Riemannian multilevel optimization for solving large-scale Lyapunov equations

Bart Vandereycken

We present a new multilevel algorithm to solve large-scale Lyapunov equations. The method consists of minimizing a certain objective function on the Riemannian manifold of symmetric and positive semidefinite matrices of fixed rank. By employing the principle of so-called retraction-based Riemannian optimization, we can extend the existing Riemannian methods to a multilevel optimization algorithm. Numerical experiments indicate robust, mesh-independent convergence of the newly proposed method.

1 Lyapunov equations and low-rank approximations

Let $A, C \in \mathbf{R}^{n\times n}$ be given matrices. Consider the *Lyapunov equation*,

$$AX + XA^T = C, \tag{1}$$

which is a linear matrix equation in $X \in \mathbf{R}^{n\times n}$. Throughout this paper, we assume that A and C are symmetric matrices and, furthermore, that A is positive definite. It is well known, that under these assumptions, there is a unique solution X, which is symmetric and positive semidefinite. The Lyapunov equation is of significant importance in control theory and model reduction of dynamical systems; see, e.g., Benner *et al.* (2005); Antoulas (2005).

In general, the solution of (1) is a dense matrix. Applied to large-scale systems, like discretized PDEs, solving these Lyapunov equations is a formidable task. Indeed, when the PDE is discretized with n degrees of freedom, direct methods require $O(n^3)$ work. Even optimal methods, like multigrid, that have an $O(n^2)$ complexity are still out of reach when $n \gg 10^4$.

A popular approach to circumvent this squaring of the complexity is to exploit the fact that X can often be well approximated by a low-rank matrix. For example, if C is of low rank and A has condition number κ, it follows from Grasedyck (2004, Remark 1) that one can approximate X with a relative accuracy of ϵ using a matrix of rank $p = O(\log(\kappa)\log(1/\epsilon))$.

A number of solvers have been developed that compute such a low-rank approximation directly; see, e.g., Penzl (1999); Li & White (2004); Simoncini (2007); Grasedyck & Hackbusch (2007). In this paper, we will devise a new method based on combining two techniques: tensor-product multigrid and Riemannian optimization.

2 Tensor-product multigrid

The Lyapunov equation associated with a PDE has a multilevel structure. It is therefore natural to employ a multigrid algorithm for solving (1). By regarding a matrix in $\mathbf{R}^{n\times n}$ as an element of the tensor-product space $\mathbf{R}^n \otimes \mathbf{R}^n$, one can construct a multigrid algorithm by taking tensor products of standard multigrid components. The result is a so-called *tensor-product multigrid*, devised in Rosen & Wang (1995); Penzl (1997).

We refer to Rosen & Wang (1995); Penzl (1997) for a detailed description of this multigrid algorithm. Here, we only describe the intergrid transfer operators between the coarse and fine grid. Let $p : \mathbf{R}^N \to \mathbf{R}^n$ and $r : \mathbf{R}^n \to \mathbf{R}^N$ be the prolongation and restriction operator for a standard multigrid iteration to solve $Ax = b$. Then these transfer operators for the tensor-product multigrid are given by

$$I_h^H : \mathbf{R}^n \otimes \mathbf{R}^n \simeq \mathbf{R}^{n\times n} \to \mathbf{R}^{N\times N},\ X \mapsto rXr^T, \tag{2}$$

$$I_H^h : \mathbf{R}^N \otimes \mathbf{R}^N \simeq \mathbf{R}^{N\times N} \to \mathbf{R}^{n\times n},\ X \mapsto pXp^T. \tag{3}$$

In order to obtain low-rank approximations of X in (1), however, this tensor-product multigrid is not directly applicable. In Grasedyck & Hackbusch (2007), the authors propose a modification that enables to directly compute such a low-rank approximation. Essentially, it performs a tensor-product multigrid on a factored matrix, while after every step, computing a truncated eigenvalue decomposition of the iterate to make it stay of low rank. The downside of this algorithm is that the smoothers need to be computed in

this low-rank format also. Apart from simple point smoothers, like Richardson and weighted Jacobi, it not clear how to do this in a low-rank format.

Our modification of tensor-product multigrid to a low-rank solver will allow to employ more powerful smoothers, like block Jacobi smoothers. Combined with a robust optimization method, it should also be less sensitive to the particular ranks chosen at each level of the multigrid iteration.

Remark 2.1. For simplicity and notational convenience, all subsequent derivations will focus on a two-grid cycle; true multigrid algorithms (V- and W-cycles) will be straightforward to derive from this two-grid cycle. Everything that relates to the fine grid is denoted by subscript $\cdot_h$ and, likewise, $\cdot_H$ for the coarse grid.

3 Riemannian optimization

In this section, we recall the ideas of Vandereycken & Vandewalle (2010), which partly form the basis of our newly proposed method. Computing a low-rank approximation to X in (1) can also be seen as an optimization problem on the set

$$\mathbf{S}_+^{n,p} = \{x \in \mathbf{R}^{n\times n} \mid x = x^T,\ x \succeq 0,\ \mathrm{rank}(x) = p\}. \tag{4}$$

In the following, the rank p is considered fixed. In Vandereycken & Vandewalle (2010), it was demonstrated that local minimizers of

$$f : \mathbf{S}_+^{n,p} \to \mathbf{R},\ x \mapsto \mathrm{tr}(xAx) - \mathrm{tr}(xC) \tag{5}$$

provide meaningful low-rank approximations for X. Moreover, Riemannian optimization was employed to solve

$$\min f(x) \text{ subject to } x \in \mathbf{S}_+^{n,p},$$

based on the observation that $\mathbf{S}_+^{n,p}$ constitutes a smooth, Riemannian manifold.

We briefly recall the concepts of this Riemannian method that are necessary for the next sections and refer to the author's PhD thesis (Vandereycken, 2010) for a proper introduction into Riemannian optimization on $\mathbf{S}_+^{n,p}$. Furthermore, we emphasize that all the necessary expressions can be efficiently computed in case of these low-rank matrices.

The algorithm in Vandereycken & Vandewalle (2010)—as well as the one proposed in this paper—belong to the recent retraction-based framework for Riemannian optimization; see, e.g., Absil *et al.* (2008). The principle behind these retraction-based methods can be summarized by aid of Figure 1.

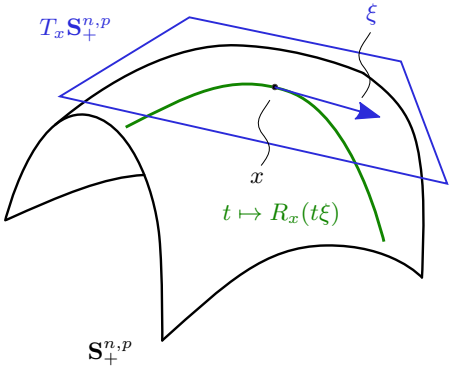

Figure 1: Retraction-based optimization.

Suppose the numerical method arrived at the iterate x on the manifold $\mathbf{S}_+^{n,p}$. At this iterate, we can construct the *tangent space* $T_x\mathbf{S}_+^{n,p}$ to the manifold. The update to a better iterate proceeds in the direction of a vector ξ that lies in this tangent space. Once this update has been found, one orthogonally projects $x + \xi$ to the manifold $\mathbf{S}_+^{n,p}$. This operation is encoded by

$$R_x : T_x\mathbf{S}_+^{n,p} \to \mathbf{S}_+^{n,p},\ \xi \mapsto \arg\min_{z\in\mathbf{S}_+^{n,p}} \|x + \xi - z\|_{\mathrm{F}},$$

called the *retraction mapping*. In our case, this orthogonal projection can be efficiently computed by a truncated eigenvalue decomposition[1]. After one such step, the iteration repeats itself by taking a different step in the tangent space of the new iterate. The optimization procedure is robustified by a Trust-Region

[1] Since $\mathbf{S}_+^{n,p}$ is non-convex, this projection does not need to be unique nor always exist. Since we use a Trust-Region framework and R_x is locally well-defined, this is not a problem for the algorithm, in theory and in practice.

framework and the global convergence of this method is proven in Absil *et al.* (2007) for general manifolds and objective functions.

Let

$$g^{\mathrm{E}}(Z_1, Z_2) := \mathrm{tr}(Z_1^T Z_2) \tag{6}$$

denote the Euclidean metric on $\mathbf{R}^{n\times n}$. Since $\mathbf{S}_+^{n,p}$ is an embedded submanifold of $\mathbf{R}^{n\times n}$, the metric g^{E} restricted to tangent vectors is also a *Riemannian metric* on $\mathbf{S}_+^{n,p}$. The update vector ξ of Figure 1 is found by solving the so-called Trust-Region subproblem

$$\min_{\xi\in T_x\mathbf{S}_+^{n,p}, \|\xi\|\leq\Delta} m_x(\xi) := f(x) + g^{\mathrm{E}}(\mathrm{grad}\, f(x), \xi) + \frac{1}{2} g^{\mathrm{E}}(\mathrm{Hess}\, f(x)[\xi], \xi),$$

which is the minimization of a second-order model $m_x : T_x\mathbf{S}_+^{n,p} \to \mathbf{R}$ associated with x, subject to a trust-region bound on $T_x\mathbf{S}_+^{n,p}$. Let $R := Ax + xA - C$ be the residual at x. Then this model is defined using the *Riemannian gradient*,

$$\mathrm{grad}\, f(x) := \mathrm{P}_x^{\mathrm{t}}(R) \in T_x\mathbf{S}_+^{n,p},$$

and the *Riemannian Hessian*, a symmetric, linear operator that satisfies

$$\mathrm{Hess}\, f(x) : T_x\mathbf{S}_+^{n,p} \to T_x\mathbf{S}_+^{n,p},\ \xi \mapsto \mathrm{P}_x^{\mathrm{t}}(A\xi + \xi A) + \mathrm{P}_x^{\mathrm{t,p}}(R\,\mathrm{P}_x^{\mathrm{t,p}}(\xi)x^\dagger + x^\dagger \mathrm{P}_x^{\mathrm{t,p}}(\xi)R).$$

Here, $\mathrm{P}_x^{\mathrm{t}} : \mathbf{R}^{n\times n} \to T_x\mathbf{S}_+^{n,p}$ is the orthogonal projection onto $T_x\mathbf{S}_+^{n,p}$, and $\mathrm{P}_x^{\mathrm{t,p}}$ the orthogonal projection onto a specific subspace of $T_x\mathbf{S}_+^{n,p}$.

The actual minimization of the TR subproblems is done by a matrix-free, preconditioned truncated CG (tCG) method. The preconditioner is the projected Euclidean part of the Riemannian Hessian, i.e.,

$$\mathrm{Prec}\ f(x) : T_x\mathbf{S}_+^{n,p} \to T_x\mathbf{S}_+^{n,p},\ \xi \mapsto \mathrm{P}_x^{\mathrm{t}}(A\xi + \xi A). \tag{7}$$

The dominating cost of applying $[\mathrm{Prec}\ f(x)]^{-1}$ to a tangent vector is the solution of p^2 shifted linear systems $A + \lambda I$ with $\lambda > 0$.

4 Riemannian multilevel optimization

This section contains the main contribution of the paper: a generalization of multilevel optimization to Riemannian manifolds. Although this generalization is applicable to arbitrary manifolds, we restrict our presentation to the particular case $\mathbf{S}_+^{n,p}$, defined in (4). This will allow us to compute low-rank solutions for Lyapunov equations based on minimizing f over $\mathbf{S}_+^{n,p}$ by a multilevel algorithm. We explain this generalization in three steps.

4.1 Nonlinear multigrid in Euclidean space

Anticipating the nonlinear nature of manifold $\mathbf{S}_+^{n,p}$, we switch to the framework of nonlinear multigrid; and more specifically, the Full Approximation Scheme (FAS); see, e.g., Trottenberg *et al.* (2000, Section 5.3).

In case of nonlinear multigrid, the discretized equations that need to be solved can be nonlinear. On the fine grid, this nonlinear equation is given by

$$A_h(x_h) = f_h. \tag{8}$$

On the coarse grid, we have the usual coarse discretization of this operator, denoted by A_H. Algorithm 1 lists the standard[2] nonlinear FAS two-grid cycle to solve (8). As is well known, the difference with the standard linear two-grid cycle is that the smoothed approximation $\overline{x}_h$ is also transferred to the coarse grid. In addition, the right-hand side of the coarse-grid equation is modified to

$$A_H(w_H) = A_H(\overline{x}_H + e_H) = r_H + A_H(\overline{x}_H).$$

In Algorithm 1, `smooth` stands for a nonlinear relaxation method which has suitable error smoothing properties. Much in the same way of linear multigrid, this smoother can be based on weighted versions of Jacobi and Gauss–Seidel, and their block versions. For simplicity, we only recall the Jacobi iteration here. In the later sections, we will specify the block generalization applied to the Lyapunov equation.

Suppose we write the fine-grid equation (8) component-wise,

$$A_{h,j}(x_{h,1}, x_{h,2}, \ldots, x_{h,n}) = f_{h,j}, \quad j = 1, \ldots, n,$$

as n non-linear equations $A_{h,j}$ in n unknowns $x_{h,j}$. Then, at iterate $x_h^{(i)}$, the nonlinear Jacobi iteration consists of solving the equations

$$A_{h,j}(x_{h,1}^{(i)}, \ldots, x_{h,j-1}^{(i)}, \widetilde{x}_{h,j}, x_{h,j+1}^{(i)}, \ldots, x_{h,n}^{(i)}) = f_{h,j}, \quad j = 1, \ldots, n. \tag{9}$$

[2] For simplicity we assume that the restriction operator for $\overline{x}_h$ is the same as for r_h.

for $\widetilde{x}_{h,j}$. After weighting by ω_h,

$$\overline{x}_{h,j} = (1-\omega_h)\, x_{h,j}^{(i)} + \omega_h\, \widetilde{x}_{h,j}, \quad j = 1, \dots, n, \tag{10}$$

one obtains the smoothed iterates $\overline{x}_{h,j}$.

Hence, we require solving n scalar equations of the form (9) that are possibly nonlinear. Most of the time, it is not necessary to solve these equations to full accuracy, but only a few Newton iterations suffice. One Newton iteration applied to (9) results in

$$\widetilde{x}_{h,j} = x_{h,j}^{(i)} - (A_{h,j}(x_h^{(i)}) - f_{h,j})/D_{j,j}, \quad j = 1, \dots, n,$$

where $D_{i,j}(x) := \partial A_{h,i}(x)/\partial x_j$ is the (i,j)-th element of the Jacobian matrix of A at x. Plugging this into (10), one obtains the Jacobi–Newton iteration (Trottenberg *et al.*, 2000, Sect. 5.3.2)

$$\overline{x}_{h,j} = x_{h,j}^{(i)} - \omega_h\, (A_{h,j}(x_h^{(i)}) - f_{h,j})/D_{j,j}, \quad j = 1, \dots, n. \tag{11}$$

Algorithm 1 Nonlinear FAS two-grid cycle

1: **for** $i = 1, 2, \dots$ **do**
2: $\quad \overline{x}_h = \mathtt{smooth}^{\nu_1}(x_h^{(i)}, A_h, f_h)$
3: $\quad r_h = f_h - A_h(\overline{x}_h)$
4: $\quad r_H = I_h^H r_h$
5: $\quad \overline{x}_H = I_h^H \overline{x}_h$
6: $\quad$ solve $A_H(\overline{x}_H + e_H) = r_H + A_H(\overline{x}_H)$ for e_H
7: $\quad e_h = I_H^h e_H$
8: $\quad \hat{x}_h = \overline{x}_h + e_h$
9: $\quad x_h^{(i+1)} = \mathtt{smooth}^{\nu_2}(\hat{x}_h, A_h, f_h)$
10: **end for**

This FAS two-grid cycle is visualized in Figure 2. We denote the right-hand sides by b_h and b_H; they satisfy

$$b_h \equiv f_h, \qquad b_H \equiv r_H + A_H(\overline{x}_H).$$

Observe that Figure 2 clearly emphasizes the difference between iterates (points in Euclidean space) and updates (vectors, denoted by full arrows). It is obvious that the coarse grid corrections e_h and e_H are the updates of the iterates $\overline{x}_h$ and $\overline{x}_H$, respectively. In addition, the action of a smoother also constitutes one or several update vectors. In case the smoother is the Jacobi–Newton iteration of (11), we get

$$\overline{x}_h = x_h^{(i)} + \omega_h\, p_h, \quad p_h = \operatorname{diag}\left(\frac{\partial A_h(x_h^{(i)})}{\partial x}\right)^{-1} (f_h - A_h(x_h^{(i)})),$$

$$x_h^{(i+1)} = \hat{x}_h + \omega_h\, \hat{p}_h, \quad \hat{p}_h = \operatorname{diag}\left(\frac{\partial A_h(\hat{x}_h)}{\partial x}\right)^{-1} (f_h - A_h(\hat{x}_h)).$$

For more general block-smoothers, the update vectors p_h and $\hat{p}_h$ will be obtained as the solutions of suitable defined, inexpensive linear systems.

Before generalizing this cycle to a manifold, we need to turn it into an optimization algorithm first.

4.2 Multilevel optimization in Euclidean space

In the previous section, we introduced (5) as the objective function f, which after minimization results in a low-rank approximation for the solution of a Lyapunov equation. If we ignore for now that $\mathbf{S}_+^{n,p}$ is not Euclidean, the previous FAS scheme can be easily generalized to a multilevel algorithm for minimizing f.

In Section 2, we assumed that the Lyapunov equation is associated to a discretization of a PDE. Hence, the matrices A, C and the function f have a natural multilevel structure. In our two-grid setting, this is made explicit with the notation

$$f_h(x) := \operatorname{tr}(xA_hx) - \operatorname{tr}(xC_h), \quad f_H(x) := \operatorname{tr}(xA_Hx) - \operatorname{tr}(xC_H), \tag{12}$$

where we used the fine-grid matrices A_h, C_h; and likewise for the coarse-grid matrices A_H, C_H.

Following the principle behind FAS, there will be a modification to the coarse-grid objective function $f_H(x_h)$. Let $g^{\mathrm{E}}(\cdot,\cdot)$ denote the Euclidean inner product of (6). Then it turns out that, by minimizing the objective function

$$\psi_H(x_H) := f_H(x_H) - g^{\mathrm{E}}(x_H, \operatorname{grad} f_H(\overline{x}_H) - I_h^H g_h), \quad g_h := \operatorname{grad} f_h(\overline{x}_h), \tag{13}$$

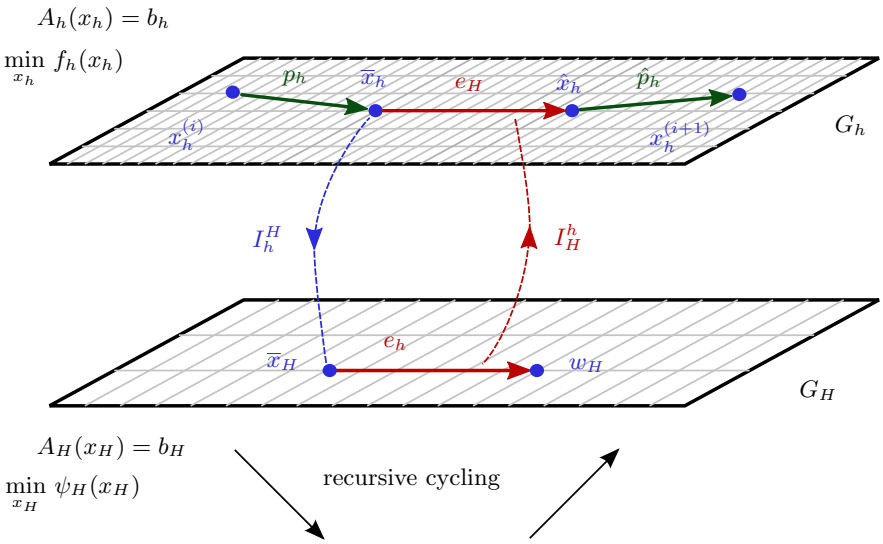

Figure 2: The FAS two-grid cycle for solving a nonlinear equation or minimizing an objective function.

one obtains an effective two-grid cycle for optimizing f_h. This linear modification to f_H is one of the key ideas of multilevel optimization, as proposed in the MG/Opt method of Nash (2000) and Lewis & Nash (2005). The reasoning behind this modification is the following: if one takes as objective function the residual equation of a nonlinear equation, the minimizer of $\psi_H(x_H)$ reduces to the FAS coarse grid correction for this equation.

Remark that the minimization of (13) starts at the initial guess $\overline{x}_H$. Hence, one seeks an update e_H such that

$$\psi_H(\overline{x}_H + e_H) := f_H(\overline{x}_H + e_H) - g^{\mathrm{E}}(\overline{x}_H + e_H, \operatorname{grad} f_H(\overline{x}_H) - I_h^H g_h), \tag{14}$$

with $g_h := \operatorname{grad} f_h(\overline{x}_h)$, is sufficiently minimized.

For the coarse grid correction e_H to be useful, the error has to be representable on the coarse grid, i.e., it has to be smooth. Much like in classic multigrid, the usual cheap first-order optimization methods can be used to smooth the error. Practice has shown that (weighted versions) of steepest descent, coordinate search and limited memory BFGS are effective smoothers for a wide range of large-scale multilevel optimization problems, see, e.g., Gratton *et al.* (2010).

Although we perform optimization, the principle behind the two-grid cycle does not change, except for the introduction of (13). In Figure 2, we therefore added the minimization of the fine and coarse scale objective functions f_h and ψ_H instead of solving equations with the operators A_h and A_H. Finally, we are in the position to generalize this to $\mathbf{S}_+^{n,p}$.

Remark 4.1. Our exposition of multilevel optimization is overly simplistic. In contrast to linear multigrid, (multilevel) optimization methods need to be robustified with line-searches or a Trust-Region updating scheme. This is an active area of research and we refer to Borzí (2005); Gratton *et al.* (2008); Wen & Goldfarb (2009) for the actual convergence proofs of these methods.

4.3 Generalization to Riemannian manifolds

We detail how the two-grid optimization cycle from above can be generalized to the retraction-based framework on a manifold. Let $\mathcal{M}_h$ denote a fine scale manifold and $\mathcal{M}_H$ a coarse scale manifold. Both manifolds have their own metric g_h and g_H (which are, in this presentation, assumed to be the Euclidean metric g^{E}).

In Figure 3, we have illustrated the Riemannian two-grid cycle between the manifolds $\mathcal{M}_h$ and $\mathcal{M}_H$. This two-grid cycle is based on the typical multigrid components.

Transfer operators. As usual, we need intergrid transfer operators between the fine and coarse scales. The manifolds $\mathcal{M}_h, \mathcal{M}_H$ are embedded in the Euclidean spaces of matrices (though of different sizes), so we can regard elements of $\mathcal{M}_h, \mathcal{M}_H$ as standard matrices belonging to a fine and a coarse tensor-product grid. This

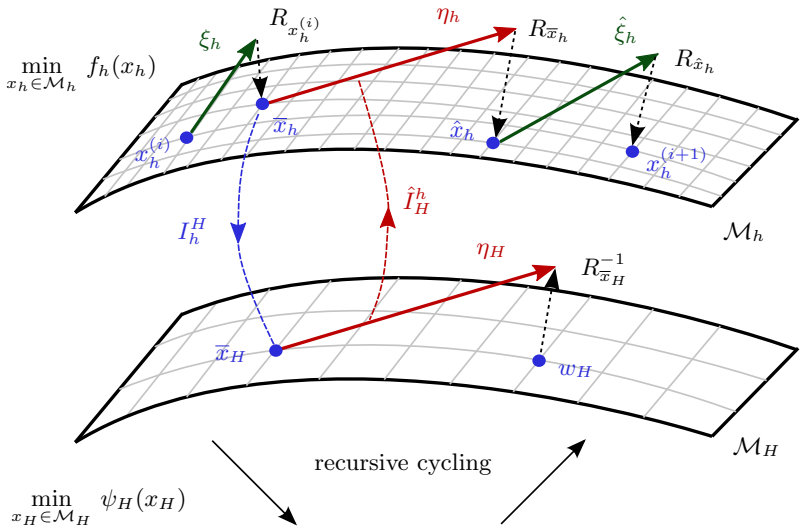

Figure 3: The Riemannian two-grid cycle for minimizing an objective function on a manifold.

allows us to simply use the intergrid transfers (2)–(3) of tensor-product multigrid of Section 2. Indeed, by restricting these operators to the manifold, one can verify that

$$I_h^H|_{\mathcal{M}_h} : \mathcal{M}_h \to \mathcal{M}_H, \quad I_H^h|_{\mathcal{M}_H} : \mathcal{M}_H \to \mathcal{M}_h. \tag{15}$$

yields suitable operators.

Moreover, we need an additional set of intergrid transfer operators for the tangent vectors:

$$\hat{I}_h^H : T_{x_h}\mathcal{M}_h \to T_{x_H}\mathcal{M}_H, \quad \text{with } x_h \in \mathcal{M}_h \text{ and } x_H := I_h^H(x_h) \in \mathcal{M}_H, \tag{16}$$

$$\hat{I}_H^h : T_{x_H}\mathcal{M}_H \to T_{x_h}\mathcal{M}_h, \quad \text{with } x_H \in \mathcal{M}_H \text{ and } x_h := I_H^h(x_H) \in \mathcal{M}_h. \tag{17}$$

Note that, $\hat{I}_h^H$ depends on x_h, and $\hat{I}_H^h$ on x_H.

Since these tangent spaces are linear spaces of the same size as the manifolds, we can also use the intergrid transfer operators of tensor-product multigrid for (16)–(17). However, the range of these operators is not necessarily the tangent space of the transferred points x_h or x_H. We therefore follow the transfer by an orthogonal projection onto the new tangent space,

$$\hat{I}_h^H := \mathrm{P}_{x_H}^{\mathrm{t}} \circ I_h^H|_{T_{x_h}\mathcal{M}_h}, \quad \hat{I}_H^h := \mathrm{P}_{x_h}^{\mathrm{t}} \circ I_H^h|_{T_{x_H}\mathcal{M}_H}.$$

Smoothers. The smoother on $\mathcal{M}_h$ is conceptually any relatively inexpensive method that aims at minimizing f_h: given $x_h^{(i)}$, it returns a tangent vector ξ_h such that after retraction, the error of the new iterate $\overline{x}_h = R_{x_h^{(i)}}(\xi_h)$ is smooth.

Although there are many choices, we base our smoother on a familiar Riemannian optimization method: the truncated CG (tCG) algorithm as mentioned in Section 3. The tCG method will give a tangent vector that, after a possible weighting, is retracted to yield a new iterate. Observe that one successful iteration of tCG is a step of steepest descent. In case of linear systems, this update is closely related to the Richardson iteration, which is a well-known smoother in classic multigrid. We expect that this Riemannian iteration will have similar smoothing properties.

The tCG algorithm can benefit from preconditioning by a positive definite approximation of the Hessian. This allows us to use more powerful preconditioners than the basic Richardson iteration. Suppose we can approximate the Riemannian Hessian by its diagonal, then one successful application of tCG will be a step of the Jacobi iteration. It is clear that better preconditioners should lead to better smoothers. We will demonstrate that this is indeed the case in the numerical experiments of Section 5.

This preconditioner used in the smoother can be computed from the same principles as the preconditioner Prec $f(x)$ in (7) of Section 3. Let $\widetilde{A}$ be an approximation of A such that $(\widetilde{A} \otimes I + I \otimes \widetilde{A})^{-1}$ is a smoother

for a tensor-product multigrid cycle on the Lyapunov equation. Then the preconditioner

$$T_x\mathbf{S}_+^{n,p} \to T_x\mathbf{S}_+^{n,p},\ \xi \mapsto \mathrm{P}_x^{\mathrm{t}}(\widetilde{A}\xi + \xi\widetilde{A})$$

can be directly applied in a tCG iteration for any symmetric and positive definite approximation $\widetilde{A}$ since it has exactly the same form as Prec $f(x)$. Of course, it must be feasible to solve shifted systems with $\widetilde{A}$ efficiently. However, it is still possible to use powerful Jacobi block-smoothers.

The coarse-grid correction. On the coarse level $\mathcal{M}_H$, we need to modify the objective function in a similar way as in FAS. To generalize this modification to a manifold, we first rewrite the Euclidean modification of (14). Let $\kappa_H := \operatorname{grad} f_H(\overline{x}_H) - I_h^H g_h$ with $g_h := \operatorname{grad} f_h(\overline{x}_h)$. Then (14) becomes

$$\begin{aligned}\psi_H(\overline{x}_H + e_H) &:= f_H(\overline{x}_H + e_H) - g^{\mathrm{E}}(\overline{x}_H + e_H, \kappa_H),\\ &= f_H(\overline{x}_H + e_H) - g^{\mathrm{E}}(e_H, \kappa_H) + c, \quad c \in \mathbf{R}.\end{aligned}$$

Since we will optimize for e_H, the factor $g^{\mathrm{E}}(\overline{x}_H, \kappa_H)$ is constant and we omit it from the objective function.

The key idea in our generalization is that we regard e_H not as the update itself, but as the direction of the update. In other words, it will be a tangent vector in $T_{\overline{x}_H}\mathcal{M}_H$. Looking at the principle behind retraction-based optimization, like in Section 3, this seems very natural. Indeed, it allows us to lift the function ψ_H by aid of the retraction $R_{\overline{x}_H}$. This gives a coarse objective function that is suitable for Riemannian optimization:

$$\hat{\psi}_{\overline{x}_H} : T_{\overline{x}_H}\mathcal{M}_H \to \mathbf{R},\ \eta_H \mapsto f_H(R_{\overline{x}_H}(\eta_H)) - g_{\overline{x}_H}(\eta_H, \kappa_H), \tag{18}$$

with $\kappa_{\overline{x}_H} := \operatorname{grad} f_H(\overline{x}_H) - \hat{I}_h^H(\operatorname{grad} f_h(\overline{x}_h))$. Obviously, the gradients are the Riemannian gradients of f_H and f_h. By definition of $\hat{I}_h^H$, the subtraction of the two tangent vectors is well defined.

Riemannian two-grid cycle. In Algorithm 2, we listed the final two-grid cycle to optimize an objective function on a Riemannian manifold. The smoother is denoted by the function `tCG`, where the preconditioning by Prec $f(x)$ is silently assumed. After this preconditioned tCG step, a possible scaling by ω_h of the tangent vector is possible and after retracting, we obtain the smoothed iterate $\overline{x}_h$.

Like Remark 4.1, the actual implementation of Algorithm 2 will need to be robustified with line-search or Trust-Region. In our numerical experiments, we used a Trust-Region mechanism inspired by the RTR algorithm of Absil *et al.* (2007) combined with the Recursive Trust-Region method from Gratton *et al.* (2008).

Algorithm 2 TG-RTR: Riemannian two-grid cycle to minimize f_h

1: **for** $i = 1, 2, \ldots$ **do**
2: $\quad \xi_h = \mathtt{tCG}^{\nu_1}(x_h^{(i)}, f_h)$
3: $\quad \overline{x}_h = R_{x_h^{(i)}}(\omega_h\, \xi_h)$
4: $\quad \overline{x}_H = I_h^H \overline{x}_h$
5: $\quad \kappa_H = \operatorname{grad} f_H(\overline{x}_H) - \hat{I}_h^H(\operatorname{grad} f_h(\overline{x}_h))$
6: $\quad$ define the coarse-grid function $\hat{\psi}_{\overline{x}_H}$ of (18) based on κ_H
7: $\quad$ minimize $\hat{\psi}_{\overline{x}_H}$ for the coarse-grid correction η_H
8: $\quad \eta_h = \hat{I}_H^h \eta_H$
9: $\quad \hat{x}_h = R_{\overline{x}_h}(\eta_h)$
10: $\quad \hat{\xi}_h = \mathtt{tCG}^{\nu_2}(\hat{x}_h, f_h)$
11: $\quad x_h^{(i+1)} = R_{\hat{x}_h}(\omega_h\, \hat{\xi}_h)$
12: **end for**

5 Numerical results

We report on the numerical properties of Algorithm 2 applied to the Lyapunov equation for the singularly perturbed ($\epsilon \to 0$) two-dimensional (2D) diffusion equation

$$\frac{\partial^2 u}{\partial x^2} + \epsilon\frac{\partial^2 u}{\partial y^2} = b(x, y), \quad b(x, y) := e^{x+2y}\sin(3\pi x)\sin(\pi y), \tag{19}$$

with homogeneous boundary conditions on the square $[0, 1]^2$. The multigrid components were all implemented in MATLAB which is sufficient to illustrate the typical multigrid properties: mesh-independent convergence and a computational cost that scales linearly with the problem size.

After discretization of (19) using the usual five-point stencil with a mesh of size $h = 1/(\sqrt{n}+1)$, one obtains the system matrix

$$A = \overline{A} \otimes I_n + \epsilon I_n \otimes \overline{A} \in \mathbf{R}^{n \times n}, \tag{20}$$

with $\overline{A}$ the discretization of a one-dimensional Laplace equation with a three-point stencil. The right-hand side $b(x, y)$ is discretized into a vector $c \in \mathbf{R}^n$. The corresponding Lyapunov equation is constructed using this A and with $C = cc^T$. To assess the accuracy of the obtained solutions, we use the following relative residual:

$$r(x) := \|Ax + xA^T\|_{\mathrm{F}} / (2\|A\|_2 \|x\|_{\mathrm{F}} + \|C\|_{\mathrm{F}}). \tag{21}$$

This residual is also used in the low-rank solver of Simoncini (2007).

Remark 5.1. Remark that the structure of (20) is only needed to illustrate the effectiveness of our multigrid algorithm for anisotropic diffusion and for notational convenience. The actual algorithm does not need this specific structure and any other block smoother can be used.

Classical multigrid theory tells us that a point smoother in combination with standard coarsening will not be an effective solver for equations with anisotropic diffusion. We can confirm this for the Riemannian multilevel algorithm as well. We took the diagonal part of A as preconditioner for tCG followed by a weighting of $\omega = 0.75$. This is the equivalent of a weighted Jacobi smoother in standard multigrid. Equation (19) is discretized with meshes $h = 32 \cdot 2^{-l}$ for growing levels $l = 0, 1, 2, \ldots, l_{\max}$. The coarsest problem of size $n = 32$, i.e., level $l = 0$, is solved to full numerical accuracy by the method on $\mathbf{S}_+^{32,8}$ of Section 3.

Figure 4 shows the convergence history for the cases $\epsilon = 1$ and $\epsilon = 10^{-6}$ when solving the Lyapunov equations for a rank $p = 8$ minimizer of f. It is clear that the strategy of a point smoother is indeed effective for $\epsilon = 1$ but not for $\epsilon = 10^{-6}$.

In Table 1, the time per iteration and the final residuals are displayed. First notice that all problems for $\epsilon = 1$ were indeed solved up to full numerical accuracy if the error is measured by the relative residual (21). Since these problems were clearly over-solved, we displayed the time *per iteration* in Table 1. Despite the fact that the implemented TG-RTR algorithm has to follow a Trust-Region strategy and not a fixed V-cycle pattern, the amount of work scales almost linearly with the problem size.

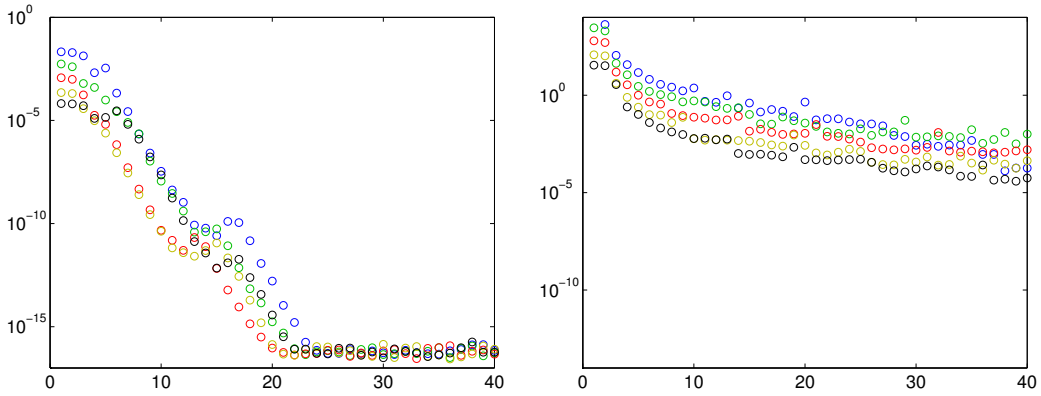

Figure 4: Convergence of the gradient for TG-RTR with a point smoother. The Lyapunov equation originates from a 2D diffusion equation with isotropic (left panel) and anisotropic (right panel) diffusion. The sizes of the discretizations are 4096 (o), 16384 (o), 65536 (o), 262144 (o) and 1048576 (o).

ϵ	size n	4096	16384	65536	262144	1048576
1	time (s.)	0.37	1.25	5.01	21.4	99.4
	$r(x_{40})/10^{-16}$	300	23	1.1	0.01	0.06
10^{-6}	time (s.)	0.37	1.29	4.98	21.1	97.8
	$r(x_{40})/10^{-8}$	17	1.3	0.4	0.09	0.007

Table 1: Time per iteration and final residual for the problem of Figure 4. Observe that the residuals for $\epsilon = 10^{-6}$ are 10^8 times greater than those for $\epsilon = 1$.

It is well known that a line smoother in combination with standard coarsening is an effective strategy for multigrid on the PDE (19) when $\epsilon \to 0$. From a standard LFA analysis (Vandereycken, 2010, Section 5.2.3),

we know that a smoother which is based on the approximation

$$\widetilde{L} = \widetilde{A} \otimes I_N + I_N \otimes \widetilde{A}, \quad \text{with } \widetilde{A} := \operatorname{diag}(\overline{A}) \otimes I_n + \epsilon I_n \otimes \overline{A},$$

will be effective for tensor-product multigrid on the corresponding Lyapunov equation. In Figure 5, we have applied this operator as a preconditioner in tCG for TG-RTR. It is obvious that this is again an effective strategy.

Comparing the time per iteration in Table 2 with that of Table 1, we see that the method using line Jacobi is basically twice as costly as that of point Jacobi, while both methods converge equally fast (compare the left panels of Figure 4 and Figure 5). However, this extra cost of line Jacobi gives a more robust multigrid strategy.

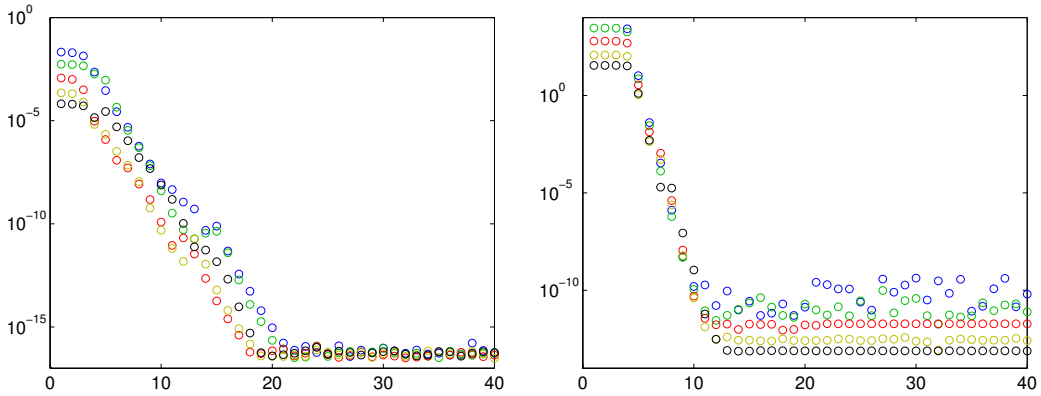

Figure 5: Same experimental setup as used in Figure 4 but now with a line smoother instead of a point smoother.

ϵ	size n	4096	16384	65536	262144	1048576
1	time (s.)	0.6	2.4	9.7	41	183
	$r(x_{40})/10^{-16}$	300	23	1.5	0.01	0.01
10^{-6}	time (s.)	0.7	2.4	9.5	40	173
	$r(x_{40})/10^{-16}$	3822	243	3.1	0.16	0.01

Table 2: Time per iteration and final residual for the problem of Figure 5.

References

Absil, P.-A., Baker, C. & Gallivan, K. (2007) Trust-region methods on Riemannian manifolds. *Found. Comput. Math.*, **7**, 303–330.

Absil, P.-A., Mahony, R. & Sepulchre, R. (2008) *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton University Press.

Antoulas, A. C. (2005) *Approximation of Large-Scale Dynamical Systems*. Adv. Des. Control. SIAM, Philadelphia.

Benner, P., Mehrmann, V. & Sorensen, D. (eds) (2005) *Dimension Reduction of Large-Scale Systems*. Springer- Verlag.

Borzí, A. (2005) On the convergence of the MG/OPT method. *PAMM*, **5**, 735–736.

Grasedyck, L. (2004) Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation. *Numer. Linear Algebra Appl.*, **11**, 371–389.

Grasedyck, L. & Hackbusch, W. (2007) A multigrid method to solve large scale Sylvester equations. *SIAM J. Matrix Anal. Appl.*, **29**, 870–894.

Gratton, S., Sartenaer, A. & Toint, P. L. (2008) Recursive trust-region methods for multiscale nonlinear optimization. *SIAM J. Optim.*, **19**, 414–448.

Gratton, S., Mouffe, M., Sartenaer, A., Toint, P. L. & Tomanos, D. (2010) Numerical experience with a recursive trust-region method for multilevel nonlinear optimization. *Optim. Meth. Softw.*, **25**, 359–386.

Lewis, R. M. & Nash, S. G. (2005) Model problems for the multigrid optimization of systems governed by differential equations. *SIAM J. Sci. Comput.*, **26**, 1811–1837.

Li, J.-R. & White, J. (2004) Low-rank solution of Lyapunov equations. *SIAM Rev.*, **46**, 693–713.

Nash, S. G. (2000) A multigrid approach to discretized optimization problems. *Optim. Meth. Softw.*, **14**, 99–116.

Penzl, T. (1997) A multi-grid method for generalized Lyapunov equations. *SFB393/97-24*. Technische Universität Chemnitz.

Penzl, T. (1999) A cyclic low-rank Smith method for large sparse Lyapunov equations. *SIAM J. Sci. Comput.*, **4**, 1401–1418.

Rosen, I. G. & Wang, C. (1995) A multilevel technique for the approximate solution of operator Lyapunov and algebraic Riccati equations. *SIAM J. Numer. Anal.*, **32**, 514–541.

Simoncini, V. (2007) A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Sci. Comput.*, **29**, 1268–1288.

Trottenberg, U., Oosterlee, C. W. & Schuller, A. (2000) *Multigrid*. Academic Press.

Vandereycken, B. (2010) Riemannian and multilevel optimization for rank-constrained matrix problems. *Ph.D. thesis*, Department of Computer Science, Katholieke Universiteit Leuven.

Vandereycken, B. & Vandewalle, S. (2010) A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, **31**, 2553–2579.

Wen, Z. & Goldfarb, D. (2009) A line search multigrid method for large-scale nonlinear optimization. *SIAM J. Optim.*, **3**, 1478–1503.