although each step of the K-PIK algorithm is cheap in comparison with the RTR method, in Fig. 8.2 we can clearly see that these low-rank solutions are far from optimal. Once the solution is accurate enough, the factor Y can be compressed to one with much smaller numerical rank. For big problems, this may pose significant memory problems for K-PIK, whereas RTR needs much less memory.

8.3. Without mass matrix. We performed Algorithm 2 on a Lyapunov equation resulting from a heat equation on a square. The problem has the identity matrix as mass matrix, i.e. $M = I$, and the right-hand side is a rank one matrix C of all ones. The tolerance on the relative residual was chosen to be $\tau = 10^{-6}$ and Algorithm 1 was run with a convergence tolerance of 10^{-10} on the norm of the gradient. The preconditioner of §7 was used and the action of $(A + \lambda I)^{-1}$ was approximated by one V-cycle of a geometric multigrid with 2 pre and 2 post smoothing steps (red-black Gauss-Seidel). We performed two experiments: one with caching of the intermediate results and one without, as explained in §7.3.

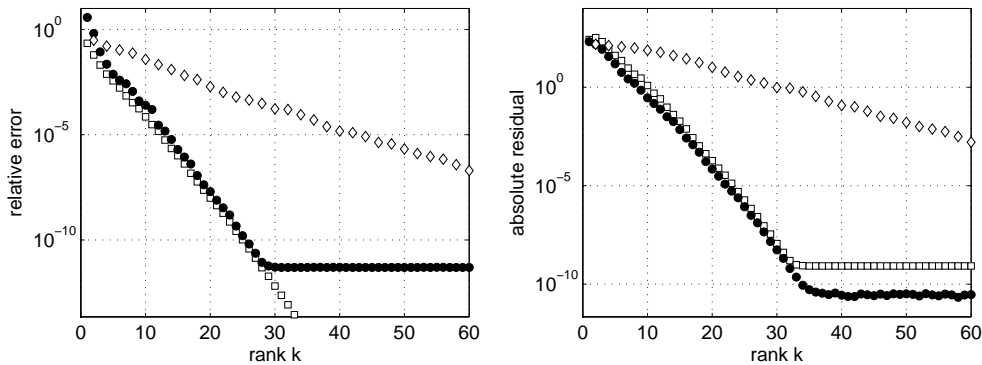


FIG. 8.2. The relative error and the absolute residual for the simplified steel cooling problem with $n = 1357$. The best rank- k approximations \square are compared to approximations computed with RTR \bullet and K-PIK \diamond .

The experimental results are visible in Tables 8.1 and 8.2. All problems were solved to convergence and the required relative residual was achieved. For both tables, we can make several observations that confirm the theory. First, the rank needed to obtain a certain fixed precision grows with the problem size as $O(\log n)$, see [19, Remark 1]. Since the RTR approximations are nearly as good as the best rank- k approximations, see §8.2, this is optimal. Second, the recursive nature of Algorithm 2 makes that the number of outer iterations will grow with the problem size since the optimal rank will grow as well.

The difference between the two tables is caching of temporary results for the preconditioner. Without caching (Table 8.1) the memory requirements stay very modest while wall time grows quickly. On the other hand, with caching enabled (Table 8.2) there is a significant speed-up but at the cost of significantly more memory.

8.4. With mass matrix. As a final experiment, we solved the simplified steel cooling problem with mass matrix for a series of discretizations. We used Algorithm 2 to obtain an approximation with a relative residual of $\tau = 10^{-6}$ and Algorithm 1 was run with a convergence tolerance of 10^{-7} on the norm of the gradient. The cached preconditioner of §7 with the approximation $M = I$ was used. To apply $(A + \lambda I)^{-1}$ we compute the sparse Cholesky factorization of $(A + \lambda I)$ with CHOLMOD [12] of MATLAB and store the result.

The results are visible in Table 8.3 and, as a whole, they are very similar to the results without mass matrix. However, the number of iterations (and time) is significantly higher than in §8.3 for systems with comparable size. This is due to the approximation of the mass matrix by an identity matrix in the preconditioner. In order to verify the efficacy of this preconditioner, we solved the same problem but now without a preconditioner. In Table 8.4, we observe a significant increase in the total number of iterations as expected. In addition, the time to solve the approximation is also higher. This confirms that despite the crude approximation of the mass matrix, the projected preconditioner indeed accelerates the optimization algorithm.

Acknowledgments. We thank Lars Grasedyck for helpful discussions on matrix equations. We gratefully acknowledge Pierre-Antoine Absil for introducing us to Riemannian optimization and his subsequent support.

Bart Vandereycken is a Research Assistant of the Research Foundation–Flanders

TABLE 8.1

Experimental results for the heat equation solved by Algorithm 2, preconditioned with one V-cycle of geometric multigrid, no caching

n	time (sec)	mem (Mb)	inner its / outer its	k	norm of gradient	relative residual
3 969	14	36	223 / 58	10	8.39e-11	2.28e-07
6 241	29	38	261 / 63	10	9.39e-11	6.59e-07
9 025	55	40	317 / 71	11	9.94e-11	2.81e-07
12 321	83	41	334 / 73	11	5.20e-11	5.64e-07
16 129	110	42	345 / 75	11	7.35e-11	9.44e-07
25 281	251	47	400 / 80	12	9.09e-11	5.31e-07
36 481	499	54	455 / 88	13	9.88e-11	2.37e-07
49 729	820	61	501 / 91	13	9.70e-11	4.11e-07
65 025	1 213	69	525 / 95	13	2.65e-11	7.32e-07
101 761	2 428	92	571 / 97	14	9.45e-11	3.56e-07
146 689	4 095	118	616 / 102	14	9.89e-11	6.25e-07
199 809	7 185	154	689 / 106	15	9.72e-11	2.60e-07
261 121	10 771	191	733 / 109	15	9.46e-11	4.23e-07
408 321	22 393	280	865 / 117	15	8.62e-11	7.38e-07
588 289	40 694	406	949 / 123	16	8.86e-11	4.68e-07
801 025	61 634	541	991 / 123	16	7.58e-11	6.28e-07

(FWO). Both authors have been partially supported by the Research Council K.U. Leuven, CoE EF/05/006 Optimization in Engineering (OPTEC) and present results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors.

REFERENCES

- [1] P.-A. ABSIL, C.G. BAKER, AND K.A. GALLIVAN, *Trust-region methods on Riemannian manifolds*, *Found. Comput. Math.*, 7 (2007), pp. 303–330.
- [2] P.-A. ABSIL, M. ISHTEVA, L. DE LATHAUWER, AND S. VAN HUFFEL, *A geometric newton method for Oja's vector field*, ucl-inma-2008.013, UCL, 2008.
- [3] P.-A. ABSIL, R. MAHONY, AND R. SEPULCHRE, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, 2008.
- [4] A. C. ANTOUNAS, *Approximation of Large-Scale Dynamical Systems*, *Adv. Des. Control*, SIAM, Philadelphia, 2005.
- [5] A. C. ANTOUNAS, D. C. SORESENSEN, AND Y. ZHOU, *On the decay rate of Hankel singular values and related issues*, *Systems & Control Letters*, 46 (2002), pp. 323–342.
- [6] R. H. BARTELS AND G. W. STEWART, *Solution of the matrix equation $AX + XB = C$* , *Communications of the ACM*, 15 (1972), pp. 820–826.
- [7] P. BENNER, *Control Theory*, *Handbook of Linear Algebra*, Chapman & Hall/CRC, 2006.
- [8] P. BENNER AND J. SAAK, *Efficient numerical solution of the LQR-problem for the heat equation*, *Proc. Appl. Math. Mech.*, 4 (2004), pp. 648–649.
- [9] M. BENZI, G. H. GOLUB, AND J. LIESEN, *Acta Numerica*, Cambridge University Press, 2005, ch. Numerical Solution of Saddle Point Problems, pp. 1–137.
- [10] W. M. BOOTHBY, *An Introduction to Differentiable Manifolds and Riemannian Geometry*, Academic Press, 2nd ed., 1986.
- [11] S. BURER AND R. D.C. MONTEIRO, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, *Math. Programming*, 95 (2003), pp. 329–357.
- [12] Y. CHEN, T. A. DAVIS, W. W. HAGER, AND S. RAJAMANICKAM, *Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate*, *ACM Trans. Math.*

TABLE 8.2

Experimental results for the heat equation solved by Algorithm 2, preconditioned with one V-cycle of geometric multigrid, with caching

n	time (sec)	mem (Mb)	inner its / outer its	k	norm of gradient	relative residual
3 969	5	42	223 / 58	10	8.39e-11	2.28e-07
6 241	9	48	261 / 63	10	9.39e-11	6.60e-07
9 025	17	57	317 / 71	11	9.94e-11	2.81e-07
12 321	26	64	334 / 73	11	5.20e-11	5.64e-07
16 129	35	72	345 / 75	11	7.35e-11	9.44e-07
25 281	78	104	400 / 80	12	9.09e-11	5.31e-07
36 481	153	150	455 / 88	13	9.88e-11	2.37e-07
49 729	250	193	501 / 91	13	9.70e-11	4.12e-07
65 025	376	241	525 / 95	13	2.65e-11	7.32e-07
101 761	748	404	571 / 97	14	9.45e-11	3.56e-07
146 689	1 240	567	616 / 102	14	9.89e-11	6.25e-07
199 809	2 128	857	689 / 106	15	9.72e-11	2.60e-07
261 121	3 766	1 109	733 / 109	15	9.46e-11	4.24e-07
408 321			out of memory: more than 2 Gb			
588 289			out of memory: more than 2 Gb			
801 025			out of memory: more than 2 Gb			

TABLE 8.3

Experimental results for the simplified steel cooling benchmark solved by the nested solver, cached preconditioner with CHOLMOD.

n	time (sec)	inner its / outer its	k	norm of gradient	relative residual
1 357	124	2 782 / 158	19	5.94e-08	9.62e-07
5 177	1 123	4 125 / 195	22	9.04e-09	5.25e-07
20 209	7 851	5 141 / 236	24	4.73e-08	6.19e-07
79 841	30 654	6 422 / 281	26	4.73e-08	7.01e-07

Softw., 35 (2008), pp. 1–14.

- [13] K.-W.E. CHU, *The solution of the matrix equation $AXB - CXD = Y$ and $(YA - DZ, YC - BZ) = (E, F)$* , Linear Algebra Appl., 93 (1987), pp. 93–105.
- [14] A. R. CONN, N. I. M. GOULD, AND PH. L. TOINT, *Trust-region methods*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, and Mathematical Programming Society (MPS), Philadelphia, PA, 2000.
- [15] A. EDELMAN, T. A. ARIAS, AND S. T. SMITH, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 303–353.
- [16] H. ELMAN, D. SILVESTER, AND A. WATHEN, *Finite Element and Fast Iterative Solvers*, Oxford University Press, 2005.
- [17] GENE H. GOLUB AND CHARLES F VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, 1989.
- [18] N. I. M. GOULD, S. LUCIDI, M. ROMA, AND PH. L. TOINT, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim., 9 (1999), pp. 504–525.
- [19] L. GRASEDYCK, *Existence of a low rank or \mathcal{H} -matrix approximant to the solution of a Sylvester equation*, Numer. Linear Algebra Appl., 11 (2004), pp. 371–389.
- [20] L. GRASEDYCK AND W. HACKBUSCH, *A multigrid method to solve large scale Sylvester equations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 870–894.
- [21] S. GUGERCIN, D.C. SORENSEN, AND A.C. ANTOLAS, *A modified low-rank Smith method for large-scale Lyapunov equations*, Numer. Algorithms, 32 (2003), pp. 27–55.
- [22] U. HELMKE AND J. MOORE, *Optimization and Dynamical Systems*, Springer-Verlag, London,

TABLE 8.4

Experimental results for the simplified steel cooling benchmark solved by the nested solver, no preconditioner.

n	time (sec)	inner its / outer its	k	norm of gradient	relative residual
1 357	161	22 295 / 161	19	5.94e-08	9.62e-07
5 177	3 041	61 081 / 202	22	6.95e-09	5.25e-07
20 209	21 339	154 294 / 258	24	8.61e-10	6.19e-07
79 841		out of time: longer than 5 days			

UK, 1994.

- [23] U. HELMKE AND M. A. SHAYMAN, *Critical points of matrix least squares distance functions*, Linear Algebra Appl., 215 (1995), pp. 1–19.
- [24] R. A. HORN AND C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.
- [25] I. JAIMOUKHA AND E. KASENALLY, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251.
- [26] K. JBILOU AND A. J. RIQUET, *Projection methods for large Lyapunov matrix equations*, Linear Algebra Appl., 415 (2006), pp. 344–358.
- [27] M. JOURNÉE, F. BACH, P.-A. ABSIL, AND R. SEPULCHRE, *Low-rank optimization for semidefinite convex problems*, tech. report, Liege, 2008.
- [28] P. LANCASTER, *Explicit solutions of linear matrix equations*, SIAM Rev., 12 (1970), pp. 544–566.
- [29] P. LANCASTER AND M. TISMENETSKY, *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.
- [30] A. S. LEWIS AND J. MALICK, *Alternating projections on manifolds*, Math. Oper. Res., 33 (2008), pp. 216–234.
- [31] J.-R. LI AND J. WHITE, *Low-rank solution of Lyapunov equations*, SIAM Rev., 46 (2004), pp. 693–713. SIGEST selection.
- [32] S. A. MILLER AND J. MALICK, *Newton methods for nonsmooth convex minimization: connections among U -Lagrangian, Riemannian Newton and SQP methods*, Math. Programming, 104 (2005), pp. 609–633.
- [33] B. MOORE, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Trans. Automat. Control, 26 (1981), pp. 17–32.
- [34] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.
- [35] T. PENZL, *Numerical solution of generalized Lyapunov equations*, Adv. Comput. Math., 8 (1998), pp. 33–48.
- [36] ———, *A cyclic low-rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 4 (2000), pp. 1401–1418.
- [37] ———, *Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case*, Systems Control Lett., 40 (2000), pp. 139–144.
- [38] Y. SAAD, *Numerical solution of large Lyapunov equations*, in Signal Processing, Scattering, Operator Theory, and Numerical Methods, M. A. Kaashoek, J. H. Van Schuppen, and A. C. M. Ran, eds., Birkhäuser, Boston, MA, 1990, pp. 503–511.
- [39] N. SCHEERLINCK, P. VERBOVEN, J. D. STIGTER, J. DE BAERDEMAEKER, J. F. VAN IMPE, AND B. M. NICOLAI, *A variance propagation algorithm for stochastic heat and mass transfer problems in food processes*, Internat. J. Numer. Methods Engrg., 51 (2001), pp. 961–983.
- [40] V. SIMONCINI, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.
- [41] D. C. SORENSEN AND Y. ZHOU, *Bounds on eigenvalue decay rates and sensitivity of solutions to Lyapunov equations*, Tech. Report 02-07, Computational and Applied Mathematics, Rice University, 2002.
- [42] T. STEihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.
- [43] PH. L. TOINT, *Sparse Matrices and Their Uses*, Academic Press, London, New York, 1981, ch. Towards an efficient sparsity exploiting Newton method for minimization, pp. 57–88.
- [44] C. F. VAN LOAN, *The ubiquitous Kronecker product*, J. Comput. Appl. Math., 123 (2000), pp. 85–100.