



École Polytechnique Fédérale de Lausanne
School of Basic Sciences, Mathematics Section

Master of Science in Computational Science and Engineering
Master's Project - Projet de Master

Host institution: Università di Pisa - Dipartimento di Matematica

Preconditioned Low-Rank Riemannian Optimization for Multiterm Linear Matrix Equations

Advisors:

Prof. Daniel Kressner,
EPFL

Prof. Leonardo Robol,
Università di Pisa

Candidate:

Ivan Bioli

Academic Year 2023-2024

Alla mia famiglia,
a Flaminia.

Ringraziamenti

Di tutto questo lavoro di tesi, vorrei che fosse chiaro almeno il mio enorme Grazie alle persone che mi hanno accompagnato lungo questo percorso. Questa è in realtà una seconda stesura dei ringraziamenti, dato che la prima era stata troppo frettolosa e il poco tempo dedicatole non rendeva giustizia a tutti coloro che intendo ringraziare.

Vorrei innanzitutto ringraziare la mia famiglia, per avere sempre supportato il loro Baccello. Un Grazie a Mamma e Babbo, ma anche ai miei fratelli Ettore e Cesare, a cui voglio bene nonostante tutte le azzuffate. Grazie ai miei "Nonni di Pisa" (anche se sono di La Spezia) Gabriella e Gianfranco, e ai miei "Nonni di Terra", Lidia e Adriano, per l'affetto come solo i nonni sanno dimostrare. Grazie a Zia Annalisa e Zio Marcello, i miei zii preferiti (oltre che gli unici...), e a Edoardo. Grazie a Zio Piero e Massimo. E perché no ... Grazie anche a Diana, Sally, Argo e Minù.

Un Grazie a tutti i miei amici. Una menzione d'onore per Gianluca, con cui ho condiviso tredici anni di scuola, ma anche per Cosimo, Francesco, Virginia, Francesca, Martina ed Eleonora. Grazie di essere rimasti legati, anche se siamo fisicamente lontani. Grazie a Lorenzo, per farmi sempre sentire Nibali quando andiamo in bici insieme. Grazie a tutti gli amici che ho incontrato in questi due anni a Losanna, e in particolare modo a Federico, Matteo R., Nicolò, Laura, Giacomo, Giulia, Anna, Paolo, Matteo C. e Michele. Un Grazie a Filippo, Davide, Dania, Leonardo e Giuseppe, per i bei momenti trascorsi al Dip e dintorni.

Ringrazio i miei due relatori, il Prof. Daniel Kressner e il Prof. Leonardo Robol, senza i quali questo lavoro di tesi non sarebbe stato possibile. Vorrei però ringraziare anche tutti gli insegnanti che hanno contribuito alla mia formazione, accademica e non solo. Tra questi, ringrazio in maniera particolare i professori Gaiffi, Danti, De Biase, Viaggio e Di Lena, e le maestre Nadia e Margherita.

Ho lasciato per ultima la persona più importante, Flaminia. Grazie per essere sempre stata al mio fianco in questi anni, per avermi supportato e sopportato, per essere cresciuti insieme. Non vedo l'ora che i nostri progetti diventino realtà, sempre insieme.

Pisa, 19 Gennaio 2024

Ivan Bioli

Acknowledgements

Of all this thesis work, I would like it to be clear, at least, my enormous gratitude to the people who have accompanied me on this journey. This is actually a second draft of acknowledgements, as the first was too rushed, and the little time devoted to it did not do justice to all those I intend to thank.

First and foremost, I would like to thank my family for always supporting their Baccello. Thanks to Mum and Dad, but also to my brothers Ettore and Cesare, whom I love despite all the fights. Thanks to my "Nonni di Pisa" (even though they are from La Spezia), Gabriella and Gianfranco, and to my "Nonni di Terra", Lidia and Adriano, for the affection as only grandparents can show. Thanks to Zia Annalisa and Zio Marcello, my favorite (and only...) uncles, and to Edoardo. Thanks to Zio Piero and Massimo. And why not... Thanks also to Diana, Sally, Argo, and Minù.

Thanks to all my friends. Special mention to Gianluca, with whom I shared thirteen years of school, but also to Cosimo, Francesco, Virginia, Francesca, Martina, and Eleonora. Thank you for staying connected even though we are physically distant. Thanks to Lorenzo, for always making me feel like Nibali when we ride bikes together. Thanks to all the friends I have met in these two years in Lausanne, and especially to Federico, Matteo R., Nicolò, Laura, Giacomo, Giulia, Anna, Paolo, Matteo C., and Michele. Thanks to Filippo, Davide, Dania, Leonardo, and Giuseppe, for the great times spent at the Dip and around.

I thank my two supervisors, Prof. Daniel Kressner and Prof. Leonardo Robol, without whom this thesis work would not have been possible. However, I would also like to thank all the teachers who have contributed to my education, both academic and otherwise. Among these, I particularly thank professors Gaiffi, Danti, De Biase, Viaggio, and Di Lena, and teachers Nadia and Margherita.

I saved the most important person, Flaminia, for last. Thank you for always being by my side over these years, for supporting and enduring me, for growing together. I cannot wait for our projects to become true, always together.

Pisa, January 19, 2024

Ivan Bioli

Abstract

This thesis focuses on the development and analysis of Riemannian optimization methods for solving multiterm linear matrix equations, when the associated linear operator is symmetric positive definite. Assuming low-rank approximability of the solution, we employ first- and second-order Riemannian optimization algorithms on low-rank matrix manifolds. By alternating between fixed-rank Riemannian optimization and rank updates, we devise a rank-adaptive algorithm that does not require fixing the rank a priori. To enhance competitiveness with existing solvers on realistic problems, significant efforts are dedicated to developing efficient Riemannian preconditioners.

In the initial part of the thesis, we detail the application of preconditioned Riemannian optimization algorithms to multiterm linear matrix equations, using an energy-norm-based cost function. We provide a detailed analysis of global and local convergence guarantees, extending them to the preconditioned case by interpreting Riemannian preconditioners as metric changes. Additionally, we propose the use of the projected Euclidean Hessian as a model Hessian for the Riemannian Trust-Region algorithm, proving that this retains local quadratic convergence to the exact solution of the multiterm linear matrix equation, if it is on the manifold, along with notable computational advantages over the Riemannian Hessian.

Our focus then shifts to the development of effective Riemannian preconditioners derived from dominant terms in the equation. We contribute with exact and approximate preconditioners based on terms of generalized Sylvester type. Significant attention is paid to ensuring efficient application of the preconditioner.

In numerical experiments, we illustrate the effectiveness of the developed Riemannian preconditioners, comparing our approach with established solvers such as Conjugate Gradient with truncation. Results indicate that the Riemannian optimization approach, in both its fixed-rank and rank-adaptive variants, is competitive with other solvers and can avoid excessive growth of intermediate ranks.

Key words: multiterm linear matrix equations, Riemannian optimization, Riemannian preconditioning, matrix manifolds, low-rank.

Résumé

Cette thèse se concentre sur le développement et l'analyse de méthodes d'optimisation riemannienne pour résoudre des équations matricielles linéaires à plusieurs termes, lorsque l'opérateur linéaire associé est symétrique et défini positif. En supposant une approximabilité de bas rang de la solution, nous utilisons des algorithmes d'optimisation riemannienne du premier et du second ordre sur des variétés de matrices de bas rang. En alternant entre l'optimisation riemannienne à rang fixe et les mises à jour de rang, nous concevons un algorithme adaptatif au rang qui ne nécessite pas de fixer le rang a priori. Pour améliorer la compétitivité par rapport aux solveurs existants sur des problèmes réalistes, des efforts significatifs sont consacrés au développement de préconditionneurs riemanniens efficaces.

Dans la partie initiale de la thèse, nous détaillons l'application d'algorithmes d'optimisation riemannienne préconditionnés aux équations matricielles linéaires à plusieurs termes, en utilisant une fonction coût basée sur la norme énergétique. Nous fournissons une analyse détaillée des garanties de convergence globale et locale, les étendant au cas préconditionné en interprétant les préconditionneurs riemanniens comme des changements de métrique. De plus, nous proposons l'utilisation de l'hessien euclidien projeté comme hessien modèle pour l'algorithme de Riemannian Trust-Region, démontrant ainsi qu'il conserve une convergence quadratique locale vers la solution exacte de l'équation matricielle linéaire à plusieurs termes, si celle-ci se trouve sur la variété, en plus d'offrir des avantages computationnels notables par rapport à l'hessien riemannien.

Notre attention se tourne ensuite vers le développement de préconditionneurs riemanniens efficaces dérivés des termes dominants dans l'équation. Nous contribuons avec des préconditionneurs exacts et approximatifs basés sur des termes de type Sylvester généralisé. Une attention particulière est accordée à garantir une application efficace du préconditionneur.

Dans des expériences numériques, nous illustrons l'efficacité des préconditionneurs riemanniens développés, en comparant notre approche à des solveurs établis tels que le gradient conjugué avec troncature. Les résultats indiquent que l'approche d'optimisation riemannienne, dans ses variantes à rang fixe et adaptatives au rang, est compétitive avec d'autres solveurs et peut éviter une croissance excessive des rangs intermédiaires.

Mots-clés : équations matricielles linéaires à plusieurs termes, optimisation riemannienne, préconditionnement riemannien, variétés de matrices, bas rang.

Sommario

Questa tesi si concentra sullo sviluppo e sull'analisi di metodi di ottimizzazione Riemanniana per la risoluzione di equazioni matriciali lineari multitermine, quando l'operatore lineare associato è simmetrico definito positivo. Supponendo che la soluzione sia approssimabile a rango basso, utilizziamo algoritmi di ottimizzazione Riemanniana del primo e secondo ordine su varietà di matrici a rango basso. Alternando tra ottimizzazione Riemanniana a rango fisso e aggiornamenti di rango, sviluppiamo un algoritmo adattivo al rango che non richiede il fissare il rango a priori. Per migliorare la competitività su problemi realistici rispetto agli altri risolutori, dedichiamo sforzi significativi allo sviluppo di preconditionatori Riemanniani efficienti.

Nella parte iniziale della tesi, dettagliamo l'applicazione di algoritmi di ottimizzazione Riemanniana preconditionati a equazioni matriciali lineari multitermine, utilizzando una funzione costo basata sulla norma energia. Forniamo un'analisi dettagliata delle garanzie di convergenza globale e locale, estendendole al caso preconditionato interpretando i preconditionatori Riemanniani come cambiamenti di metrica. Inoltre, proponiamo l'uso dell'Hessiano Euclideo proiettato come Hessiano modello per l'algoritmo di Riemannian Trust-Region, dimostrando che questo preserva la convergenza quadratica locale alla soluzione esatta dell'equazione matriciale multitermine, se questa è sulla varietà, insieme a notevoli vantaggi computazionali rispetto all'Hessiano Riemanniano.

Il nostro focus si sposta quindi sullo sviluppo di preconditionatori Riemanniani efficaci derivati dai termini dominanti nell'equazione. Contribuiamo con preconditionatori esatti e approssimati basati su termini di tipo Sylvester generalizzato. Prestiamo particolare attenzione a garantire un'applicazione efficiente del preconditionatore.

Negli esperimenti numerici, illustriamo l'efficacia dei preconditionatori Riemanniani sviluppati, confrontando il nostro approccio con risolutori consolidati come il gradiente coniugato con troncamento. I risultati indicano che l'approccio di ottimizzazione Riemanniana, sia nelle sue varianti a rango fisso che adattive al rango, è competitivo con altri risolutori e può evitare una crescita eccessiva dei ranghi intermedi.

Parole chiave: equazioni matriciali lineari multitermine, ottimizzazione Riemanniana, preconditionamento Riemanniano, varietà di matrici, rango basso.

Contents

Ringraziamenti	ii
Acknowledgements	iii
Abstract (English/Français/Italiano)	iv
Notation	x
1 Introduction	1
1.1 Preconditioned Riemannian optimization	2
1.2 Contributions and outline of the thesis	3
1.3 Notational Conventions	4
1.4 Introduction to low-rank approximations and low-rank methods	5
1.4.1 Singular value decomposition and low-rank approximations	5
1.4.2 Low-rank methods for multiterm linear matrix equations	6
2 Riemannian optimization over fixed-rank matrices	11
2.1 Basic tools for Riemannian optimization	11
2.1.1 Riemannian gradient and Hessian	11
2.1.2 Moving along curves and necessary optimality conditions	14
2.1.3 Retractions	15
2.1.4 Transporters	17
2.2 Embedded geometry of fixed-rank matrices	18
3 Riemannian optimization algorithms	25
3.1 The notion of convergence	25
3.2 Line-search methods on manifolds	27
3.2.1 General framework	27
3.2.2 Riemannian gradient descent (R-GD)	29
3.2.3 Riemannian nonlinear conjugate gradient (R-NLCG)	30
3.3 Riemannian Trust-Region (R-TR)	32
3.3.1 Convergence	34
3.3.2 R-TR with Riemannian Hessian	36
3.3.3 R-TR with projected Euclidean Hessian	37

3.4	Preconditioning	38
3.4.1	What preconditioning is and why it is needed	38
3.4.2	Riemannian projected preconditioned methods	39
3.4.3	Riemannian preconditioning	39
3.4.4	Preconditioned Riemannian nonlinear conjugate gradient	40
3.4.5	Preconditioned Riemannian Trust-Region method	41
3.4.6	Preconditioners of the form $\mathcal{P}_X = \text{Proj}_X \circ \mathcal{P} \circ \text{Proj}_X$	44
4	Cost function and convergence for multiterm linear matrix equations	47
4.1	Cost function, gradient and Hessian	47
4.1.1	Cost function	47
4.1.2	Gradient	50
4.1.3	Hessian	51
4.2	Line-search initialization	51
4.3	Critical points and convergence	52
4.3.1	Critical points and residual	52
4.3.2	Local convergence	53
4.3.3	Global convergence with regularized cost function	54
4.4	Rank adaptivity	55
5	Preconditioners for multiterm linear matrix equations	59
5.1	$\mathcal{P}X = EXD$ where $E, D \succ 0$	60
5.1.1	Applying $\mathcal{P}_X^{-1}$	60
5.1.2	Geometry with the metric induced by $\mathcal{P}X = EXD$	62
5.1.3	Difference between the two approaches	68
5.2	$\mathcal{P}X = AXD + EXB$ where $A, B, D, E \succ 0$	69
5.2.1	The case $\mathcal{P}X = AX + XB$	69
5.2.2	The case $\mathcal{P}X = AXD + EXB$	72
5.3	Alternating-Direction Implicit (ADI) iteration	74
5.3.1	ADI iteration for Sylvester equations	75
5.3.2	Factored ADI (fADI) iteration	78
5.3.3	ADI-like iteration on the tangent space (tangADI)	78
6	Optimizing over fixed-rank symmetric positive semidefinite matrices	82
6.1	Embedded geometry of fixed-rank symmetric positive semidefinite matrices	82
6.2	Preconditioners	84
6.2.1	$\mathcal{P}X = MXM$ where $M \succ 0$	84
6.2.2	Applying $\mathcal{P}_X^{-1}$	84
6.2.3	$\mathcal{P}X = AXM + MXA$ where $A, M \succ 0$	85
6.2.4	tangADI	85
6.3	Rank adaptivity	86

7	Numerical Experiments	87
7.1	Finite difference discretization of 2D PDEs with separable variables . . .	88
7.1.1	Numerical results	89
7.2	Stochastic Galerkin matrix equations	94
7.2.1	Numerical results	97
7.3	Modified Bilinear Rail problem	101
7.3.1	Numerical results	101
8	Conclusions	104
8.1	Future outlooks	105
	Bibliography	107
A	Convergence of RTR with tCG	118
A.1	Proof of local convergence of RTR with tCG	118

Notation

Basic Symbols and Linear Algebra

$\mathbb{R}$	Real numbers
$\mathbb{R}^n$	n -dimensional vectors over $\mathbb{R}$
$\mathbb{R}^{m \times n}$	Matrices of size $m \times n$ over $\mathbb{R}$
$\text{Sym}(n)$	Symmetric matrices in $\mathbb{R}^{n \times n}$
$\mathcal{L}(\mathcal{E}, \mathcal{F})$	Linear maps between the vector spaces $\mathcal{E}$ and $\mathcal{F}$
$\ \cdot\ _F$	Frobenius norm
$\langle \cdot, \cdot \rangle, \ \cdot\ $	Generic inner product, and corresponding norm; if the arguments are matrices, and it is not specified otherwise, $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product
$\langle \cdot, \cdot \rangle_{\mathcal{L}}, \ \cdot\ _{\mathcal{L}}$	Inner product induced by $\mathcal{L}$, and corresponding norm
$\kappa(A), \kappa_2(A)$	Condition number of A in a generic norm $\ \cdot\ $, and in the 2-norm
$\Lambda(A), \Lambda(A, B)$	Spectrum of A , and of the matrix pencil (A, B)
$\lambda_i, \lambda_i(X)$	i -th eigenvalue of X , from largest to smallest if the spectrum is real
$\rho(A)$	Spectral radius of A
$A \succ 0$ ($A \succeq 0$)	A is symmetric positive definite (symmetric positive semidefinite)
$\sigma_i, \sigma_i(X)$	i -th singular value of X , from largest to smallest
$\text{diag}(d_1, \dots, d_k)$	Diagonal matrix with diagonal entries $d_1, \dots, d_k$
$\otimes$	Kronecker product
vec	Vectorization

Differential Geometry and Riemannian Optimization

$\mathcal{M}$	Manifold
$\mathcal{E}, \mathcal{F}$	Vector spaces, often endowed with a Euclidean structure
$\mathcal{M}_r$	Manifold of rank- r matrices
$\text{S}_{n,r}^+$	Manifold of rank- r symmetric positive semidefinite matrices in $\mathbb{R}^{n \times n}$
$\text{St}(p, r)$	Stiefel manifold, embedded in $\mathbb{R}^{p \times r}$
$T_X \mathcal{M}$	Tangent space to $\mathcal{M}$ at X
η, ξ	Tangent vectors
$T\mathcal{M}$	Tangent bundle of $\mathcal{M}$
$N_X \mathcal{M}$	Normal space to $\mathcal{M}$ at X

$\text{Proj}_X, \text{Proj}_X^\perp$	Orthogonal projector to $T_X\mathcal{M}$ and $N_X\mathcal{M}$
$\langle \cdot, \cdot \rangle_X, \ \cdot\ _X$	Riemannian metric on $T_X\mathcal{M}$, and corresponding norm
∇	Levi-Civita connection
$\frac{D}{dt}$	Covariant derivative induced by the Levi-Civita connection ∇
f	Scalar-valued smooth function
$\bar{f}$	Smooth extension of the smooth function f
Df	Differential of the function f
$\text{grad} f$	Riemannian gradient of the function f ; if f is defined on a Euclidean space, it is the Euclidean gradient
$\text{Hess} f$	Riemannian Hessian of the function f ; if f is defined on a Euclidean space, it is the Euclidean Hessian
$\hat{f}_X$	Pullback of f to the tangent space at X
R, R_X	Retraction
$\text{Proj}_{\mathcal{M}}$	Metric projection to $\mathcal{M}$
$T_{Y \leftarrow X}$	Transporter, from $T_X\mathcal{M}$ to $T_Y\mathcal{M}$
$\mathcal{P}_X$	Riemannian preconditioner on $T_X\mathcal{M}$
$\langle \cdot, \cdot \rangle_{\mathcal{P}_X}, \ \cdot\ _{\mathcal{P}_X}$	Inner product induced by $\mathcal{P}_X$ on $T_X\mathcal{M}$, and corresponding norm

Other symbols

$\mathcal{A}$	Linear operator $\mathcal{A}X = A_1XB_1^\top + A_2XB_2^\top + \cdots + A_\ell XB_\ell^\top$
$X_\star$	Solution to the multiterm linear matrix equation; it can also denote a non-degenerate local minimizer
$\mathcal{P}, \mathcal{B}$	Preconditioner
$A \doteq (A_1, \dots, A_k)$	A is represented by the tuple $(A_1, \dots, A_k)$

Abbreviations

ADI	Alternating Direction Implicit
CG	Conjugate gradient
FD	Finite differences
FE, FEM	Finite elements, and finite elements method
GD	Gradient descent
NLCG	Nonlinear conjugate gradient
PDE	Partial differential equation
SPD	Symmetric positive definite
SVD	Singular value decomposition
tCG	Steihaug-Toint truncated conjugate gradient
TR	Trust-region
TRs	Trust-region subproblem
TSVD	Truncated SVD

Chapter 1

Introduction

The focus of this work is the approximate solution of *multiterm linear matrix equations* of the form

$$A_1XB_1^\top + A_2XB_2^\top + \cdots + A_\ell XB_\ell^\top = F, \quad (1.1)$$

where $A_i \in \mathbb{R}^{m \times m}$, $B_i \in \mathbb{R}^{n \times n}$ for $i = 1, \dots, \ell$ are known coefficient matrices, $F = F_L F_R^\top \in \mathbb{R}^{m \times n}$ is a known right-hand side with $F_L \in \mathbb{R}^{m \times r_F}$, $F_R \in \mathbb{R}^{n \times r_F}$, and $X \in \mathbb{R}^{m \times n}$ is the unknown. We refer to (1.1) as *multiterm Sylvester equation* [Sim16]. A specific instance is the *multiterm Lyapunov equation*

$$AXM^\top + MXA^\top - \sum_{i=1}^{\ell} N_i X N_i^\top = F, \quad (1.2)$$

where $A, M, N_i \in \mathbb{R}^{n \times n}$, and $F = F^\top \in \mathbb{R}^{n \times n}$. Notably, if $X_\star \in \mathbb{R}^{n \times n}$ is a solution of (1.2), so is $X_\star^\top$; hence, if the solution is unique, it is also symmetric. Applications of multiterm linear matrix equations span various domains, including but not limited to stochastic and parametrized PDEs [PSS17; LPS14; BG13; KS11a; KT11; PE09], bilinear and stochastic control problems [BB13; BD11; Dam08; Dam04; Won68; Kle69], time-dependent PDE-constrained optimization problems [SB15], fluid-structure interaction problems [WBR20], all-at-once discretization of time-dependent PDEs [Pal21; MPW18], and matrix-based discretization of PDEs [PS16; HS21].

For $\ell = 2$ multiterm Sylvester equations (1.1) reduce to generalized Sylvester equations $AXD^\top + EXB^\top = F$. This reduction encompasses the well-known standard Sylvester equation $AX + XB^\top = F$. Similarly, multiterm Lyapunov equations include both the generalized Lyapunov equation $AXM^\top + MXA^\top = F$ and the standard Lyapunov equation $AX + XA^\top = F$. Due to their widespread applications, (generalized) Sylvester and Lyapunov equations have been extensively studied in the literature. For comprehensive surveys, we refer to [Sim16; BS13].

Equation (1.1) is a linear equation in X defined by the linear operator

$$\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}, \quad \mathcal{A}X = A_1XB_1^\top + A_2XB_2^\top + \cdots + A_\ell XB_\ell^\top. \quad (1.3)$$

1.1. Preconditioned Riemannian optimization

We confine our focus to the *symmetric positive definite* (SPD) scenario, i.e., we assume that $\mathcal{A}$ is a SPD linear operator. This assumption enables us to reformulate (1.1) in the form of an optimization problem:

$$\min_{X \in \mathbb{R}^{m \times n}} f(X) := \frac{1}{2} \langle \mathcal{A}X, X \rangle - \langle X, F \rangle. \quad (1.4)$$

This formulation is well-known for being equivalent to minimizing the $\mathcal{A}$ -induced norm of the error $\|X - \mathcal{A}^{-1}F\|_{\mathcal{A}}$, as discussed in detail in Section 4.1.1.

Additionally, we consider scenarios with large and sparse coefficient matrices. Despite the sparsity of the coefficient matrices, the solution is in general full. Consequently, explicitly computing and storing in memory an $m \times n$ full matrix X becomes impractical for large sizes m, n . However, in many applications the singular values of the solution X_* exhibit strong decay, making X_* suitable for a *low-rank* approximation $X_* \approx YW^\top$ where $Y \in \mathbb{R}^{m \times r}$, $W \in \mathbb{R}^{n \times r}$ and $r \ll m, n$. This property enables the development of low-rank algorithms, relying on computing and storing solely the thin factors Y and W . Our method is based on preconditioned Riemannian optimization on the smooth Riemannian manifold of (low) rank- r matrices.

1.1 Preconditioned Riemannian optimization

To obtain the "optimal" rank- r approximation of the solution, we consider a constrained version of the optimization problem (1.4), namely

$$\min_{X \in \mathcal{M}_r} f(X) \quad \text{where} \quad \mathcal{M}_r = \{X \in \mathbb{R}^{m \times n} : \text{rank}(X) = r\}. \quad (1.5)$$

Rather than resorting to conventional techniques for constrained optimization, we exploit that $\mathcal{M}_r$ is a smooth Riemannian manifold and apply methods from Riemannian optimization [AMS08; Bou23a; Sat21].

Riemannian optimization extends classical Euclidean optimization techniques to operate on smooth Riemannian manifolds. Instead of treating (1.5) as a constrained optimization problem in $\mathbb{R}^{m \times n}$, we reinterpret it as an unconstrained optimization problem on the curved Riemannian manifold $\mathcal{M}_r$. This necessitates the generalization of Euclidean optimization algorithms to enable optimization on such curved, smooth manifolds. All the methods we consider fit into the retraction-based framework of Riemannian optimization, which is illustrated in Figure 1.1. Given the current iterate X , the basic idea is to compute a tangent direction $\xi \in T_X \mathcal{M}_r$, where $T_X \mathcal{M}_r$ denotes the tangent space at X to $\mathcal{M}_r$. The next iterate is then found by moving along a curve starting from X with an initial velocity of ξ . A retraction R represents a smooth choice of such a curve, with respect to X and ξ . Within this framework, we can generalize standard Euclidean optimization algorithms, including gradient descent (GD) [Bou23a; AMS08], nonlinear conjugate gradient (NLCG) [Sat21; Sat22], and the trust-region (TR) method [ABG07; AMS08; Bou23a].

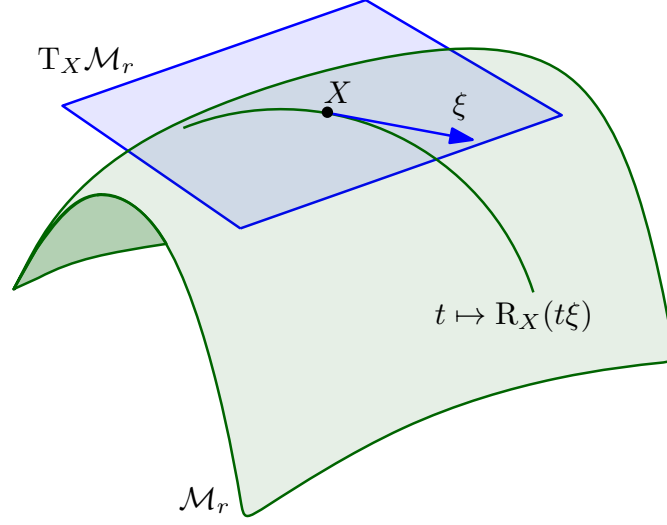


Figure 1.1: Illustration of the retraction-based framework in Riemannian optimization.

Low-rank Riemannian optimization has proven successful in various problem domains, including matrix completion/recovery [Van13; DKM10; GA22; UV15; Tan+14], linear regression [Mey11; Mey11; Mis+14], algebraic Riccati equations [MV14], phase retrieval [HGZ17], relaxation of sparse PCA and maximum-cut problems [Jou+10]. More pertinent to our focus are applications in solving generalized Lyapunov equations [VV10; Van10] and linear systems with tensor-product structure [KSV16]. The primary aim of this thesis is to apply similar techniques to multiterm linear matrix equations.

Similar to standard iterative methods for the solution of linear systems, preconditioning is essential to accelerate the convergence of Riemannian optimization when dealing with ill-conditioned problems. As observed in [KSV16; VV10], the ill-conditioning of the operator $\mathcal{A}$ results in a similarly ill-conditioned Riemannian Hessian. Analogous to optimization on Euclidean spaces, this ill-conditioning hampers the convergence of first-order line-search methods and increases the number of inner iterations required for computing (approximate) Newton-like steps via iterative methods. This highlights the necessity of preconditioning. Two categories of Riemannian preconditioned methods are employed in this work: those relying on the application of the inverse of a Riemannian preconditioner derived from an approximation of the Riemannian Hessian, as seen in [KSV16; VV10; BA15]; and those based on the explicit modification of the manifold's metric, possibly influencing also its parametrization, as explored in, e.g., [MS16; NS12; MS14; KM16]. Our study connects and integrates both approaches, also in combination, to derive effective Riemannian preconditioners from dominant terms in $\mathcal{A}$.

1.2 Contributions and outline of the thesis

In Chapter 1, we introduced the problem of solving multiterm linear matrix equations and presented a brief overview of the fundamental ideas of preconditioned Riemannian

optimization. A literature review on low-rank methods for multiterm linear matrix equation follows, with a focus on Conjugate Gradient (CG) with truncation as our primary competitor, given the assumed positive definiteness of $\mathcal{A}$.

Chapters 2 and 3 form the foundation for preconditioned Riemannian optimization. Chapter 2 introduces fundamental concepts of Riemannian geometry and presents the embedded geometry of fixed-rank matrices. Chapter 3 covers (preconditioned) Riemannian optimization algorithms, drawing on well-established results from [Bou23a; AMS08; VV10; Sat21]. The presentation style and topics chosen are intended to prepare for applying Riemannian optimization to multiterm linear matrix equations. Section 3.4 introduces Riemannian preconditioning, defining preconditioned Riemannian algorithms.

Chapter 4 focuses on applying Riemannian optimization to multiterm linear matrix equations. It defines the cost function, and details computationally efficient calculations for cost, Riemannian gradient, and Riemannian Hessian. The chapter discusses local and global convergence guarantees, emphasizing a result on local quadratic convergence of Riemannian Trust-Region with local model based on the projected Euclidean Hessian. Additionally, it outlines a rank-adaptive algorithm obtained by alternating between fixed-rank Riemannian optimization and rank updates.

Chapter 5 presents our main contributions: computationally efficient methods for applying Riemannian preconditioners derived from dominant terms in $\mathcal{A}$, of the form $\mathcal{P}X = EXD$ or $\mathcal{P}X = AXD + EXB$, where A, B, D, E are symmetric positive definite matrices of size compatible with X . This chapter addresses challenges posed by preconditioners where X is surrounded on both sides by coefficient matrices, compared to the one-sided counterparts $\mathcal{P}X = EX$ and $\mathcal{P}X = AX + XA$ discussed in [VV10; KSV16]. For $\mathcal{P}X = EXD$, we detail the exact inversion of the corresponding Riemannian preconditioner and discuss the geometry of fixed-rank matrices as a Riemannian submanifold of $\mathbb{R}^{m \times n}$ endowed with the metric induced by $\mathcal{P}$. For $\mathcal{P}X = AXD + EXB$, we propose an exact inversion method and an approximation using an Altering-Direction-Implicit-like iteration on the tangent space, when exact inversion is too computationally expensive.

Chapter 6 briefly extends the developed techniques to fixed-rank positive semidefinite matrices.

Chapter 7 evaluates the proposed algorithms through numerical experiments on three representative model problems. It demonstrates that two-sided preconditioners significantly accelerate convergence of Riemannian optimization, making it competitive with established solvers such as CG with truncation and MultiRB.

Finally, Chapter 8 summarizes the thesis, highlighting key contributions and suggesting potential future developments.

1.3 Notational Conventions

We introduce essential notational conventions for use throughout this thesis. Additional conventions will be introduced during the thesis's development, with the primary ones summarized in the section Notation.

1.4. Introduction to low-rank approximations and low-rank methods

The notation $\mathcal{A}$ designates the linear operator (1.3). Matrices are denoted by capital letters, even when considered as points on a manifold. For matrices A and B of the same size, $\langle A, B \rangle = \text{tr}(A^\top B)$ represents the Frobenius inner product. The Frobenius norm is denoted as $\|\cdot\|_F$. If $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, we denote its singular values as $\sigma_1(A) \geq \dots \geq \sigma_n(A)$, with the shorthand $\sigma_1 \geq \dots \geq \sigma_n$ when the matrix is clear from context. For a square matrix $A \in \mathbb{R}^{n \times n}$, $\Lambda(A)$ denotes its spectrum, $\rho(A)$ its spectral radius, $\kappa(A) = \|A\| \|A^{-1}\|$ the condition number with respect to a generic matrix norm $\|\cdot\|$, and $\kappa_2(A)$ the condition number in the 2-norm. If the spectrum of A is real, we use $\lambda_1(A) \geq \dots \geq \lambda_n(A)$ to denote the eigenvalues. Similar notations apply to linear operators.

We denote by $\mathcal{E}$ a finite dimensional Euclidean space, and with $\langle \cdot, \cdot \rangle$ its inner product. If $\mathcal{L} : \mathcal{E} \rightarrow \mathcal{E}$ is a symmetric positive definite linear operator, with respect to $\langle \cdot, \cdot \rangle$, we denote $\langle x, y \rangle_{\mathcal{L}} = \langle \mathcal{L}x, y \rangle$ the inner product induced by $\mathcal{L}$ and $\|x\|_{\mathcal{L}} = \sqrt{\langle x, x \rangle_{\mathcal{L}}}$ the corresponding norm. We denote by $\mathcal{M}$ a smooth embedded submanifold of $\mathcal{E}$. In the following, when referring to a "manifold", we always mean a smooth embedded submanifold. $T_X \mathcal{M}$ represents the tangent space to $\mathcal{M}$ at $X \in \mathcal{M}$, $N_X \mathcal{M}$ the normal space, and $T\mathcal{M}$ the tangent bundle of $\mathcal{M}$. Tangent vectors are typically denoted by the Greek letters ξ and η . We use Proj_X and $\text{Proj}_X^\perp$ to indicate the orthogonal projection to $T_X \mathcal{M}$ and $N_X \mathcal{M}$, respectively. If $\mathcal{M}$ is a Riemannian manifold, $\langle \cdot, \cdot \rangle_X$ denotes the Riemannian metric on $T_X \mathcal{M}$, and $\|\cdot\|_X$ represents the corresponding induced norm. If the Riemannian metric coincides with the metric of the embedding space $\mathcal{E}$, we refer to $\mathcal{M}$ as a *Riemannian submanifold*.

1.4 Introduction to low-rank approximations and low-rank methods

1.4.1 Singular value decomposition and low-rank approximations

One of the fundamental tools we will leverage is the Singular Value Decomposition (SVD). We recall that for a matrix $X \in \mathbb{R}^{m \times n}$, an SVD always exists and is defined as follows.

Definition 1.1 (Singular Value Decomposition [GV13, Theorem 2.4.1]). *Let $X \in \mathbb{R}^{m \times n}$. There exist orthogonal matrices $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, and singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$, where $p = \min\{m, n\}$, such that*

$$X = U \Sigma V^\top, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_p) \in \mathbb{R}^{m \times n}. \quad (1.6)$$

In the definition above, if $\text{rank}(X) = t$, then $\sigma_{t+1} = \dots = \sigma_p = 0$. Thus, using MATLAB's notation one can write

$$X = U(:, 1:t) \text{diag}(\sigma_1, \dots, \sigma_t) V(:, 1:t)^\top. \quad (1.7)$$

1.4. Introduction to low-rank approximations and low-rank methods

We will refer to (1.7) as *thin SVD* or, with a slight misuse of nomenclature, simply as SVD. The SVD enables the computation of the best approximation with rank lower than or equal to r of a matrix, with respect to any unitarily invariant norm.

Theorem 1.2 (Eckhart-Young-Mirsky [HJ12; MIR60]). *Let $X \in \mathbb{R}^{m \times n}$ with SVD $X = U\Sigma V^\top$. Then for any unitarily invariant norm $\|\cdot\|$*

$$\min_{\text{rank}(Y) \leq r} \|X - Y\| = \|X - U_r \Sigma_r V_r^\top\|, \quad (1.8)$$

where U_r and V_r are the matrices obtained from the first r columns of U and V , respectively, and Σ_r denotes the leading $r \times r$ submatrix of Σ . Furthermore, for the 2-norm it holds $\|X - U_r \Sigma_r V_r^\top\|_2 = \sigma_{r+1}$, and for the Frobenius norm $\|X - U_r \Sigma_r V_r^\top\|_F = \sqrt{\sigma_{r+1}^2 + \dots + \sigma_p^2}$, $p = \min\{m, n\}$.

The truncation at rank r of the SVD of a matrix is referred to as the rank- r Truncated Singular Value Decomposition (TSVD). It is noteworthy that the matrix $X_r = U_r \Sigma_r V_r^\top$ attains the minimum (1.8), but it is not the unique minimizer if $\sigma_r = \sigma_{r+1} > 0$. Furthermore, the best approximation of rank *exactly* r coincides with (1.8) if $\text{rank}(X) > r$ and is not well-defined if $\text{rank}(X) < r$.

1.4.2 Low-rank methods for multiterm linear matrix equations

The effectiveness of low-rank methods for multiterm linear matrix equations depends on the low-rank approximability of the solution $X_\star$, i.e., the property that $X_\star$ as singular values decaying rapidly to zero. Concerning standard Sylvester and Lyapunov equations, results on the low-rank approximability of the solution are available when the right-hand side F has low rank [BT19; TW18; MPR18; BES15; ASZ02; Pen00a]. In the realm of multiterm linear matrix equations, low-rank approximability is frequently observed in practice when F has low rank, but existing theoretical results rely on restrictive conditions on the coefficient matrices [BB13; Jar+18]. In the subsequent discussion, we *assume* that the solution $X_\star$ to (1.1) is low-rank approximable, where the definition of "low" depends heavily on the specific example. We now illustrate some low-rank methods in the literature for solving multiterm linear matrix equations.

1.4.2.1 Matrix-oriented applications of standard iterative methods

The multiterm matrix equation (1.1) can be equivalently reformulated as

$$\mathcal{K} \text{vec}(X) = \text{vec}(F), \quad \mathcal{K} := \left(\sum_{i=1}^{\ell} B_i \otimes A_i \right) \in \mathbb{R}^{mn \times mn}, \quad (1.9)$$

where $\text{vec} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{mn}$ is the vectorization operator, i.e., it stacks the columns of a matrix, and $B_i \otimes A_i$ is the Kronecker product of B_i and A_i . As noted already in [Lan70], this transformation to a matrix-vector equation allows the use of standard iterative

1.4. Introduction to low-rank approximations and low-rank methods

methods for linear systems. Indeed, the Kronecker structure of $\mathcal{K}$ can be exploited to perform matrix-vector products with $\mathcal{K}$ without explicitly assembling $\mathcal{K}$, only using matrix-matrix products with the A_i 's and the B_i 's. However, a first limitation of this approach is the $\mathcal{O}(mn)$ memory requirement needed to store the vectors in $\mathbb{R}^{mn}$, i.e., the vectorization of $m \times n$ matrices, which becomes unfeasible for large-scale problems. To (partially) address these issues, given the assumed low-rank approximability of the solution, a frequently employed technique is to keep iterates and auxiliary matrices in low-rank format and apply low-rank truncation after operations that increase the rank.

Given that we restricted our study to the positive definite case, we present a low-rank version of the Conjugate Gradient (CG) method [HS52], possibly preconditioned, as introduced in [KT11; BB13; KPT14]. The CG with truncation algorithm is outlined in Algorithm 1. Aside from the truncation operations, which will be detailed later, it follows the standard CG method. The key difference lies in keeping the iterates X_k and auxiliary vectors R_k , P_k , Z_k , and Q_k in low-rank format. For instance, if $\mathcal{A}$ is the operator (1.3) and is applied to $X = Y_X W_X^\top$ with $Y_X \in \mathbb{R}^{m \times r_X}$, $W_X \in \mathbb{R}^{n \times r_X}$, the result is

$$\mathcal{A}X = \begin{bmatrix} A_1 Y_X & \cdots & A_\ell Y_X \end{bmatrix} \begin{bmatrix} B_1 W_X & \cdots & B_\ell W_X \end{bmatrix}^\top,$$

which is kept in factored format. Similarly, inner products and linear combinations of X and $Z = Y_Z W_Z^\top$ with $Y_Z \in \mathbb{R}^{m \times r_Z}$, $W_Z \in \mathbb{R}^{n \times r_Z}$ are computed as

$$\begin{aligned} \langle X, Z \rangle &= \text{tr} \left((Y_X W_X^\top)^\top (Y_Z W_Z) \right) = \text{tr} \left((W_Z^\top W_X) (Y_X^\top Y_Z) \right), \\ \alpha X + \beta Z &= \begin{bmatrix} \alpha Y_X & \beta Y_Z \end{bmatrix} \begin{bmatrix} W_X & W_Z \end{bmatrix}^\top. \end{aligned}$$

The truncation operator $\mathcal{T}$ compresses the factorization of a matrix $X = YW^\top$ with $Y \in \mathbb{R}^{m \times r_X}$, $W \in \mathbb{R}^{n \times r_X}$ into $\tilde{X} = \tilde{Y}\tilde{W}^\top$ with $\tilde{Y} \in \mathbb{R}^{m \times \tilde{r}}$, $\tilde{W} \in \mathbb{R}^{n \times \tilde{r}}$, and $\tilde{r} \leq r_X$. The compression ensures that the error $\|X - \tilde{X}\|_F$ satisfies either an absolute bound $\|X - \tilde{X}\|_F \leq \epsilon_{\text{abs}}$ or a relative one $\|X - \tilde{X}\|_F \leq \epsilon_{\text{rel}}\|X\|_F$. For this purpose a TSVD is employed. We compute the thin QR factorizations $Y = Q_Y R_Y$ and $W = Q_W R_W$, followed by a singular value decomposition of the $r_X \times r_X$ matrix $R_Y R_W^\top = U \text{diag}(\sigma_1, \dots, \sigma_{r_X}) V^\top$. The truncation rank $\tilde{r}$ is determined as the smallest integer satisfying

$$\sqrt{\sigma_{\tilde{r}+1}^2 + \dots + \sigma_{r_X}^2} \leq \epsilon_{\text{abs}} \quad \text{to ensure } \|X - \tilde{X}\|_F \leq \epsilon_{\text{abs}}, \text{ or,} \quad (1.10)$$

$$\sqrt{\sigma_{\tilde{r}+1}^2 + \dots + \sigma_{r_X}^2} \leq \epsilon_{\text{rel}} \sqrt{\sigma_1^2 + \dots + \sigma_{r_X}^2} \quad \text{to ensure } \|X - \tilde{X}\|_F \leq \epsilon_{\text{rel}}\|X\|_F. \quad (1.11)$$

Using MATLAB's notation, $\tilde{Y}$ and $\tilde{W}$ are computed as

$$\tilde{Y} = Q_Y U(:, 1 : \tilde{r}) \text{diag}(\sqrt{\sigma_1}, \dots, \sqrt{\sigma_{\tilde{r}}}), \quad \tilde{W} = Q_W V(:, 1 : \tilde{r}) \text{diag}(\sqrt{\sigma_1}, \dots, \sqrt{\sigma_{\tilde{r}}}).$$

The overall cost is $\mathcal{O}((m+n)r_X^2)$ flops, primarily dominated by the cost of computing the two QR factorizations.

1.4. Introduction to low-rank approximations and low-rank methods

Algorithm 1 Preconditioned Conjugate Gradient with truncation

Input: Positive definite linear operators $\mathcal{A}, \mathcal{P} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ acting on matrices in low-rank format, right-hand side F in low-rank format, truncation operator $\mathcal{T}$.

Parameters: Tolerance $\text{tol} > 0$, parameters of the truncation operator.

Output: Matrix $X \in \mathbb{R}^{m \times n}$ in low-rank format, satisfying $\|\mathcal{A}X - F\| \leq \text{tol}$

```

1: Set  $X_0 = 0$ ,  $R_0 = F$ ,  $Z_0 = \mathcal{P}^{-1}R_0$ ,  $P_0 = Z_0$ ,  $Q_0 = \mathcal{A}P_0$ 
2:  $\alpha_0 = \langle P_0, Q_0 \rangle$ ,  $k = 0$ 
3: while  $\|R_k\| > \text{tol}$  do
4:    $\omega_k = \langle R_k, P_k \rangle / \alpha_k$ 
5:    $X_{k+1} = X_k + \omega_k P_k$             $X_{k+1} \leftarrow \mathcal{T}(X_{k+1})$ 
6:    $R_{k+1} = B - \mathcal{A}X_{k+1}$          Optionally:  $R_{k+1} \leftarrow \mathcal{T}(R_{k+1})$ 
7:    $Z_{k+1} = \mathcal{P}^{-1}R_{k+1}$ 
8:    $\beta_k = -\langle Z_{k+1}, Q_k \rangle / \alpha_k$ 
9:    $P_{k+1} = Z_{k+1} + \beta_k P_k$         $P_{k+1} \leftarrow \mathcal{T}(P_{k+1})$ 
10:   $Q_{k+1} = \mathcal{A}P_{k+1}$            Optionally:  $Q_{k+1} \leftarrow \mathcal{T}(Q_{k+1})$ 
11:   $\alpha_{k+1} = \langle P_{k+1}, Q_{k+1} \rangle$ 
12:   $k = k + 1$ 
13: end while
14:  $X = X_k$ 

```

For truncation of the iterates X_k , a relative criterion is typically employed. However, as noted in [KPT14], using only a relative criterion also for truncating the residual would lead to unnecessarily high ranks in the last steps of CG, when the residual is expected to be very small. Therefore, it is preferable to use a mixed criterion, truncating $\tilde{R}_k = \mathcal{T}(R_k)$ to ensure

$$\|R_k - \tilde{R}_k\|_F \leq \max\{\epsilon_{\text{abs}} \cdot \|R_0\|_F, \epsilon_{\text{rel}} \cdot \|R_k\|_F\} \quad (1.12)$$

for user-defined tolerances ϵ_{abs} , ϵ_{rel} . Similar truncation criteria are also adopted for truncation of P_k and Q_k .

Truncation operations are arguably the most expensive part of an iteration of CG with truncation. As observed in [KS15], there is evidence [KT11] that an effective preconditioner is needed not only to improve convergence speed, as for the standard CG method, but also to avoid excessive growth of the rank of intermediate iterates, thus reducing the per-iteration computational effort.

In the case where $\mathcal{A}$ is not SPD, similar principles can be used to formulate matrix-oriented versions of various standard iterative methods for linear systems. Notably, these methods include BiCGStab [KT11; BB13] MINRES [SB15], GMRES and FOM [PK21].

1.4.2.2 Other methods

Projective methods Another class of solvers for multiterm linear matrix equations is obtained by generalizing the so-called *projective methods*, originally designed for standard Sylvester and Lyapunov equations [Sim16, § 4.4.1 and § 5.2.1]. In these methods, a sequence of subspaces $\mathcal{V}_k \subset \mathbb{R}^m$ and $\mathcal{W}_k \subset \mathbb{R}^n$ is constructed. Taking $V_k \in \mathbb{R}^{m \times p_v}$

1.4. Introduction to low-rank approximations and low-rank methods

and $W_k \in \mathbb{R}^{n \times p_w}$ whose columns form an orthonormal basis of $\mathcal{V}_k$ and $\mathcal{W}_k$, respectively, an approximate solution of the form $X_k = V_k Y_k W_k^\top$ is sought. The $p_v \times p_w$ matrix Y_k , often referred to as the *reduced solution matrix*, is determined by imposing that the residual $R_k := \mathcal{A}X_k - F$ satisfies the Galerkin orthogonality condition $V_k^\top R_k W_k = 0$. This results in the projected multiterm linear matrix equation

$$\sum_{i=1}^{\ell} (V_k^\top A_i V_k) Y_k (W_k^\top B_i W_k)^\top = (V_k^\top F_L) (W_k^\top F_R)^\top,$$

where the coefficient matrices have a reduced size $p_v \times p_w$, but are dense rather than sparse. Note that if the size of the matrices is strongly unbalanced, one can project on one side only, e.g., if $m \gg n$, one can set $W_k = I_n$ for all k , as done in [PSS17]. To solve the projected multiterm matrix equation, direct or iterative methods can be applied, depending on the size.

The critical aspect of projective methods is how to generate the spaces $\mathcal{V}_k$ and $\mathcal{W}_k$ so that X_k is a good approximation of the solution, without the dimensions p_v, p_w of the spaces growing too large, and how to compute numerically an orthonormal basis for them. In the case of standard Sylvester and Lyapunov equations, among the most common choices are extended Krylov subspaces [Sim07; KS11b] and rational Krylov subspaces [DKS11; DS11]. Generalizing these spaces to the multiterm case is not straightforward, as one could potentially generate extended or rational Krylov spaces with any of the matrices $A_1, \dots, A_\ell$ and $B_1, \dots, B_\ell$, or even combine them. However, it is crucial that the dimension does not grow too much. For examples of the application of projective methods to multiterm equations, along with the corresponding generation of the subspaces $\mathcal{V}_k$ and $\mathcal{W}_k$, we refer the reader to [PSS17; Jar+18].

Fixed-point iterations and related methods For generalized Lyapunov equations, assuming the existence of a convergent splitting of the multiterm Lyapunov operator, a fixed-point iteration has been proposed in [Dam08], and a closely related bilinear Alternating Direction Implicit (ADI) iteration has been introduced in [BB13]. Another approach, discussed in [SSS16], combines a fixed-point iteration with projective methods. Similar techniques could also be employed in solving multiterm Sylvester equations. However, these methods suffer from the limitation of relying on the existence of a convergent splitting, a condition that often lacks in examples of multiterm linear matrix equations.

Methods based on alternating optimization Another category comprehends methods based on alternating optimization techniques, where the factors $Y \in \mathbb{R}^{m \times r}$ and $W \in \mathbb{R}^{n \times r}$ in the $X = YW^\top$ low-rank decomposition are obtained by alternately optimizing for one and then the other. These methods are particularly suitable when the error function to be minimized is convex in each of the two factors V and W , but not jointly convex. Such techniques are known as Alternating Linear Schemes (ALS)

1.4. Introduction to low-rank approximations and low-rank methods

or Alternating Energy Minimization (AEM) methods, depending on the type of norm being minimized. Fixed-rank versions have been applied to matrix completion problems in [HH09; JNS12]. Rank-adaptive versions, based on successive rank-one updates of the approximate solution, are integral part of the Proper Generalized Decomposition (PGD) method for solving parametrized PDEs [Nou08; Nou10; TLN14]. Addressing general multiterm matrix equations, a low-rank method based on greedy rank-one updates via ALS and Galerkin projection was proposed in [KS15]. This approach was later extended in [BR19] with a focus on the case of generalized Lyapunov equations. Additionally, an AEM method for multiterm matrix equations was introduced in [Lee+20], concentrating on the positive definite case and proposing efficient procedures to enhance the accuracy of the solution through rank-one updates.

Sherman-Morrison-Woodbury formula Finally, we mention that, a matrix-oriented version of the Sherman-Morrison-Woodbury formula has been employed in [HS21].

1.4.2.3 Advantages and drawbacks of low-rank Riemannian optimization

As anticipated, the herein employed methods are based on Riemannian optimization techniques. An approach based on fixed-rank Riemannian optimization offers the advantage of inherently preventing the growth of intermediate ranks, a phenomenon often encountered in CG with truncation. Moreover, the approximate solution is updated solely as a point of the manifold $\mathcal{M}_r$, irrespective of its representation and storage in memory in factorized form. This is in contrast to the alternating optimization of the low-rank decomposition factors in ALS/AEM and the separate generation of orthogonal bases for the Span of the rows and columns in projective methods. However, determining the rank r a priori may not be straightforward. This limitation can be addressed, at least partially, with rank-adaptive strategies that alternate between fixed-rank optimization and rank updates [GA22; Zho+16; UV15].

Chapter 2

Riemannian optimization over fixed-rank matrices

As mentioned earlier, we will recast the approximate resolution of (1.1) as an optimization problem on a manifold, i.e., a problem of the form $\min_{X \in \mathcal{M}} f(X)$, where $\mathcal{M}$ is a smooth manifold and $f : \mathcal{M} \rightarrow \mathbb{R}$ is a smooth cost function. The main focus of this chapter is to introduce essential tools from Riemannian geometry that prove useful for optimization on manifolds, drawing mainly from N. Boumal's book [Bou23a]. Following this, we will delve into the Riemannian structure of fixed-rank matrices, the main manifold for the problem at hand.

Throughout this chapter, we assume the reader is familiar with the basics of Riemannian geometry. Regarding the results known in the literature, we will provide proofs only for those necessary for the development and understanding of the thesis, supplying references for the rest.

2.1 Basic tools for Riemannian optimization

2.1.1 Riemannian gradient and Hessian

The first and foremost ingredient of gradient-based optimization methods is the notion of gradient. Moreover, second-order methods, such as Newton's method or the Trust-Region method, require the notion of Hessian. As a first step towards optimization on manifolds, these notions need to be adapted to the Riemannian setting. Before delving into that, let us recall the definition of smooth extension of a smooth map between manifolds.

Definition 2.1 (Smooth extension). *Let $\mathcal{M}$ and $\mathcal{N}$ be embedded submanifolds of the Euclidean spaces $\mathcal{E}$ and $\mathcal{F}$, respectively. Let $f : \mathcal{M} \rightarrow \mathcal{N}$ be a smooth map. A smooth extension of f is a smooth map $\bar{f} : \mathcal{U} \rightarrow \mathcal{F}$ such that $\mathcal{U}$ is a neighborhood of $\mathcal{M}$ in $\mathcal{E}$ and $f(X) = \bar{f}(X)$ for all $X \in \mathcal{M}$. We also write $f = \bar{f}|_{\mathcal{M}}$.*

2.1. Basic tools for Riemannian optimization

Proposition 2.2 ([Bou23a, Proposition 3.31]). *A map $f : \mathcal{M} \rightarrow \mathcal{N}$ between embedded submanifolds is smooth if and only if it admits a smooth extension.*

2.1.1.1 Riemannian gradient

We start by recalling the definition of Riemannian gradient, with a particular focus on Riemannian submanifolds.

Definition 2.3 (Riemannian gradient). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian manifold $\mathcal{M}$. The Riemannian gradient of f is the unique vector field $\text{grad} f$ such that*

$$Df(X)[\xi] = \langle \text{grad} f(X), \xi \rangle_X \quad \forall (X, \xi) \in T\mathcal{M}.$$

Riemannian submanifolds We recall that a Riemannian submanifold of a Euclidean space is a Riemannian manifold where the Riemannian metric matches the metric of the ambient space. In case $\mathcal{M}$ is a Riemannian submanifold, the Riemannian gradient can be computed from the Euclidean gradient of any smooth extension.

Proposition 2.4 ([Bou23a, Proposition 3.61]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian submanifold $\mathcal{M}$ of a Euclidean space $\mathcal{E}$. The Riemannian gradient of f is*

$$\text{grad} f(X) = \text{Proj}_X \left(\text{grad} \bar{f}(X) \right) \quad (2.1)$$

where $\bar{f}$ is any smooth extension of f , and $\text{Proj}_X : \mathcal{E} \rightarrow T_X \mathcal{M}$ is the orthogonal projection to the tangent space at X .

Proof. By definition of Riemannian gradient of f and Euclidean gradient of $\bar{f}$, for all $\xi \in T_X \mathcal{M}$ it holds:

$$\langle \text{grad} f(X), \xi \rangle = Df(X)[\xi] = D\bar{f}(X)[\xi] = \langle \text{grad} \bar{f}(X), \xi \rangle = \langle \text{Proj}_X(\text{grad} \bar{f}(X)), \xi \rangle,$$

where in the last equality we crucially used that $\xi \in T_X \mathcal{M}$. The thesis follows by uniqueness of the Riemannian gradient. $\square$

2.1.1.2 Riemannian Hessian

Definition 2.5 (Riemannian Hessian). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian manifold $\mathcal{M}$. The Riemannian Hessian of f at $X \in \mathcal{M}$ is the linear map $\text{Hess} f(X) : T_X \mathcal{M} \rightarrow T_X \mathcal{M}$ defined as*

$$\text{Hess} f(X)[\xi] = \nabla_\xi \text{grad} f \quad \forall \xi \in T_X \mathcal{M},$$

where ∇ is the Levi-Civita connection.

Remarkably, and analogously to the Euclidean case, the Riemannian Hessian is a symmetric operator, with respect to the Riemannian metric [Bou23a, Proposition 5.15].

2.1. Basic tools for Riemannian optimization

Riemannian submanifolds In case $\mathcal{M}$ is a Riemannian submanifold, the Riemannian Hessian of f is obtained by projecting to the tangent the differential of a smooth extension of $\text{grad}f$. However, this does not coincide with the composition of projection to the tangent with the Euclidean Hessian of $\bar{f}$: we need to add a correction term which comes from the Fréchet differential of the projection operator in (2.1). This is formalized by the following two prepositions.

Proposition 2.6 ([Bou23a, Corollary 5.16]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian submanifold $\mathcal{M}$ of a Euclidean space $\mathcal{E}$. Let $\bar{G}$ be a smooth extension of $\text{grad}f$. Then the Riemannian Hessian of f at $X \in \mathcal{M}$ is*

$$\text{Hess}f(X)[\xi] = \text{Proj}_X \left(D\bar{G}(X)[\xi] \right) \quad \forall \xi \in T_X \mathcal{M}. \quad (2.2)$$

Proposition 2.7 ([Bou23a, Corollary 5.47]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian submanifold $\mathcal{M}$ of a Euclidean space $\mathcal{E}$. Let $\bar{f}$ be any smooth extension of f . Then the Riemannian Hessian of f at $X \in \mathcal{M}$ is*

$$\begin{aligned} \text{Hess}f(X)[\xi] &= \text{Proj}_X \left(\text{Hess}\bar{f}(X)[\xi] \right) + \text{Proj}_X \left(\mathcal{D}_\xi(\text{grad}\bar{f}(X)) \right) \\ &= \text{Proj}_X \left(\text{Hess}\bar{f}(X)[\xi] \right) + \mathcal{D}_\xi \left(\text{Proj}_X^\perp(\text{grad}\bar{f}(X)) \right), \end{aligned} \quad (2.3)$$

for all $\xi \in T_X \mathcal{M}$, where $\mathcal{D}_\xi$ is the differential of $X \mapsto \text{Proj}_X$ at X along ξ .

Proof. It can be shown, e.g., using orthonormal local frames, that the map $X \mapsto \text{Proj}_X$ is a smooth map from $\mathcal{M}$ to $\mathcal{L}(\mathcal{E}, \mathcal{E})$, the linear space of linear maps from $\mathcal{E}$ into itself. Let c be any smooth curve on $\mathcal{M}$ such that $c(0) = X$ and $c'(0) = \xi$. Denoting as $\frac{D}{dt}$ the covariant derivative induced by the Levi-Civita connection ∇ :

$$\begin{aligned} \text{Hess}f(X)[\xi] &= \nabla_\xi \text{grad}f(X) = \nabla_{c'(0)} \text{grad}f(c(0)) = \left. \frac{D}{dt} \text{grad}f(c(t)) \right|_{t=0} \\ &= \text{Proj}_X \left(\left. \frac{d}{dt} \text{Proj}_{c(t)}(\text{grad}\bar{f}(c(t))) \right|_{t=0} \right) \\ &= \text{Proj}_X \left(\left. \frac{d}{dt} \text{Proj}_{c(t)} \right|_{t=0} (\text{grad}\bar{f}(X)) \right) + \text{Proj}_X \left(\text{Proj}_X \left(\left. \frac{d}{dt} \text{grad}\bar{f}(c(t)) \right|_{t=0} \right) \right) \\ &= \text{Proj}_X \left(\mathcal{D}_\xi(\text{grad}\bar{f}(X)) \right) + \text{Proj}_X(\text{Hess}\bar{f}(X)[\xi]). \end{aligned}$$

To conclude, observe that differentiating the identity $\text{Proj}_{c(t)} = \text{Proj}_{c(t)} \circ \text{Proj}_{c(t)}$ and evaluating at $t = 0$ yields $\mathcal{D}_\xi = \text{Proj}_X \circ \mathcal{D}_\xi + \mathcal{D}_\xi \circ \text{Proj}_X$, which can be reorganized as $\text{Proj}_X \circ \mathcal{D}_\xi = \mathcal{D}_\xi \circ \text{Proj}_X^\perp$. $\square$

Remark 2.8. *Intuitively, the differential of the projection operator measures how the tangent spaces of $\mathcal{M}_r$ "bends" in the embedding space. Indeed, the term $\mathcal{D}_\xi(\text{Proj}_X^\perp(\text{grad}\bar{f}(X)))$ can be related to the curvature of the manifold; for a more precise statement, refer to [Abs+09, § 6]. We will frequently denote $\mathcal{D}_\xi(\text{Proj}_X^\perp(\text{grad}\bar{f}(X)))$ as the curvature term in the Riemannian Hessian. Additionally, note that this term depends solely on the normal component of $\text{grad}\bar{f}(X)$.*

2.1.2 Moving along curves and necessary optimality conditions

Gradient descent on Euclidean spaces performs updates $X_+ = X + \alpha v$, where $v = -\text{grad}f(X)$. On manifolds, updates of this form are not possible because the gradient at X is a tangent vector $\xi \in T_X\mathcal{M}$ and the sum $X + \alpha\xi$ is not a point on the manifold, in the general case. Instead of moving along straight lines, the idea will be to move along curves. It is therefore natural to question how the value of the objective function varies along curves, at least locally. A Taylor expansion suits our needs.

Proposition 2.9 ([Bou23a, § 5.9]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian manifold $\mathcal{M}$. Let $c : I \subseteq \mathbb{R} \rightarrow \mathcal{M}$ be a smooth curve on $\mathcal{M}$ such that $c(0) = X$ and $c'(0) = \xi$. Denote by $\frac{D}{dt}$ the covariant derivative induced by the Levi-Civita connection ∇ and let $c''(0) = \frac{D}{dt}c'(t)\big|_{t=0}$ be the intrinsic acceleration of c at 0. Then it holds*

$$f(c(t)) = f(X) + t \langle \text{grad}f(X), \xi \rangle_X + \mathcal{O}(t^2) \quad (2.4)$$

$$\begin{aligned} &= f(X) + t \langle \text{grad}f(X), \xi \rangle_X + \frac{t^2}{2} \langle \text{Hess}f(X)[\xi], \xi \rangle_X + \\ &\quad + \frac{t^2}{2} \langle \text{grad}f(X), c''(0) \rangle_X + \mathcal{O}(t^3) \end{aligned} \quad (2.5)$$

By moving along curves, we can also define first- and second-order critical points of a smooth map. Analogously to the Euclidean setting, being a critical point is necessary condition for being a local minimizer, and by imposing additional second-order requirements one can obtain sufficient conditions for local optimality. This is summarized in the following definition and propositions.

Definition 2.10. *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on an embedded submanifold $\mathcal{M}$. A point $X \in \mathcal{M}$ is critical (or first-order critical) for f if $(f \circ c)'(0) = 0$ for any smooth curve c on $\mathcal{M}$ such that $c(0) = X$.*

Furthermore, $X \in \mathcal{M}$ is second-order critical for f if it is critical for f and $(f \circ c)''(0) \geq 0$ for any smooth curve c on $\mathcal{M}$ such that $c(0) = X$, and it is strictly second-order critical if it also holds $(f \circ c)''(0) > 0$ for any smooth curve c on $\mathcal{M}$ such that $c(0) = X$ and $c'(0) \neq 0$.

Proposition 2.11 ([Bou23a, Proposition 6.2]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on an embedded submanifold $\mathcal{M}$. Any local minimizer of f is a second-order critical point of f . In particular, it is also a first-order critical point.*

Proposition 2.12 ([Bou23a, Proposition 6.5]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on an embedded submanifold $\mathcal{M}$. Any strict second-order critical point of f is a strict local minimizer of f .*

Remark 2.13. *Due to the result in Proposition 2.12, strict second-order critical points are often referred as non-degenerate local minimizers. We will also often adopt this nomenclature.*

2.1. Basic tools for Riemannian optimization

The notion of a critical point is formulated on generic manifolds, without requiring a metric. However, on Riemannian manifolds, the Taylor expansion allows us to characterize critical points in terms of the Riemannian gradient and Hessian, as summarized in the following proposition.

Proposition 2.14 ([Bou23a, Proposition 6.3, 6.6]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian manifold $\mathcal{M}$. Then $X \in \mathcal{M}$ is a first-order critical point of f if and only if $\text{grad}f(X) = 0$. Moreover, $X \in \mathcal{M}$ is a (strict) second-order critical point of f if and only if $\text{grad}f(X) = 0$ and $\text{Hess}f(X) \succeq 0$ ($\text{Hess}f(X) \succ 0$).*

2.1.3 Retractions

In the same spirit as in the previous section, we want to generalize updates of the type $X_+ = X + \alpha\xi$, $\xi \in T_X\mathcal{M}$, by moving along curves. Instead of performing one step of length α along the straight line $t \mapsto X + t\xi$ originating from X with velocity ξ , we can move along a curve $c : \mathbb{R} \rightarrow \mathcal{M}$ such that $c(0) = X$ and $c'(0) = \xi$. However, there may be many curves that satisfy this property. Defining a retraction means choosing a particular curve for each pair (X, ξ) , in a way that varies smoothly with (X, ξ) .

Definition 2.15 (Retraction). *Let $\mathcal{M}$ be an embedded submanifold. A retraction is a map*

$$R : T\mathcal{M} \rightarrow \mathcal{M}, \quad R(X, \xi) = R_X(\xi)$$

such that:

- i. there exists $\mathcal{U}$ open subset of $T\mathcal{M}$ such that $R : \mathcal{U} \rightarrow \mathcal{M}$ is smooth and $(X, 0) \in \mathcal{U}$ for all $X \in \mathcal{M}$;*
- ii. for every $(X, \xi) \in T\mathcal{M}$ the curve $c(t) = R_X(t\xi)$ satisfies $c(0) = X$ and $c'(0) = \xi$.*

Remark 2.16. *In the previous definition, the first requirement allows the retraction to be defined only locally, for tangent vectors sufficiently close to zero, which is typically enough for optimization purposes, as gradient-based steps should not be too large. The second requirement is equivalent to $R_X(0) = X$ and $DR_X(0) = \text{id}_{T_X\mathcal{M}}$, and it is often referred to as local rigidity.*

As already mentioned, retraction-based optimization algorithms perform steps of the form $X_+ = R_X(\alpha\xi)$, where $\xi \in T_X\mathcal{M}$ is a tangent vector. Thus, we are interested in investigating the behavior of the objective function f along particular curves: those obtained by retraction as $c(t) = R_X(t\xi)$. We start by studying the first-order Taylor expansion of the *pullback* by R of f to the tangent space at X

$$\hat{f}_X := f \circ R_X : T_X\mathcal{M} \rightarrow \mathbb{R}. \quad (2.6)$$

Observe that $\hat{f}_X$ is defined on a linear space, and it is smooth around $0 \in T_X\mathcal{M}$.

2.1. Basic tools for Riemannian optimization

Proposition 2.17 ([Bou23a, Proposition 3.59]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian manifold $\mathcal{M}$, and let R be a retraction on $\mathcal{M}$. Then, for all $X \in \mathcal{M}$*

$$\text{grad} \hat{f}_X(0) = \text{grad}(f \circ R_X)(0) = \text{grad} f(X) \quad (2.7)$$

and

$$\hat{f}_X(\xi) = f(R_X(\xi)) = f(X) + \langle \text{grad} f(X), \xi \rangle_X + \mathcal{O}(\|\xi\|_X^2). \quad (2.8)$$

Proof. It suffices to observe that $D(f \circ R_X)(0)[\xi] = Df(R_X(0))[DR_X(0)[\xi]] = Df(X)[\xi]$ and use the definitions of Euclidean gradient for $\hat{f}_X$ and Riemannian gradient for f . $\square$

We can now move to a second-order expansion of $\hat{f}_X$. However, in the general case, $\text{Hess} \hat{f}_X(0) \neq \text{Hess} f(X)$. Indeed, the Taylor expansion of $\hat{f}_X(t\xi) = f(R_X(t\xi))$ in (2.5) contains an extra term, involving $c''(0)$, which has order t^2 and depends on the retraction. This motivates the definition of second-order retractions.

Definition 2.18 (Second order retraction). *A second-order retraction R defined on a Riemannian manifold $\mathcal{M}$ is a retraction for which, given any $(X, \xi) \in T\mathcal{M}$, the curve $c(t) = R_X(t\xi)$ satisfies $c''(0) = 0$.*

Proposition 2.19 ([Bou23a, Proposition 5.44]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian manifold $\mathcal{M}$, and let R be a retraction on $\mathcal{M}$. If X is a critical point of f , then*

$$\hat{f}_X(\xi) = f(R_X(\xi)) = f(X) + \frac{1}{2} \langle \text{Hess} f(X)[\xi], \xi \rangle_X + \mathcal{O}(\|\xi\|_X^3). \quad (2.9)$$

If R is second-order, then for every $X \in \mathcal{M}$ it holds

$$\hat{f}_X(\xi) = f(R_X(\xi)) = f(X) + \langle \text{grad} f(X), \xi \rangle_X + \frac{1}{2} \langle \text{Hess} f(X)[\xi], \xi \rangle_X + \mathcal{O}(\|\xi\|_X^3). \quad (2.10)$$

Proposition 2.20 ([Bou23a, Proposition 5.45]). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map on a Riemannian manifold $\mathcal{M}$, and let R be a retraction on $\mathcal{M}$. If the retraction is second order or if X is a critical point of f*

$$\text{Hess} \hat{f}_X(0) = \text{Hess}(f \circ R_X)(0) = \text{Hess} f(X). \quad (2.11)$$

When we introduced the idea of retraction, our intention was to generalize steps along straight lines. The generalization of straight lines to manifolds are the geodesics. It is therefore natural to ask what is the relationship between retractions and the Riemannian exponential map. It can be shown that retractions are those maps that agree with the Riemannian exponential up to and including the first order [AM12, Proposition 2]. In particular, the Riemannian exponential is a retraction. Moreover second-order retractions are those maps that agree with the Riemannian exponential up to and including the second order [AM12, Proposition 3].

Metric projection retractions and Riemannian submanifolds In a sense, retractions map back tangent vectors onto the manifold. For an embedded submanifold

2.1. Basic tools for Riemannian optimization

$\mathcal{M}$ of a Euclidean space $\mathcal{E}$, a natural way to associate a point in $\mathcal{E}$ with a point on the manifold $\mathcal{M}$ is through the *metric projection*, i.e., taking the closest point on $\mathcal{M}$ with respect to the norm induced by the metric of $\mathcal{E}$. One may wonder whether metric projection is a suitable method for retracting tangent vectors. It turns out that metric projection, on its domain of definition, actually defines a retraction. In addition, such a retraction is second-order if $\mathcal{M}$ is a Riemannian submanifold. We formalize this in the next definitions and propositions. For a proof, we refer the reader to [Bou23a, § 5.12] and the references therein.

Proposition 2.21. *Let $\mathcal{M}$ be an embedded submanifold of a Euclidean space $\mathcal{E}$. For a given $Y \in \mathcal{E}$ the metric projection of Y to $\mathcal{M}$ is the (possibly empty) set*

$$\text{Proj}_{\mathcal{M}}(Y) = \{X \in \mathcal{M} : \|X - Y\| = \inf_{Z \in \mathcal{M}} \|Z - Y\|\}.$$

Let $\Omega \subseteq \mathcal{E}$ be the interior of the domain where $\text{Proj}_{\mathcal{M}}$ is single-valued. Then

- i. Ω is open, not empty and $\mathcal{M} \subset \Omega$;
- ii. The restriction $\text{Proj}_{\mathcal{M}} : \Omega \rightarrow \mathcal{M}$ is smooth and $D\text{Proj}_{\mathcal{M}}(X) = \text{Proj}_X$ for all $X \in \mathcal{M}$.

Proposition 2.22 (Metric projection retraction). *With the same notation as in Proposition 2.21, let $\mathcal{O}$ be the subset of TM defined as*

$$\mathcal{O} = \{(X, \xi) \in \text{TM} : X + \xi \in \Omega\}.$$

Then, $\mathcal{O}$ is open in TM , it contains all pairs $(X, 0) \in \text{TM}$ and the metric projection induces a retraction on $\mathcal{O}$ as

$$\text{R} : \mathcal{O} \rightarrow \mathcal{M}, \quad \text{R}(X, \xi) = \text{R}_X(\xi) = \text{Proj}_{\mathcal{M}}(X + \xi). \quad (2.12)$$

This is called *metric projection retraction*.

If $\mathcal{M}$ is a Riemannian submanifold of $\mathcal{E}$, the metric projection retraction is second-order.

2.1.4 Transporters

Suppose that we want to generalize a method such as a nonlinear conjugate gradient, where the search direction at the k -th step is defined by $\xi_k = -\text{grad}f(X_k) + \beta_k \xi_{k-1}$. The first obstacle that we encounter in a Riemannian setting is that $\xi_{k-1} \in \text{T}_{X_{k-1}}\mathcal{M}$ lies in a different tangent space from ξ_k and $\text{grad}f(X_k)$: we need to somehow *transport* it to the tangent space at X_k , in a meaningful and smooth way. From a purely theoretical point of view, the most suitable tool would be *parallel transport* (also called *parallel translation*) [Bou23a, § 10.3]. However, computing the parallel transport is too expensive in practice, and less costly versions are needed. For this reason, the notion of *transporter* was introduced [HGA15; Bou23a].

2.2. Embedded geometry of fixed-rank matrices

Definition 2.23 (Transporter). *Let $\mathcal{M}$ be an embedded submanifold. The disjoint union of linear maps between the tangent spaces of $\mathcal{M}$*

$$\mathcal{L}(\mathrm{T}\mathcal{M}, \mathrm{T}\mathcal{M}) = \{(X, Y, \mathcal{L}) : X, Y \in \mathcal{M} \text{ and } \mathcal{L} : \mathrm{T}_X \mathcal{M} \rightarrow \mathrm{T}_Y \mathcal{M} \text{ is linear}\},$$

is itself a manifold [Bou23a, Proposition 10.60]. Let $\mathcal{U} \subseteq \mathcal{M} \times \mathcal{M}$ be open and containing the diagonal. A transporter on $\mathcal{U}$ is a smooth map

$$\mathrm{T} : \mathcal{U} \rightarrow \mathcal{L}(\mathrm{T}\mathcal{M}, \mathrm{T}\mathcal{M}), \quad \mathrm{T}(X, Y) = \mathrm{T}_{Y \leftarrow X}$$

such that $\mathrm{T}_{Y \leftarrow X} : \mathrm{T}_X \mathcal{M} \rightarrow \mathrm{T}_Y \mathcal{M}$ is linear and $\mathrm{T}_{X \leftarrow X}$ is the identity.

Proposition 2.24. [Bou23a, Proposition 10.66] *Let $\mathcal{M}$ be an embedded submanifold of a Euclidean space $\mathcal{E}$. A transporter on $\mathcal{M} \times \mathcal{M}$ is given by*

$$\mathrm{T}_{Y \leftarrow X} = \mathrm{Proj}_Y|_{\mathrm{T}_X \mathcal{M}} : \mathrm{T}_X \mathcal{M} \rightarrow \mathrm{T}_Y \mathcal{M}, \quad \forall X, Y \in \mathcal{M}. \quad (2.13)$$

The notion of transporter can be linked to that of *vector transport*, which is more used to prove convergence results, such as in [Sat21; Sat22]. We refer the reader to [Bou23a, § 10.5].

2.2 Embedded geometry of fixed-rank matrices

We now want to make explicit the Riemannian geometry tools defined in Section 2.1 for the manifold of fixed-rank matrices. We can assume without loss of generality that $m \geq n$. Again, we stress that our target is the case $r \ll m, n$. This section consists of facts known in the literature, and is mainly based on [Bou23a, § 7.5] and [Van13]. We start by recalling that fixed-rank matrices are indeed an embedded submanifold.

Proposition 2.25. *Given $m \geq n$, the set $\mathcal{M}_r = \{X \in \mathbb{R}^{m \times n} : \mathrm{rank}(X) = r\}$ is an embedded submanifold of $\mathbb{R}^{m \times n}$ for every $0 \leq r \leq n$. Moreover, $\dim \mathcal{M}_r = r(m + n - r)$.*

Let us denote the Stiefel manifold [Bou23a, § 7.3] as

$$\mathrm{St}(p, r) = \{X \in \mathbb{R}^{p \times r} : X^\top X = I_r\}.$$

Using the (thin) SVD one can write

$$\mathcal{M}_r = \left\{ U \Sigma V^\top : U \in \mathrm{St}(m, r), V \in \mathrm{St}(n, r), \Sigma = \mathrm{diag}(\sigma_i), \sigma_1 \geq \dots \geq \sigma_r > 0 \right\}. \quad (2.14)$$

In the following, when we write $X = U \Sigma V^\top$ we always mean the SVD of the matrix X .

Proposition 2.26 (Tangent space). *The tangent space $\mathrm{T}_X \mathcal{M}_r$ at $X = U \Sigma V^\top \in \mathcal{M}_r$ is given by*

$$\mathrm{T}_X \mathcal{M}_r = \left\{ [U \ U_\perp] \begin{bmatrix} M & K \\ H & 0 \end{bmatrix} [V \ V_\perp]^\top : H \in \mathbb{R}^{r \times r}, K \in \mathbb{R}^{r \times (n-r)}, H \in \mathbb{R}^{(m-r) \times r} \right\} \quad (2.15)$$

2.2. Embedded geometry of fixed-rank matrices

where $U_\perp$ is such that $\begin{bmatrix} U & U_\perp \end{bmatrix} \in \mathbb{R}^{m \times m}$ is orthogonal, and $V_\perp$ is such that $\begin{bmatrix} V & V_\perp \end{bmatrix} \in \mathbb{R}^{n \times n}$ is orthogonal. Equivalently,

$$\begin{aligned} \mathcal{T}_X \mathcal{M}_r &= \left\{ U M V^\top + U_p V^\top + U V_p^\top : \right. \\ &\quad \left. M \in \mathbb{R}^{r \times r}, U_p \in \mathbb{R}^{m \times r}, V_p \in \mathbb{R}^{n \times r}, U^\top U_p = 0, V^\top V_p = 0 \right\}. \end{aligned} \quad (2.16)$$

Representing points and tangent vectors In order to be able to run optimization algorithms, in practice, we first need to find a way to uniquely identify and store in memory points of $\mathcal{M}_r$ and tangent vectors. Storing full matrices, which are of size $m \times n$, is not feasible if m, n are very large. However, we are helped by the SVD (2.14) and by (2.16). Instead of storing $X \in \mathcal{M}_r \subset \mathbb{R}^{m \times n}$, it is convenient to store its SVD factors (U, Σ, V) , which take only $\mathcal{O}(rm + rn + r^2) \ll \mathcal{O}(mn)$ memory. In the following, we write $X \doteq (U, \Sigma, V)$ or $X = U \Sigma V^\top$ to mean that X is in practice represented by the triplet (U, Σ, V) . We point out that this is just a *representation*, it is not unique and has no bearing on the geometry of $\mathcal{M}_r$.

With the same notation as in (2.16), once we have $X \in \mathcal{M}_r$ represented by its SVD, to represent a tangent vector $\xi \in \mathcal{T}_X \mathcal{M}_r$ we only need to store the triplet of matrices (M, U_p, V_p) , which are also small. When it is clear to which tangent space ξ belongs, we write $\xi \doteq (M, U_p, V_p)$.

The metric and computing inner products At least for the moment, we consider $\mathcal{M}_r$ as a Riemannian submanifold of $\mathbb{R}^{m \times n}$, with the usual Frobenius inner product $\langle A, B \rangle = \text{tr}(A^\top B)$. Observe that if $X \in \mathcal{M}_r$ is represented by the triplet (U, Σ, V) and $\xi, \xi' \in \mathcal{T}_X \mathcal{M}_r$ are such that $\xi \doteq (M, U_p, V_p), \xi' \doteq (M', U'_p, V'_p)$, then

$$\langle \xi, \xi' \rangle_X = \langle \xi, \xi' \rangle = \langle M, M' \rangle + \langle U_p, U'_p \rangle + \langle V_p, V'_p \rangle \quad (2.17)$$

can be computed as a sum of inner products of small matrices.

Orthogonal projectors We now give explicit formulas for calculating the orthogonal projection to the tangent, with respect to the Frobenius metric.

Proposition 2.27. *Consider $X = U \Sigma V^\top \in \mathcal{M}_r$ and let us define*

$$P_U = U U^\top, \quad P_V = V V^\top, \quad P_U^\perp = I_m - P_U, \quad P_V^\perp = I_n - P_V. \quad (2.18)$$

The orthogonal projection of $Z \in \mathbb{R}^{m \times n}$ to $\mathcal{T}_X \mathcal{M}_r$, with respect to the Frobenius metric, is given by

$$\text{Proj}_X(Z) = P_U Z P_V + P_U^\perp Z P_V + P_U Z P_V^\perp, \quad (2.19)$$

2.2. Embedded geometry of fixed-rank matrices

which is represented by

$$\begin{aligned} M &= U^\top ZV, \\ U_p &= ZV - UM, \\ V_p &= Z^\top U - VM^\top. \end{aligned} \tag{2.20}$$

Proof. From (2.15), the normal space at $X = U\Sigma V^\top$ is

$$N_X \mathcal{M}_r = \{U_\perp Y V_\perp : Y \in \mathbb{R}^{(m-r) \times (n-r)}\}.$$

Thus, the orthogonal projection of Z to the normal space is

$$\text{Proj}_X^\perp(Z) = (U_\perp U_\perp^\top)Z(V_\perp V_\perp^\top) = (I_m - UU^\top)Z(I_n - VV^\top) = P_U^\perp Z P_V^\perp,$$

and the orthogonal projection to its orthogonal complement $T_X \mathcal{M}_r$ is

$$\begin{aligned} \text{Proj}_X(X) &= Z - \text{Proj}_X^\perp(Z) = Z - P_U^\perp Z P_V^\perp = (P_U + P_U^\perp)Z(P_V + P_V^\perp) - P_U^\perp Z P_V^\perp \\ &= P_U Z P_V + P_U^\perp Z P_V + P_U Z P_V^\perp. \end{aligned}$$

The formulas in (2.20) are straightforward calculations. $\square$

Observe that, to efficiently compute the projection $\text{Proj}_X(Z)$, one simply needs to be able to efficiently apply Z and $Z^\top$. For instance, this is the case if Z admits a low-rank factorization.

Retraction As a retraction, we consider the metric projection retraction

$$R_X(\xi) = \arg \min_{Y \in \mathcal{M}_r} \|X + \xi - Y\|_F. \tag{2.21}$$

As discussed in Section 2.1.3, it is a (locally) well-defined retraction, and it is second-order if we consider $\mathcal{M}_r$ as a Riemannian submanifold of $\mathbb{R}^{m \times n}$ with the Frobenius metric.

By the Eckhart-Young-Mirsky theorem the metric projection to $\mathcal{M}_r$, when it exists, is given by the rank- r TSVD. However, it is not well-defined on the whole $\mathbb{R}^{m \times n}$, and in particular it is not well-defined at Y if its r -th and $(r+1)$ -th singular values coincide (if $\sigma_r = \sigma_{r+1} = 0$ the solution does not exist, if $\sigma_r = \sigma_{r+1} > 0$ it is not unique). As a consequence, the metric projection retraction is not well-defined on the whole tangent bundle. Still, where it is defined it can be computed efficiently by exploiting that $X + \xi$ has rank at most $2r$, if $(X, \xi) \in T\mathcal{M}$. Indeed, taking $X \doteq (U, \Sigma, V)$ and $\xi \doteq (M, U_p, V_p)$, it holds

$$X + \xi = U(\Sigma + M)V^\top + U_p V^\top + U V_p^\top = \begin{bmatrix} U & U_p \end{bmatrix} \begin{bmatrix} \Sigma + M & I_r \\ I_r & 0_r \end{bmatrix} \begin{bmatrix} V & V_p \end{bmatrix}. \tag{2.22}$$

2.2. Embedded geometry of fixed-rank matrices

One can first calculate thin QR decompositions

$$\begin{bmatrix} U & U_p \end{bmatrix} = Q_U R_U, \quad \begin{bmatrix} V & V_p \end{bmatrix} = Q_V R_V,$$

(assuming $2r \leq \min\{m, n\}$, otherwise the procedure is slightly changed) and then compute the rank- r TSVD of the inner $2r \times 2r$ matrix $R_U \begin{bmatrix} \Sigma^+ M & I_r \\ I_r & 0_r \end{bmatrix} R_V^\top \approx \bar{U} \bar{\Sigma} \bar{V}^\top$. The retraction is then

$$R_X(\xi) = (Q_U \bar{U}_r) \bar{\Sigma}_r (Q_V \bar{V}_r)^\top \doteq (Q_U \bar{U}_r, \bar{\Sigma}_r, Q_V \bar{V}_r),$$

where $\bar{U}_r$ and $\bar{V}_r$ are the matrices obtained from the first r columns of $\bar{U}$ and $\bar{V}$, respectively, and $\bar{\Sigma}_r$ is the leading $r \times r$ submatrix of $\bar{\Sigma}$. As a final remark, observe that if one needs to compute $R_X(t\xi)$ for multiple values of t and the same $(X, \xi) \in \text{T}\mathcal{M}$, the QR decompositions can be saved after the first retraction: it suffices to replace $\begin{bmatrix} \Sigma^+ M & I_r \\ I_r & 0_r \end{bmatrix}$ with $\begin{bmatrix} \Sigma^+ + tM & tI_r \\ tI_r & 0_r \end{bmatrix}$. A more careful analysis of the computational cost shows that subsequent iterations can be up to 3 times faster [Bou23a, p. 164]. Still, the asymptotic complexity stays the same, i.e., $\mathcal{O}((m+n)r^2)$.

Transporter As a transporter we use the orthogonal projection restricted to the tangent (2.13). An efficient implementation, using $\mathcal{O}((m+n)r^2)$ flops, is described in [Van13, Algorithm 6].

Gradient and Hessian: from Euclidean to Riemannian Since we consider the manifold of fixed-rank matrices as a Riemannian submanifold of the ambient space $\mathbb{R}^{m \times n}$, we know from Propositions 2.4 and 2.6 that the Riemannian gradient and Hessian of a smooth map $f : \mathcal{M}_r \rightarrow \mathbb{R}$ are closely related to the Euclidean gradient and Hessian of any smooth extension $\bar{f}$. In particular, $\text{grad}f(X) = \text{Proj}_X(\text{grad}\bar{f}(X))$, for which formulas can be deduced from Proposition 2.27, and the following holds for the Riemannian Hessian.

Proposition 2.28. *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth map, with smooth extension $\bar{f}$ to a neighborhood of $\mathcal{M}_r$ in $\mathbb{R}^{m \times n}$. Consider $X = U\Sigma V^\top \in \mathcal{M}_r$, $\xi \in \text{T}_X \mathcal{M}_r$ represented by the triplet (M_ξ, U_ξ, V_ξ) , and let us denote $Z = \text{grad}\bar{f}(X)$, $\dot{Z} = \text{Hess}\bar{f}(X)[\xi]$. Then*

$$\begin{aligned} \text{Hess}f(X)[\xi] &= \text{Proj}_X \left(\text{Hess}\bar{f}(X)[\xi] \right) + \\ &+ \left(P_U^\perp \text{grad}\bar{f}(X) V_\xi \Sigma^{-1} \right) V^\top + U \left(P_V^\perp \text{grad}\bar{f}(X)^\top U_\xi \Sigma^{-1} \right)^\top, \end{aligned} \quad (2.23)$$

which is represented by the triplet

$$\begin{aligned} M &= U^\top \dot{Z} V, \\ U_p &= P_U^\perp \left(\dot{Z} V + Z V_\xi \Sigma^{-1} \right), \\ V_p &= P_V^\perp \left(\dot{Z}^\top U + Z^\top U_\xi \Sigma^{-1} \right). \end{aligned} \quad (2.24)$$

2.2. Embedded geometry of fixed-rank matrices

Proof. To be able to use (2.3), let us first work out an explicit expression for $\mathcal{D}_\xi$, the differential of $X \mapsto \text{Proj}_X$ at X along ξ . Our goal is to construct a smooth curve on $\mathcal{M}_r$ such that $c(0) = X$, $c'(0) = \xi$, and use that $\mathcal{D}_\xi = \frac{d}{dt}\text{Proj}_{c(t)}|_{t=0}$. Given that the tangent to the Stiefel manifold at U contains the Span of the columns of $U_\perp$ [Bou23a, eq. 7.20], consider a smooth curve $U(t)$ on $\text{St}(m, r)$ such that $U(0) = U$, $U'(0) = U_\xi \Sigma^{-1}$. Similarly, consider a smooth curve $V(t)$ on $\text{St}(n, r)$ such that $V(0) = V$, $V'(0) = V_\xi \Sigma^{-1}$, and finally take $\Sigma(t) = \Sigma + tM$. If we define $c(t) = U(t)\Sigma(t)V(t)^\top$, then $c(t) \in \mathcal{M}_r$ (for t small enough), $c(0) = X$ and $c'(0) = \xi$. For any $Y \in \mathbb{R}^{m \times n}$, we can rewrite (2.19) as

$$\begin{aligned} \text{Proj}_{c(t)}(Y) &= P_{U(t)} Y P_{V(t)} + P_{U(t)}^\perp Y P_{V(t)} + P_{U(t)} Y P_{V(t)}^\perp \\ &= Y P_{V(t)} + P_{U(t)} Y - P_{U(t)} Y P_{V(t)}. \end{aligned} \quad (2.25)$$

The derivative of $P_{U(t)}$ at time zero is

$$\begin{aligned} \dot{P}_U &:= \frac{d}{dt} P_{U(t)} \Big|_{t=0} = \frac{d}{dt} U(t) U(t)^\top \Big|_{t=0} = U'(0) U(0)^\top + U(0) U'(0)^\top \\ &= U_\xi \Sigma^{-1} U^\top + U \Sigma^{-1} U_\xi^\top. \end{aligned}$$

Similarly,

$$\dot{P}_V := \frac{d}{dt} P_{V(t)} \Big|_{t=0} = V_\xi \Sigma^{-1} V^\top + V \Sigma^{-1} V_\xi^\top.$$

Taking the derivative of (2.25) at time 0:

$$\begin{aligned} \mathcal{D}_\xi(Y) &= \frac{d}{dt} \text{Proj}_{c(t)}(Y) \Big|_{t=0} = Y \dot{P}_V + \dot{P}_U Y - P_U Y \dot{P}_V - \dot{P}_U Y P_V \\ &= P_U^\perp Y \dot{P}_V + \dot{P}_U Y P_V^\perp. \end{aligned}$$

We can now use that $Y = \text{Proj}_X^\perp(Z) = P_U^\perp Z P_V^\perp$, together with

$$\begin{aligned} P_U^\perp (U_\xi \Sigma^{-1} U^\top + U \Sigma^{-1} U_\xi^\top) &= U_\xi \Sigma^{-1} U^\top, \\ P_V^\perp (V_\xi \Sigma^{-1} V^\top + V \Sigma^{-1} V_\xi^\top) &= V_\xi \Sigma^{-1} V^\top, \end{aligned}$$

to obtain

$$\mathcal{D}_\xi \left(\text{Proj}_X^\perp(Z) \right) = (P_U^\perp Z V_\xi \Sigma^{-1}) V^\top + U (P_V^\perp Z^\top U_\xi \Sigma^{-1}).$$

Along with (2.3), this allows to prove (2.23). The formulas in (2.24) follow from (2.19). $\square$

Note that to efficiently compute the Riemannian gradient and Hessian, one simply needs to be able to efficiently apply $\text{grad} \bar{f}(X)$, $(\text{grad} \bar{f}(X))^\top$, $\text{Hess} \bar{f}(X)$ and $(\text{Hess} \bar{f}(X))^\top$.

Remark 2.29. Notice the Σ^{-1} factors in (2.24): if some singular values are close to 0, computing the Riemannian Hessian requires solving an ill-conditioned (diagonal) system. The factors Σ^{-1} are due to the curvature term of the Riemannian Hessian, and are not

2.2. Embedded geometry of fixed-rank matrices

present in the projection of the Euclidean Hessian.

Even when the Euclidean Hessian is positive definite, and therefore also its composition with the projection to the tangent is positive definite, it is not necessarily the case that the Riemannian Hessian is positive definite. The reason is in the additional curvature term, which can render the Riemannian Hessian indefinite. The following proposition characterizes the eigenvalues of the curvature term. See also [FL18, Theorem 24].

Proposition 2.30. *Let $f : \mathcal{M}_r \rightarrow \mathbb{R}$ be a smooth function with smooth extension $\bar{f}$. The nonzero eigenvalues of the curvature term of the Riemannian Hessian at $X \in \mathcal{M}_r$, i.e., $T_X \mathcal{M}_r \ni \xi \mapsto \mathcal{D}_\xi \left(\text{Proj}_X^\perp(\text{grad} \bar{f}(X)) \right)$, are all possible products of the nonzero singular values of $\text{Proj}_X^\perp(\text{grad} \bar{f}(X))$ and the diagonal entries of Σ^{-1} , and their negatives.*

Proof. With the same notation as in Proposition 2.28, let $U_\perp$ be such that $\begin{bmatrix} U & U_\perp \end{bmatrix} \in \mathbb{R}^{m \times m}$ is orthogonal, and $V_\perp$ be such that $\begin{bmatrix} V & V_\perp \end{bmatrix} \in \mathbb{R}^{n \times n}$ is orthogonal. Then we can write $U_\xi = U_\perp B_U$ and $V_\xi = V_\perp B_V^\top$ for $B_U \in \mathbb{R}^{(m-r) \times r}$ and $B_V \in \mathbb{R}^{r \times (n-r)}$. Vectorizing

$$\text{vec}(\xi) = \text{vec}(U M_\xi V^\top + U_\xi V^\top + U V_\xi^\top) = \underbrace{\begin{bmatrix} V \otimes U & V \otimes U_\perp & V_\perp \otimes U \end{bmatrix}}_W \begin{bmatrix} \text{vec}(M_\xi) \\ \text{vec}(B_U) \\ \text{vec}(B_V) \end{bmatrix}.$$

Denote the normal component of the gradient as $\text{Proj}_X^\perp(Z) = U_\perp(U_\perp^\top Z V_\perp) V_\perp^\top = U_\perp N V_\perp^\top$, and observe that the nonzero singular values of $\text{Proj}_X^\perp(Z)$ are the nonzero singular values of N . Moreover

$$\begin{aligned} \text{vec} \left(\mathcal{D}_\xi \left(\text{Proj}_X^\perp(Z) \right) \right) &= \text{vec}(P_U^\perp Z V_\xi \Sigma^{-1} V^\top + U \Sigma^{-1} U_\xi^\top Z P_V^\perp) \\ &= \text{vec} \left(U_\perp U_\perp^\top Z V_\perp B_V^\top \Sigma^{-1} V^\top + U \Sigma^{-1} (U_\perp B_U)^\top Z V_\perp V_\perp^\top \right) \\ &= \text{vec} \left(U_\perp N B_V^\top \Sigma^{-1} V^\top + U \Sigma^{-1} B_U^\top N V_\perp^\top \right) \\ &= W \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \Pi_{m-r,r} (N \otimes \Sigma^{-1}) \\ 0 & (N^\top \otimes \Sigma^{-1}) \Pi_{m-r,r} & 0 \end{bmatrix} \begin{bmatrix} \text{vec}(M_\xi) \\ \text{vec}(B_U) \\ \text{vec}(B_V) \end{bmatrix}, \end{aligned}$$

where $\Pi_{m-r,r} \in \mathbb{R}^{r(m-r) \times r(m-r)}$ is the *perfect shuffle matrix* [Ros80], i.e., the matrix such that $\text{vec}(K^\top) = \Pi_{m-r,r} \text{vec}(K)$ for all $K \in \mathbb{R}^{(m-r) \times r}$. Therefore, after vectorization, in the basis W the curvature term is represented by the matrix

$$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & K^\top \\ 0 & K & 0 \end{bmatrix}, \quad K = (N^\top \otimes \Sigma^{-1}) \Pi_{m-r,r}.$$

The eigenvalues of C are 0 with multiplicity r^2 , and the singular values of K and their negatives. Since $\Pi_{m-r,r}$ is a permutation matrix, the singular values of K are the singular

2.2. Embedded geometry of fixed-rank matrices

values of $N^\top \otimes \Sigma^{-1}$. To conclude, it suffices to observe that the nonzero singular values of $N^\top \otimes \Sigma^{-1}$ are all possible products of the nonzero singular values of N and the diagonal entries of Σ^{-1} . $\square$

Chapter 3

Riemannian optimization algorithms

In this chapter, we examine optimization algorithms for problems of the form $\min_{X \in \mathcal{M}} f(X)$, where $\mathcal{M}$ is a Riemannian manifold and $f : \mathcal{M} \rightarrow \mathbb{R}$ is a smooth cost function. All the methods we consider fit into the retraction-based framework of Riemannian optimization [AMS08; VV10; Bou23a], that is, are based on updates of the form

$$X_{k+1} = R_{X_k}(\xi_k)$$

where R is a retraction on $\mathcal{M}$ and $\xi_k \in T_{X_k} \mathcal{M}$ is a tangent vector at X_k . We specifically consider the extension of line-search and trust-region methods to Riemannian manifolds. Finally, we introduce the notion of Riemannian preconditioner as a change of the Riemannian metric, allowing to easily define preconditioned Riemannian algorithms and to extend convergence guarantees to the preconditioned case. Indeed, as it will be clear from (4.8), an ill-conditioned operator $\mathcal{A}$ is likely to result in a Riemannian Hessian with similar ill-conditioning. Analogous to the Euclidean case, an ill-conditioned Riemannian Hessian worsens the convergence factor of first-order line-search methods and increases the computational cost of computing (approximate) Newton-like steps via matrix-free methods. Therefore, preconditioning becomes essential to accelerate the convergence of ill-conditioned problems, as the ones of interest.

We particularly focus on Riemannian submanifolds, as they are most pertinent to the current problem. However, much of the discussion remains applicable to generic Riemannian manifolds, assuming sufficient regularity.

3.1 The notion of convergence

In utilizing iterative algorithms on $\mathcal{M}$, it becomes crucial to precisely define convergence. When referring to *convergence*, *limit*, and *accumulation points* of a sequence $\{X_k\}_{k=0}^{+\infty} \subset \mathcal{M}$, we specifically mean in the topology of the manifold $\mathcal{M}$. Embedded submanifolds of Euclidean spaces are always Hausdorff, hence the limit is unique.

3.1. The notion of convergence

Defining the convergence of an algorithm is a more delicate matter. Seeking a solution to $\min_{X \in \mathcal{M}} f(X)$, what we aim for is a global minimizer of f on $\mathcal{M}$ (assuming it exists). However, this goal is in most cases unattainable, even in Euclidean optimization. Finding local minimizers is also challenging, primarily because first- and second-order optimality conditions (Proposition 2.11) are necessary but not sufficient conditions. Typically, both in Euclidean and Riemannian optimization, the approach is to *aim for minima, but guarantee small gradients*. Ideally, the iterates should also converge to a point. This rationale underpins the following definitions of global and local convergence.

Definition 3.1 (Global convergence). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth function on a Riemannian manifold $\mathcal{M}$. An iterative method is globally convergent if, for any initial point X_0 , the sequence $\{X_k\}_{k=0}^{+\infty}$ generated by the method is such that*

$$\lim_{k \rightarrow +\infty} \|\text{grad} f(X_k)\|_{X_k} = 0.$$

Remark 3.2. *Observe that the global convergence property does not guarantee convergence of the iterates X_k to a critical point. What it does guarantee is that all accumulation points are critical points.*

Definition 3.3 (Local convergence). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth function on a Riemannian manifold $\mathcal{M}$. We say that an iterative method is locally convergent to $X_\star \in \mathcal{M}$ if there exist a neighborhood $\mathcal{U}$ of $X_\star$ such that, if $X_K \in \mathcal{U}$ for some K , then $X_k \in \mathcal{U}$ for all $k \geq K$ and $X_k \rightarrow X_\star$.*

In addition to the convergence of a sequence, understanding *how quickly* the sequence converges is crucial. To address this, we recall the definition of R-convergence order on metric spaces.

Definition 3.4 (Order of convergence, [Bou23a, Definitions 4.14, 4.15]). *In a metric space with a distance dist , a sequence $\{X_k\}_{k=0}^{+\infty}$ converges at least with order $p \geq 1$ to $X_\star$ if there exist positive reals $\{\varepsilon_k\}_{k=1}^{+\infty}$ converging to zero such that $\text{dist}(X_k, X_\star) \leq \varepsilon_k$ and $\lim_{k \rightarrow \infty} \frac{\varepsilon_{k+1}}{\varepsilon_k^p} = \mu$ for some finite $\mu \geq 0$.*

If $p = 1$, we say that the sequence converges at least linearly. Moreover, the infimum over such μ 's is termed the linear convergence factor. If this factor equals zero, the convergence is labeled as superlinear.

If $p = 2$, we say that the sequence converges at least quadratically.

To make the notion of convergence order meaningful for optimization on Riemannian manifolds, it is essential to clarify how to define a distance on a Riemannian manifold. Here, we introduce a classical result concerning the *Riemannian distance*.

Proposition 3.5 ([Lee18, Theorem 2.55]). *Let $\mathcal{M}$ be a Riemannian submanifold. A smooth curve segment $c : [a, b] \subset \mathbb{R} \rightarrow \mathcal{M}$ is called regular if $c'(t) \neq 0$, and piecewise regular if there exists a finite set of times $a = t_0 < t_1 < \dots < t_k = b$ such that the*

3.2. Line-search methods on manifolds

restrictions $c|_{[t_{i-1}, t_i]}$ are regular. For any piecewise regular curve segment define the length of c , denoted as $L(c)$, as the integral of its speed $\|c'(t)\|_{c(t)}$. For $X, Y \in \mathcal{M}$ define

$$\text{dist}(X, Y) = \inf_c L(c)$$

where the infimum is taken over all piecewise regular curve segments on $\mathcal{M}$ which connect X and Y . If $\mathcal{M}$ is connected, then dist is a distance on $\mathcal{M}$, and it is called the Riemannian distance. Moreover, with the distance function dist , $\mathcal{M}$ is a metric space whose metric topology is the same as the manifold topology

If $\mathcal{M}$ is not connected, we can apply this result to each connected component of $\mathcal{M}$ and assign $\text{dist}(X, Y) = +\infty$ if X and Y belong to different connected components. However, in retraction-based optimization iterates are connected by retraction curves, hence it is reasonable to restrict to connected manifolds only.

3.2 Line-search methods on manifolds

3.2.1 General framework

Line-search methods on Euclidean spaces perform updates

$$X_{k+1} = X_k + \alpha_k d_k, \quad (3.1)$$

where d_k is often referred to as the *search direction*, and $\alpha_k > 0$ is the *step-size*. As the nomenclature suggests, this type of update can be interpreted as moving in the direction of d_k , i.e., along the line $t \mapsto X_k + td_k$, by a step of size α_k . Given a manifold $\mathcal{M}$ equipped with a retraction R , the update (3.1) gets generalized by selecting a search direction $\xi_k \in T_{X_k} \mathcal{M}$ and computing the next iterate as

$$X_{k+1} = R_{X_k}(\alpha_k \xi_k), \quad (3.2)$$

i.e., moving by a step of size α_k along the curve $t \mapsto c(t) = R_{X_k}(t\xi_k)$, which by definition of retraction has the property $c'(0) := \xi_k$. The general structure of a line-search method on a manifold is summarized in Algorithm 2. Fully specifying such a method requires choosing: a retraction R on $\mathcal{M}$, a strategy for selecting the search direction $\xi_k \in T_{X_k} \mathcal{M}$, and a strategy for computing the step-size α_k . Setting aside, for a moment, the choice of retraction and search direction, we now introduce a few possible strategies for step-size selection, with a particular emphasis on Armijo backtracking.

3.2.1.1 Step-size selection and Armijo backtracking

Consider $X \in \mathcal{M}$ and a search direction $\xi \in T_X \mathcal{M}$. For the update $X_+ = R_X(\alpha\xi)$ to be meaningful, we need the objective function f to decrease along the curve $t \mapsto R_X(t\xi)$, at least for sufficiently small steps. From the first-order Taylor expansion (2.8), this

3.2. Line-search methods on manifolds

Algorithm 2 Line-search method on Riemannian manifolds

Require: Riemannian manifold $\mathcal{M}$, smooth objective function $f : \mathcal{M} \rightarrow \mathbb{R}$

Input: Retraction R , initial point $X_0 \in \mathcal{M}$

Output: Sequence $\{X_k\} \subset \mathcal{M}$

```

1: for  $k = 0, 1, 2, \dots$  do
2:   Compute a search direction  $\xi_k \in T_{X_k}\mathcal{M}$ 
3:   Compute a step-size  $\alpha_k > 0$ 
4:    $X_{k+1} = R_{X_k}(\alpha_k \xi_k)$ 
5: end for
```

happens if and only if ξ is a *descent direction* for f at X , meaning

$$\langle \text{grad} f(X), \xi \rangle_X < 0. \quad (3.3)$$

Assuming ξ is a descent direction, the optimal choice for α would be $\alpha \in \arg \min_{t>0} R_X(t\xi)$, known as an *exact line-search*. However, an exact line-search is usually not feasible in practice, and we settle for α to satisfy less stringent requirements that still ensure it is acceptable. We employ the *Armijo-Goldstein* condition, which demands that the actual decrease in the objective function is at least a fraction r of the decrease of the first-order approximation.

Definition 3.6 (Armijo-Goldstein condition). *Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth function on a Riemannian manifold $\mathcal{M}$ equipped with a retraction R . Consider $X \in \mathcal{M}$ and assume that $\xi \in T_X\mathcal{M}$ is a descent direction. Given a constant $r \in]0, 1[$, a step-size $\alpha > 0$ satisfies the Armijo-Goldstein condition with constant r if*

$$f(R_X(\alpha\xi)) \leq f(X) + r \cdot \alpha \langle \text{grad} f(X), \xi \rangle_X. \quad (3.4)$$

Note that the Armijo-Goldstein condition is always met for sufficiently small step-sizes, if ξ is a descent direction. To compute a step-size α satisfying it, the *Armijo backtracking line-search algorithm* (Algorithm 3) is employed. Starting with an initial step-size $\bar{\alpha} > 0$, the algorithm reduces the tentative step-size by a factor $\tau \in]0, 1[$ at each step until condition (3.4) is satisfied.

As a final remark, we point out that the Armijo-Goldstein condition is a standard choice in Riemannian optimization, used as the default also in [AMS08; Bou23a]. For completeness, it is worth mentioning that the Wolfe and strong-Wolfe conditions can also be generalized to optimization problem on Riemannian manifolds [Sat21, § 3.5], but this involves computing the differential of R_X at points other than the origin. Although we know that, by definition of a retraction, the differential at the origin is the identity map, it can be challenging and computationally expensive to compute it at points other than the origin. For the manifold of fixed-rank matrices, formulas can be found in [DE99; FL18], and the computational cost of a Wolfe line-search is not excessive, albeit greater than that of Armijo backtracking. Nevertheless, for the sake of a simpler implementation,

3.2. Line-search methods on manifolds

Algorithm 3 Armijo backtracking line-search on Riemannian manifolds

Require: Riemannian manifold $\mathcal{M}$, smooth objective function $f : \mathcal{M} \rightarrow \mathbb{R}$, retraction R , point $X \in \mathcal{M}$, descent direction $\xi \in T_X \mathcal{M}$

Parameters: $r, \tau \in]0, 1[$; we employ $r = 10^{-4}$ and $\tau = 0.5$

Input: Initial step-size $\bar{\alpha}$

Output: Step-size $\alpha > 0$ satisfying the Armijo-Goldstein condition (3.4)

```

1: Set  $\alpha = \bar{\alpha}$ 
2: while  $f(R_X(\alpha\xi)) > f(X) + r \cdot \alpha \langle \text{grad} f(X), \xi \rangle_X$  do
3:    $\alpha \leftarrow \tau \cdot \alpha$ 
4: end while
5: return  $\alpha$ 

```

we opted to use the standard Armijo-Goldstein condition.

3.2.2 Riemannian gradient descent (R-GD)

Similarly to gradient descent in Euclidean optimization, *Riemannian gradient descent* (R-GD) selects the negative gradient as the descent direction:

$$\xi_k = -\text{grad} f(X_k), \quad (3.5)$$

which represents the optimal search direction according to the first-order Taylor expansion (2.8) of the pullback. This choice leads to the following global convergence guarantees.

Proposition 3.7 (Global convergence of R-GD). *Let $\{X_k\}_{k=0}^{+\infty} \subset \mathcal{M}$ be a sequence of points generated by Riemannian gradient descent with Armijo backtracking. Assume that the iterates are contained in a compact subset of $\mathcal{M}$. Then $\lim_{k \rightarrow +\infty} \|\text{grad} f(X_k)\|_{X_k} = 0$. In particular, if the sublevel sets $\{X \in \mathcal{M} : f(X) \leq c\}$ are compact for all $c \in \mathbb{R}$ Riemannian gradient descent with Armijo backtracking is globally convergent.*

Proof. Analogous to the proof of [AMS08, Corollary 4.3.2]. $\square$

The previous proposition provides guarantees about global convergence in the sense of Definition 3.1. It is important to recall that this does not imply that the sequence generated by R-GD converges to a point; rather, it implies that the accumulation points of the sequence, if any, are critical points. Additionally, results in the literature typically ensure, under suitable additional assumptions, convergence of the gradient norm to zero as fast as $1/\sqrt{k}$ [Bou23b, § 4.3-4.5], which is often pessimistic compared to what observed in practice. To gain a better understanding of practical behavior, it is useful to examine local convergence of R-GD.

Proposition 3.8 (Local convergence of R-GD, [AMS08, Theorem 4.5.6] and [Bou23a, pp. 65–67]). *Let $X_\star \in \mathcal{M}$ be a non-degenerate local minimizer of f . Then Riemannian gradient descent with Armijo backtracking is locally convergent to $X_\star$, at least linearly.*

Moreover, $\|\text{grad}f(X_k)\|_{X_k}$ also converges to 0, and $f(X_k)$ converges to $f(X_*)$, both at least linearly.

It is also possible to provide an upper bound on the linear convergence factor of R-GD. For precise statements, refer to [AMS08, Theorem 4.5.6] and [Bou23a, Theorem 4.20]. In essence, the convergence factor is of the form $\mu = 1 - c/\kappa$, where c is a constant depending on the step-size selection procedure, and $\kappa = \kappa(\text{Hess}f(X_*))$ denotes the condition number of the Riemannian Hessian at X_* . This justifies our earlier statement that an ill-conditioned Riemannian Hessian is expected to result in slow convergence of R-GD.

3.2.3 Riemannian nonlinear conjugate gradient (R-NLCG)

In nonlinear conjugate gradient (NLCG) on Euclidean spaces, the search direction is obtained through the formula

$$d_k = -\text{grad}f(X_k) + \beta_k d_{k-1} \quad (3.6)$$

where $\beta_k \in \mathbb{R}$, and $d_0 = -\text{grad}f(X_0)$. A potential rationale for this update stems from considering the resolution of the system $Ax = b$ with $A \in \mathbb{R}^{n \times n}$ symmetric positive definite, which is known to be equivalent to finding the minimum of the quadratic $q: \mathbb{R}^n \rightarrow \mathbb{R}$ defined as $q(x) := \frac{1}{2}x^\top Ax - b^\top x$. Applying gradient descent with cost function q corresponds to Richardson's iteration [Ric11]. However, a better iterative method for solving symmetric positive definite linear systems is the conjugate gradient method [HS52], which performs updates that are equivalent to (3.6) with cost function q and $\beta_k = \|\text{grad}q(X_k)\|^2 / \|\text{grad}q(X_{k-1})\|^2$. Extending this logic to generic objective functions f , akin to the superiority of conjugate gradient over Richardson's iteration, one can expect nonlinear conjugate gradient to outperform gradient descent, especially near a non-degenerate minimizer, where f is approximately a strongly convex quadratic. For calculating β_k in (3.6), various heuristic formulas exist, all of which are equivalent to $\beta_k = \|\text{grad}f(X_k)\|^2 / \|\text{grad}f(X_{k-1})\|^2$ for $f(x) := \frac{1}{2}x^\top Ax - b^\top x$, but differ when dealing with general objective functions f . We refer to [HZ06] for an extended survey.

Riemannian Nonlinear Conjugate Gradient (R-NLCG) extends the nonlinear conjugate gradient method on Euclidean spaces to Riemannian optimization. The challenge in this generalization lies in the fact that $\xi_{k-1} \in T_{X_{k-1}}\mathcal{M}$ and $\text{grad}f(X_k) \in T_{X_k}\mathcal{M}$ lie in different tangent spaces, and cannot be summed. Transporters, introduced in Section 2.1.4 for this very purpose, allow transporting ξ_{k-1} to the tangent at the current iterate. The resulting search direction is given by:

$$\xi_k = -\text{grad}f(X_k) + \beta_k T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \quad (3.7)$$

for some $\beta_k \in \mathbb{R}$. Notably, when $\beta_k = 0$, R-NLCG is equivalent to Riemannian GD. To ensure that ξ_k is a descent direction, if $\langle \text{grad}f(X_k), \xi_k \rangle_{X_k} \geq 0$, the search direction is reset to the negative gradient $\xi_k = -\text{grad}f(X_k)$. A pseudocode of R-NLCG is provided

3.2. Line-search methods on manifolds

in Algorithm 4.

Algorithm 4 Riemannian nonlinear conjugate gradient

Require: Riemannian manifold $\mathcal{M}$, smooth objective function $f : \mathcal{M} \rightarrow \mathbb{R}$

Input: Retraction R , transporter T , initial point $X_0 \in \mathcal{M}$

Output: Sequence $\{X_k\} \subset \mathcal{M}$

```

1: Set  $\xi_0 = -\text{grad}f(X_0)$ 
2: Compute a step-size  $\alpha_0 > 0$ 
3:  $X_1 = R_{X_0}(\alpha_0 \xi_0)$ 
4: for  $k = 0, 1, 2, \dots$  do
5:   Compute  $\beta_k \in \mathbb{R}$ 
6:    $\xi_k = -\text{grad}f(X_k) + \beta_k T_{X_k \leftarrow X_{k-1}}(\xi_{k-1})$ 
7:   if  $\langle \text{grad}f(X_k), \xi_k \rangle_{X_k} \geq 0$  then                                 $\triangleright$  If  $\xi_k$  is not a descent direction
8:      $\xi_k = -\text{grad}f(X_k)$                                                  $\triangleright$  Reset to the negative gradient
9:   end if
10:  Compute a step-size  $\alpha_k > 0$ 
11:   $X_{k+1} = R_{X_k}(\alpha_k \xi_k)$ 
12: end for

```

Compared to R-GD, the Riemannian nonlinear conjugate gradient method necessitates a few additional tools - specifically, a transporter and a strategy for computing β_k . A comprehensive survey of methods for calculating β_k is provided in [Sat22]. Among various options, we selected the Hestenes-Stiefel modified rule [Sat22, § 6.2.2] due to its superior experimental performance on the problem at hand. Denoting $g_k := \text{grad}f(X_k)$

$$\beta_k = \max \left(0, \min(\beta_k^{HS}, \beta_k^{DY}) \right), \quad (3.8)$$

where

$$\begin{aligned} \beta_k^{HS} &= \frac{\|g_k\|_{X_k}^2 - \langle g_k, T_{X_k \leftarrow X_{k-1}}(g_{k-1}) \rangle_{X_k}}{\langle g_k, T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \rangle_{X_k} - \langle g_{k-1}, \xi_{k-1} \rangle_{X_{k-1}}}, \\ \beta_k^{DY} &= \frac{\|g_k\|_{X_k}^2}{\langle g_k, T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \rangle_{X_k} - \langle g_{k-1}, \xi_{k-1} \rangle_{X_{k-1}}}. \end{aligned} \quad (3.9)$$

We point out that using a transporter for transporting the previous descent direction to the current tangent space is nonstandard. Typically, vector transports, possibly scaled, are employed for this purpose [Sat21; Sat22; AMS08]. Nevertheless, a transporter defines a vector transport (see [Bou23b, § 10.5]), and for the manifold of fixed-rank matrices it is suggested in [Van13; Bou+13] to use a transporter. Therefore, we opted for a simpler presentation of R-NLCG using transporters.

Conducting a convergence analysis of nonlinear conjugate gradient is intricate, already on Euclidean spaces [GN92], and more so on Riemannian manifolds. Global convergence guarantees for R-NLCG have been established in [SI15; Sat21; Sat22] but

rely on a more complex version of R-NLCG than the one outlined in Algorithm 4. This advanced version imposes stricter requirements on vector transport and a line-search procedure enforcing the strong-Wolfe conditions, as described in [Sat21; Sat16]. Given the satisfactory performance of Algorithm 4 in our numerical experiments, we carry on with this simpler version of R-NLCG as an acceleration technique for R-GD, acknowledging the absence of formal convergence guarantees.

3.3 Riemannian Trust-Region (R-TR)

The Riemannian Trust-Region (R-TR) method, originally introduced in [ABG07], extends the classical Trust-Region method [CGT00] to Riemannian manifolds. Unlike line-search methods, R-TR does not involve first selecting a search direction and then computing a suitable step-size. Instead, it computes the step directly from a quadratic model for the pullback $\hat{f}_{X_k} = f \circ R_{X_k}$.

The core concept of R-TR involves constructing a quadratic model $m_{X_k} : T_{X_k}\mathcal{M} \rightarrow \mathbb{R}$ such that $m_{X_k} \approx \hat{f}_{X_k}$ in the vicinity of the origin. Recognizing that the model is only a local approximation of $\hat{f}_{X_k}$, it is *trusted* only in a ball of radius Δ_k around the origin: the *trust-region*. A candidate step is proposed by approximately minimizing m_{X_k} within the trust-region and is then accepted or rejected based on the validity of the model, that is, assessing if the actual decrease of the objective function aligns approximately with that predicted by the model. The radius Δ_k of the trust-region may also be adjusted accordingly.

The model Let $X \in \mathcal{M}$ be the current iterate. The best local quadratic model of the pullback around the origin of $T_X\mathcal{M}$ would be the second-order Taylor expansion

$$\begin{aligned} m_X(\xi) &:= \hat{f}_X(0) + \langle \text{grad} \hat{f}_X(0), \xi \rangle_X + \frac{1}{2} \langle \text{Hess} \hat{f}_X(0)[\xi], \xi \rangle_X \\ &= f(X) + \langle \text{grad} f(X), \xi \rangle_X + \frac{1}{2} \langle \text{Hess} \hat{f}_X(0)[\xi], \xi \rangle_X. \end{aligned}$$

Note that, according to Proposition 2.20, if the retraction is second order $\text{Hess} \hat{f}_X(0) = \text{Hess} f(X)$. However, the computation of $\text{Hess} \hat{f}_X(0)$ may be challenging for a general retraction R . For this reason, we consider a broader class of models $m_X : T_X\mathcal{M} \rightarrow \mathbb{R}$ defined as

$$m_X(\xi) = f(X) + \langle \text{grad} f(X), \xi \rangle_X + \frac{1}{2} \langle H_X[\xi], \xi \rangle_X, \quad (3.10)$$

where there is no assumption on $H_X : T_X\mathcal{M} \rightarrow T_X\mathcal{M}$ other than being a symmetric linear operator. We will often refer to H_X as *the model Hessian*. We anticipate that the global convergence of R-TR is essentially independent of the choice of H_X (as long as its operator norm does not explode), while to achieve local superlinear convergence H_X is required to be an "approximate Hessian", in a sense specified later.

The Trust-Region subproblem (TRs) After defining the model through the selection of H_X , the computation of the candidate next iterate involves approximately solving the *Trust-Region subproblem* (TRs)

$$\min_{\xi \in T_X \mathcal{M}} m_X(\xi) \quad \text{subject to} \quad \|\xi\|_X \leq \Delta. \quad (3.11)$$

This inner problem is arguably the most computationally demanding part of the R-TR algorithm. To solve it approximately, we use the *Steihaug-Toint truncated conjugate gradient* (tCG) method [CGT00, § 7.5.1]. The fundamental idea of tCG is to run CG on the quadratic (3.10), defined on the Euclidean space $T_X \mathcal{M}$, and simultaneously:

1. stop if we would leave the trust-region $\{\xi \in T_X \mathcal{M} : \|\xi\|_X \leq \Delta\}$, either because we detect that H_X has a negative eigenvalue (indicating that m_X is unbounded below on $T_X \mathcal{M}$) or because the next iterate of CG would be outside the trust-region;
2. avoid spending too much time unnecessarily.

The tCG method is outlined in Algorithm 5 for generic quadratics on finite-dimensional Euclidean vector spaces. When reading it, consider $(T, \langle \cdot, \cdot \rangle)$ as $(T_X \mathcal{M}, \langle \cdot, \cdot \rangle_X)$, $H = H_X$ and $g = -\text{grad}f(X)$. Note that tCG is identical to CG, except for lines 7 to 10 and lines 13 to 15, which both early stop the iterations. Until entering the block (7-9), the directions p_i generated by tCG are the same as CG. In particular, for all $i = 1, \dots, k-1$ the directions p_i are H -conjugate and $\langle p_i, H p_i \rangle > 0$ holds. Hence, H is positive definite on $\text{Span}(p_1, \dots, p_{k-1})$, and up to the $(k-1)$ -th iterate tCG retains the same properties as CG. The if condition in line 7 is met in one of the following two cases:

1. H is not positive definite, and we found a direction p_k along which the quadratic is unbounded below on T .
2. The next iterate ξ_k^+ of CG would be outside the trust-region. Since the iterates of CG grow in norm [CGT00, Theorem 7.5.1], where in our case the norm is that of the tangent space $\|\cdot\|_X$, there is no hope of getting back inside the trust-region with subsequent iterations.

In both cases, it makes sense to move away from ξ_k along p_k as much as we can, i.e., until hitting the boundary of the trust-region (line 8). Regarding the stopping criterion in (line 13), it is technical and serves to ensure superlinear convergence with order $\min(1 + \theta, 2)$. The underlying idea is as follows: considering that the inner problem (3.11) is just a means to an end, excessive time should not be spent trying to solve it too accurately, especially when far from critical points (i.e., if $\|r_0\| = \|\text{grad}f(X)\|_X$ is large), where Newton-like steps are often worse than GD-like steps. Conversely, if close to a critical point (i.e., if $\|r_0\| = \|\text{grad}f(X)\|_X$ is small), it may be worth solving the TRs accurately to achieve the local quadratic convergence typical of Newton's method.

3.3. Riemannian Trust-Region (R-TR)

Algorithm 5 Steihaug-Toint truncated conjugate gradient (tCG) method

Parameters: $\kappa \geq 0, \theta \in]0, 1]$; common choices are $\kappa = 0.1$ and $\theta = 1$

Input: Finite dimensional euclidean space $(T, \langle \cdot, \cdot \rangle)$, symmetric $H : T \rightarrow T$, $g \in T$, radius $\Delta > 0$

Output: Approximate minimizer of $m(\xi) = \frac{1}{2} \langle \xi, H\xi \rangle - \langle g, \xi \rangle$ s.t. $\|\xi\| \leq \Delta$

```

1: Set  $\xi_0 = 0, r_0 = g, p_0 = r_0$ 
2: for  $k = 0, 1, 2, \dots$  do
3:   Compute  $Hp_k$  ▷ This is the only call to H
4:   Compute  $\langle p_k, Hp_k \rangle$ 
5:    $\alpha_{k+1} = \|r_k\|^2 / \langle p_k, Hp_k \rangle$ 
6:    $\xi_k^+ = \xi_k + \alpha_{k+1}p_k$ 
7:   if  $\langle p_k, Hp_k \rangle < 0$  or  $\|\xi_k^+\| \geq \Delta$  then ▷ if H has a negative eigenvalue or the
next CG step is outside the region
8:     Set  $\xi_{k+1} = \xi_k + tp_k$  with  $t \geq 0$  s.t.  $\|\xi_{k+1}\| = \Delta$  ▷ Hit the boundary
9:     return  $\xi_{k+1}$ 
10:  end if
11:   $\xi_{k+1} = \xi_k^+$ 
12:   $r_{k+1} = r_k - \alpha_{k+1}Hp_k$ 
13:  if  $\|r_{k+1}\| \leq \|r_0\| \cdot \min(\|r_0\|^\theta, \kappa)$  then ▷ if the approx. solution is good enough
14:    return  $\xi_{k+1}$ 
15:  end if
16:   $\beta_{k+1} = \|r_{k+1}\|^2 / \|r_k\|^2$ 
17:   $p_{k+1} = r_{k+1} + \beta_k p_k$ 
18: end for

```

Final algorithm The Riemannian Trust-Region algorithm is outlined in Algorithm 6, with pseudocode heavily inspired by [BA15, Algorithm 3]. Starting from the approximate solution $\xi_k \in T_{X_k}\mathcal{M}$ of the TRs, a candidate next iterate $X_k^+ = R_{X_k}(\xi_k)$ is proposed (line 4). If the actual decrease of the objective function is at least a fraction ρ' of the decrease predicted by the model (line 7), where $\rho' \in]0, \frac{1}{4}[$ is a parameter, the candidate step is accepted and $X_{k+1} = X_k^+$ (line 8). Otherwise, it is rejected and $X_{k+1} = X_k$ (line 10). Finally, the trust region radius Δ_k gets: reduced if the model's prediction was too optimistic (lines 12 and 13), enlarged if the model's prediction was good but being constrained within the trust-region prevented further diminishing the objective value (lines 14 and 15), and left unchanged otherwise.

3.3.1 Convergence

The convergence guarantees for Riemannian Trust-Region with tCG are discussed below, with results that are slight modifications of [Bou23a, Proposition 6.25, Theorems 6.29, 6.30] and [AMS08, Theorems 7.4.2, 7.4.10, 7.4.11]. The formulations are adapted to be more immediately applicable to the specific problem at hand. For more general convergence results, the reader is referred to [Bou23a, § 6] and [AMS08, § 7.4].

We begin by discussing the global convergence of R-TR. Assuming that the model

3.3. Riemannian Trust-Region (R-TR)

Algorithm 6 Riemannian Trust-Region (R-TR) method

Require: Riemannian manifold $\mathcal{M}$, smooth objective function $f : \mathcal{M} \rightarrow \mathbb{R}$

Parameters: Maximum radius $\bar{\Delta} > 0$, threshold $\rho' \in]0, 1/4[; \kappa \geq 0, \theta \in]0, 1]$ for tCG

Input: Retraction R , initial point X_0 , initial radius $\Delta_0 \in]0, \bar{\Delta}]$

Output: Sequence $\{X_k\} \subset \mathcal{M}$

```

1: for  $k = 0, 1, 2, \dots$  do
2:   Pick  $H_{X_k} : T_{X_k}\mathcal{M} \rightarrow T_{X_k}\mathcal{M}$  to define the model (3.10)
3:   Compute  $\xi_k \in T_{X_k}\mathcal{M}$  applying tCG (Algorithm 5) to the TRs (3.11)
4:    $X_k^+ = R_{X_k}(\xi_k)$  ▷ Candidate next iterate
5:    $\rho_f = f(X_k) - f(X_k^+)$  ▷ Actual improvement
6:    $\rho_m = m_{X_k}(0) - m_{X_k}(\xi_k)$  ▷ Model improvement
7:   if  $\rho_f/\rho_m > \rho'$  then ▷ If the actual decrease is at least a fraction  $\rho'$  of the
predicted one
8:      $X_{k+1} = X_k^+$  ▷ Accept the step
9:   else ▷ Otherwise
10:     $X_{k+1} = X_k$  ▷ Reject the step
11:  end if
12:  if  $\rho_f/\rho_m < 1/4$  then ▷ If the model made a poor prediction
13:     $\Delta_{k+1} = \Delta_k/4$  ▷ Reduce the trust-region radius
14:  else if  $\rho_f/\rho_m > 3/4$  and  $\|\xi_k\| = \Delta_k$  then ▷ If the model was good, but
the region was too small
15:     $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$  ▷ Enlarge the radius
16:  else
17:     $\Delta_{k+1} = \Delta_k$ 
18:  end if
19: end for

```

Hessians are uniformly bounded, we can state an analogue of Proposition 3.7.

Proposition 3.9 (Global convergence of R-TR with tCG). *Let $\{X_k\}_{k=0}^{+\infty} \subset \mathcal{M}$ be a sequence of points generated by Riemannian Trust-Region with tCG (Algorithm 6). Assume that exists a constant c_0 such that for all iterations k with $\|\text{grad}f(X_k)\|_{X_k} > 0$ it holds*

$$\|H_{X_k}\|_{X_k} \leq c_0, \quad (3.12)$$

where $\|H_{X_k}\|_{X_k}$ denotes the operator norm of $H_{X_k} : T_{X_k}\mathcal{M} \rightarrow T_{X_k}\mathcal{M}$. Furthermore, assume that the iterates are contained in a compact subset of $\mathcal{M}$. Then $\lim_{k \rightarrow +\infty} \|\text{grad}f(X_k)\|_{X_k} = 0$. In particular, if (3.12) holds and the sublevel sets $\{X \in \mathcal{M} : f(X) \leq c\}$ are compact for all $c \in \mathbb{R}$, Riemannian Trust-Region with tCG is globally convergent.

Proof. Combine [Bou23a, Proposition 6.25] and [Bou23a, Proposition 6.31]. Note that [Bou23a, Proposition 6.31] is formulated for $H_X = \text{Hess}\hat{f}_X(0)$, but we use it in [Bou23a, Proposition 6.25] only to verify assumptions A6.5, A6.6 and A6.8, which are independent of the model Hessian. $\square$

3.3. Riemannian Trust-Region (R-TR)

Remark 3.10. Observe that (3.12) holds if the iterates are contained in a compact subset of $\mathcal{M}$ and the map $X \mapsto \|H_X\|_X$ is continuous on $\mathcal{M}$.

Next, we delve into the local convergence properties of R-TR. Specifically, we discuss the local convergence to a non-degenerate minimizer, which corresponds to an analogue of Proposition 3.8.

Proposition 3.11 (Local convergence of R-TR with tCG). *Let $X_\star \in \mathcal{M}$ be a non-degenerate local minimizer of f , i.e., $\text{grad}f(X_\star) = 0$ and $\text{Hess}f(X_\star) \succ 0$. Assume that in a neighborhood of $X_\star$ the model Hessian $H_X : T_X\mathcal{M} \rightarrow T_X\mathcal{M}$ is positive definite and $X \mapsto \|H_X\|_X$, $X \mapsto \|H_X^{-1}\|_X$ are bounded. Then Riemannian Trust-Region with tCG is locally convergent to $X_\star$.*

Proof. Analogous of [AMS08, Theorem 7.4.10] using only local arguments. See Appendix A.1. $\square$

Finally, to achieve local superlinear convergence it is required that H_{X_k} is an "approximate Hessian", in the sense of the following proposition.

Proposition 3.12 (Superlinear convergence of R-TR with tCG). *Let $\{X_k\}_{k=0}^{+\infty} \subset \mathcal{M}$ be a sequence of points generated by Riemannian Trust-Region with tCG (Algorithm 6). Assume that there exists a constant β_H such that*

$$\|H_{X_k} - \text{Hess}(f \circ R_{X_k})(0)\|_{X_k} \leq \beta_H \|\text{grad}f(X_k)\|_{X_k}, \quad (3.13)$$

where $\|\cdot\|_{X_k}$ denotes also the operator norm induced by $\|\cdot\|_{X_k}$. Let $X_\star \in \mathcal{M}$ be a non-degenerate local minimizer of f . If the sequence generated by the algorithm converges to $X_\star$, then there exists a constant $c > 0$ and an index $K \in \mathbb{N}$ such that for all $k > K$

$$\text{dist}(X_{k+1}, X_\star) \leq c \cdot \text{dist}(X_k, X_\star)^{\min(1+\theta, 2)}, \quad (3.14)$$

where $\theta > 0$ is the parameter in the stopping criterion of tCG. Moreover, $\|\text{grad}f(X_k)\|_{X_k}$ also converges to 0, and $f(X_k)$ converges to $f(X_\star)$, both with order at least $\min(1+\theta, 2)$. In particular, with $\theta = 1$ convergence is at least quadratic.

Proof. Follows from [AMS08, Theorem 7.4.11], observing that [AMS08, (7.36)] always holds if $f : \mathcal{M} \rightarrow \mathbb{R}$ is smooth, by [Bou23a, Lemma 10.57]. The statements about the convergence of $\|\text{grad}f(X_k)\|_{X_k}$ and $f(X_k)$ follow from [AMS08, Lemma 7.4.8] and Taylor's theorem. $\square$

3.3.2 R-TR with Riemannian Hessian

To approximate the Hessian of the pullback $\text{Hess}\hat{f}_X(0) = \text{Hess}(f \circ R_X)(0)$, a natural choice for H_X is the Riemannian Hessian

$$H_X = \text{Hess}f(X). \quad (3.15)$$

3.3. Riemannian Trust-Region (R-TR)

As discussed in Section 2.1.3, in general $\text{Hess}f(X) \neq \text{Hess}\hat{f}_X(0)$, but equality holds if the retraction is second order or if X is a critical point (Proposition 2.20).

Convergence The choice $H_X = \text{Hess}f(X)$ is particularly advantageous for the convergence guarantees of R-TR. Assuming that the iterates are contained in a compact set ensures that (3.12) is verified, by smoothness of f . Additionally, in a neighborhood of a non-degenerate minimizer, all the assumptions on the model Hessian in Proposition 3.11 are satisfied, ensuring local convergence of R-TR. Moreover, (3.13) holds with $\beta_H = 0$ for all $X \in \mathcal{M}$ if the retraction is second order, but interestingly it also holds locally independently of the retraction [AMS08, § 7.5.1], with β_H possibly nonzero. This ensures local superlinear convergence of R-TR with Riemannian Hessian.

3.3.3 R-TR with projected Euclidean Hessian

Consider a Riemannian submanifold $\mathcal{M}$ of a Euclidean space $(\mathcal{E}, \langle \cdot, \cdot \rangle)$. As per Proposition 2.7,

$$\text{Hess}f(X)[\xi] = \text{Proj}_X \left(\text{Hess}\bar{f}(X)[\xi] \right) + \mathcal{D}_\xi \left(\text{Proj}_X^\perp(\text{grad}\bar{f}(X)) \right),$$

where $\bar{f}$ is a smooth extension, and $\mathcal{D}_\xi$ is the differential of $X \mapsto \text{Proj}_X$ at X along ξ . We can obtain a model Hessian for R-TR by neglecting the curvature term, i.e., considering the projected Euclidean Hessian

$$H_X = \text{Proj}_X \circ \text{Hess}\bar{f}(X) = \text{Proj}_X \circ \text{Hess}\bar{f}(X) \circ \text{Proj}_X : T_X\mathcal{M} \rightarrow T_X\mathcal{M}. \quad (3.16)$$

Convergence With such a choice of the model Hessian, global convergence guarantees continue to hold: (3.12) is verified if the iterates are contained in a compact set, by smoothness of Proj_X and $\bar{f}$. Similarly, local convergence to a non-degenerate minimizer $X_\star$ is ensured by Proposition 3.11, as long as $\text{Proj}_{X_\star} \left(\text{Hess}\bar{f}(X_\star) \right) : T_{X_\star}\mathcal{M} \rightarrow T_{X_\star}\mathcal{M}$ is positive definite. On the other hand, we do not expect (3.13) to hold in general, given that for a second-order retraction

$$\begin{aligned} \|H_{X_k} - \text{Hess}(f \circ R_{X_k})(0)\|_{X_k} &= \left\| \mathcal{D}_\xi \left(\text{Proj}_{X_k}^\perp(\text{grad}\bar{f}(X_k)) \right) \right\|_{X_k}, \\ \|\text{grad}f(X_k)\|_{X_k} &= \|\text{Proj}_{X_k}(\text{grad}\bar{f}(X_k))\|_{X_k}, \end{aligned}$$

and a priori there is no link between the normal and the tangent components of the gradient. However, under suitable assumptions, (3.13) may still hold as long as the normal component of the gradient is somehow controlled. We defer to Proposition 4.3 for a proof related to the problem at hand. In general, it is reasonable to expect that the projected Euclidean Hessian serves as a viable approximation of the Riemannian Hessian, leading to relatively rapid convergence, albeit not superlinear.

3.4 Preconditioning

3.4.1 What preconditioning is and why it is needed

Let us again consider the resolution of $Ax = b$, where $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite. As already discussed, this system is equivalent to minimizing the cost function

$$q(x) = \frac{1}{2}x^\top Ax - b^\top x. \quad (3.17)$$

Gradient descent with cost function q corresponds to the Richardson iteration, while nonlinear CG with cost function q is equivalent to the conjugate gradient method. It is well known that convergence of these methods is slower the higher $\kappa(A)$ [Saa03, Example 4.1 and § 6.11.3]. Let $P \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix with Cholesky decomposition $P = C^\top C$. Consider the equivalent symmetric positive definite linear system

$$\tilde{A}y = \tilde{b} \quad \text{where} \quad \tilde{A} = C^{-\top}AC^{-1}, \tilde{b} = C^{-\top}b, \quad (3.18)$$

after the change of variables $y = Cx$. The Richardson iteration applied to (3.18) is

$$y_{k+1} = y_k - \alpha_k(\tilde{A}y_k - \tilde{b}) = y_k - \alpha_k C^{-\top}(AC^{-1}y_k - b)$$

and on the variable $x = C^{-1}y$ results in

$$x_{k+1} = x_k - \alpha_k P^{-1}(Ax_k - b) = x_k - \alpha_k P^{-1} \text{grad} q(x_k), \quad (3.19)$$

which is known as *preconditioned Richardson iteration*. This is equivalent to applying gradient descent in $\mathbb{R}^n$ endowed with the metric induced by P . Similarly, the preconditioned conjugate gradient method is derived from applying CG to (3.18) and then transforming back to x , or equivalently from applying nonlinear conjugate gradient with cost function (3.17) in the metric induced by P . Both methods can be expressed solely in terms of P^{-1} , without needing to apply P or to know its Cholesky factorization [Saa03, § 9.2]. The advantage is that convergence now depends on $\kappa(C^{-\top}AC^{-1}) = \kappa(P^{-1}A)$, and if P^{-1} is a good approximation of A^{-1} one can expect $P^{-1}A$ to be well-conditioned.

In the context of nonlinear optimization, preconditioned gradient descent involves the search direction

$$d_k = -\mathcal{P}(X_k)^{-1} \text{grad} f(X_k)$$

where $\mathcal{P}(X_k) : \mathcal{E} \rightarrow \mathcal{E}$ is an invertible linear operator, possibly symmetric positive definite. Analogously, preconditioned nonlinear conjugate gradient employs $d_k = -\mathcal{P}(X_k)^{-1} \text{grad} f(X_k) + \beta_k d_{k-1}$, with adjusted formulas for computing β_k (see [HZ06, § 8]). An effective preconditioner $\mathcal{P}(X)$ is typically derived from an approximation of the Hessian at X . This approach can be justified by the fact that the convergence speed around a non-degenerate minimizer, where the function resembles a strongly convex quadratic, depends on the condition number of the Hessian [WN06, Theorem 3.4].

As a final remark, we point out that in numerical linear algebra a preconditioner for a linear system $Ax = b$ can be any invertible matrix $P \in \mathbb{R}^{n \times n}$, where n is the size of the system. In particular, it is not required for P to be symmetric positive definite. However, we choose to focus on symmetric positive definite preconditioners as they can be interpreted as a change of metric, and so lend themselves to be naturally extended to the Riemannian framework. Additionally, even for linear systems, SPD preconditioners are usually preferred if A is SPD, as it is the case with the linear operator $\mathcal{A}$ in our scenario.

3.4.2 Riemannian projected preconditioned methods

Consider a Riemannian submanifold $\mathcal{M}$ of a Euclidean space $\mathcal{E}$. Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth objective function, and let $\bar{f}$ be any smooth extension of f . The search direction for Riemannian gradient descent is given by

$$\xi_k = -\text{grad}f(X_k) = \text{Proj}_{X_k}(d_k), \quad d_k = -\text{grad}\bar{f}(X_k),$$

which is the projection to the tangent of the search direction d_k for gradient descent on $\mathcal{E}$. Given a preconditioner $\mathcal{P}(X_k)$, the search direction for preconditioned gradient descent on $\mathcal{E}$ is $d_k = -\mathcal{P}(X_k)^{-1}\text{grad}\bar{f}(X_k)$. It appears natural to project the preconditioned search direction to obtain a "geometric version" of the preconditioned method through

$$\xi_k = -\text{Proj}_{X_k}\left(\mathcal{P}(X_k)^{-1}\text{grad}\bar{f}(X_k)\right). \quad (3.20)$$

The corresponding line-search method is referred to as *Riemannian projected preconditioned gradient descent*. In our numerical experiments, we employ an accelerated version of Riemannian projected preconditioned gradient descent, providing heuristic justification through preconditioned R-NLCG. Further discussion is deferred to Section 3.4.4.1.

As previously mentioned in the introduction, we will employ preconditioned Riemannian optimization on the manifold of fixed-rank matrices, with the cost function $f(X) = \frac{1}{2}\langle \mathcal{A}X, X \rangle - \langle X, F \rangle$. Using a point-independent preconditioner $\mathcal{P}(X) = \mathcal{P}$, the resulting update of Riemannian projected preconditioned gradient descent is given by

$$X_{k+1} = R_{X_k}(\alpha_k \xi_k) = \arg \min_{Y \in \mathcal{M}_r} \left\| X_k + \alpha_k \text{Proj}_X(\mathcal{P}^{-1}(F - \mathcal{A}X_k)) - Y \right\|,$$

which is known as the *Riemannian Truncated Preconditioned Richardson Iteration* [KSV16]. This nomenclature will be consistently used throughout. Riemannian projected preconditioned gradient descent is a straightforward generalization of this method.

3.4.3 Riemannian preconditioning

Riemannian projected preconditioned methods are a natural heuristic adaptation of preconditioned Euclidean methods to Riemannian optimization. However, their applicability is confined to Riemannian submanifolds, and they lack a solid theoretical foundation

from a Riemannian perspective. To develop purely Riemannian preconditioned methods, independent of the embedding space $\mathcal{E}$, it is insightful to interpret a positive definite preconditioner as a change of metric. With this perspective in mind, we introduce the following definition of a Riemannian preconditioner.

Definition 3.13. *Let $\mathcal{M}$ be a Riemannian manifold with Riemannian metric $\langle \cdot, \cdot \rangle_X$. A Riemannian preconditioner is a map $\mathcal{P}$ that associates each $X \in \mathcal{M}$ with a symmetric positive definite linear operator $\mathcal{P}_X : T_X \mathcal{M} \rightarrow T_X \mathcal{M}$ such that the metric*

$$\langle \xi, \eta \rangle_{\mathcal{P}_X} := \langle \xi, \mathcal{P}_X \eta \rangle_X \quad \forall \xi, \eta \in T_X \mathcal{M}, \quad (3.21)$$

is a Riemannian metric on $\mathcal{M}$.

Remark 3.14. *An equivalent and somewhat more geometric way to define a Riemannian preconditioner is to consider it as a smooth section of $\text{End}(T\mathcal{M})$ (the vector bundle of endomorphisms of the tangent bundle), such that each $\mathcal{P}_X : T_X \mathcal{M} \rightarrow T_X \mathcal{M}$ is a symmetric positive definite linear operator. However, we preferred the above definition as it better highlights how a preconditioner is de facto a change of the Riemannian metric.*

3.4.4 Preconditioned Riemannian nonlinear conjugate gradient

Let $\mathcal{M}$ be a Riemannian manifold equipped with a Riemannian metric $\langle \cdot, \cdot \rangle_X$, and consider a Riemannian preconditioner $\mathcal{P}$ on $\mathcal{M}$. By *preconditioned* Riemannian nonlinear conjugate gradient (with preconditioner $\mathcal{P}$), we mean applying R-NLCG on $\mathcal{M}$ endowed with the metric $\langle \cdot, \cdot \rangle_{\mathcal{P}_X}$. Akin to its Euclidean counterpart, it is possible to express the preconditioned algorithm exclusively in terms of $\mathcal{P}_X^{-1}$, as we show below.

To implement preconditioned R-NLCG, it is necessary to compute Riemannian gradients in the metric induced by the preconditioner and generalize the computation of β_k . For a smooth function $f : \mathcal{M} \rightarrow \mathbb{R}$, denote $\text{grad}_{\mathcal{P}_X} f(X)$ the Riemannian gradient of f with respect to the metric $\langle \cdot, \cdot \rangle_{\mathcal{P}_X}$. For all $\xi \in T_X \mathcal{M}$ it holds

$$\langle \xi, \text{grad}_{\mathcal{P}_X} f(X) \rangle_{\mathcal{P}_X} = Df(X)[\xi] = \langle \xi, \text{grad} f(X) \rangle_X = \langle \xi, \mathcal{P}_X^{-1} \text{grad} f(X) \rangle_{\mathcal{P}_X}.$$

Hence, by uniqueness of the Riemannian gradient,

$$\text{grad}_{\mathcal{P}_X} f(X) = \mathcal{P}_X^{-1} \text{grad} f(X). \quad (3.22)$$

The search direction is then given by

$$\begin{aligned} \xi_k &= -\text{grad}_{\mathcal{P}_{X_k}} f(X_k) + \beta_k T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \\ &= -\mathcal{P}_{X_k}^{-1} \text{grad} f(X_k) + \beta_k T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}). \end{aligned} \quad (3.23)$$

Choosing $\beta_k = 0$ results in *preconditioned Riemannian gradient descent*. Additionally,

the formulas in (3.9) can be generalized to the metric induced by the preconditioner as

$$\begin{aligned}\beta_k^{HS} &= \frac{\langle \mathcal{P}_{X_k}^{-1} g_k, \mathcal{P}_{X_k}^{-1} g_k \rangle_{\mathcal{P}_{X_k}} - \langle \mathcal{P}_{X_k}^{-1} g_k, T_{X_k \leftarrow X_{k-1}}(\mathcal{P}_{X_{k-1}}^{-1} g_{k-1}) \rangle_{\mathcal{P}_{X_k}}}{\langle \mathcal{P}_{X_k}^{-1} g_k, T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \rangle_{\mathcal{P}_{X_k}} - \langle \mathcal{P}_{X_{k-1}}^{-1} g_{k-1}, \xi_{k-1} \rangle_{\mathcal{P}_{X_{k-1}}}} \\ &= \frac{\langle g_k, \mathcal{P}_{X_k}^{-1} g_k \rangle_{X_k} - \langle g_k, T_{X_k \leftarrow X_{k-1}}(\mathcal{P}_{X_{k-1}}^{-1} g_{k-1}) \rangle_{X_k}}{\langle g_k, T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \rangle_{X_k} - \langle g_{k-1}, \xi_{k-1} \rangle_{X_{k-1}}},\end{aligned}\tag{3.24}$$

and similarly

$$\beta_k^{DY} = \frac{\langle g_k, \mathcal{P}_{X_k}^{-1} g_k \rangle_{X_k}}{\langle g_k, T_{X_k \leftarrow X_{k-1}}(\xi_{k-1}) \rangle_{X_k} - \langle g_{k-1}, \xi_{k-1} \rangle_{X_{k-1}}}.\tag{3.25}$$

Note that also in the computation of β_k , we apply only $\mathcal{P}_{X_k}^{-1}$ and not $\mathcal{P}_{X_k}$. Moreover, we need to apply $\mathcal{P}_{X_k}^{-1}$ only to $\text{grad} f(X_k)$, once per iteration.

Observe that if $X_\star$ is a non-degenerate minimizer, then according to Proposition 2.20 the Riemannian Hessian at $X_\star$ with respect to the metric $\langle \cdot, \cdot \rangle_{\mathcal{P}_X}$ is given by $\text{Hess}_{\mathcal{P}_X} f(X_\star) = \text{Hess}_{\mathcal{P}_X}(f \circ R_{X_\star})(0) = \mathcal{P}_{X_\star}^{-1} \text{Hess} f(X_\star)$. Therefore, when $\mathcal{P}_{X_\star}^{-1}$ is a good approximation of $\text{Hess} f(X_\star)^{-1}$, we can expect a significantly faster local convergence.

3.4.4.1 A preconditioned R-NLCG-like acceleration for the Riemannian projected preconditioned method

A heuristic acceleration technique for Riemannian projected preconditioned gradient descent can be derived from the formulas for preconditioned R-NLCG. Specifically, in eqs. (3.23) to (3.25), we substitute the preconditioned Riemannian gradient $\text{grad}_{\mathcal{P}_X} f(X) = \mathcal{P}_X^{-1} \text{grad} f(X)$ with the projected preconditioned Euclidean gradient $\text{Proj}_X(\mathcal{P}(X)^{-1} \text{grad} \bar{f}(X))$. We refer to the resulting method as *R-NLCG with projected preconditioned search direction* (or *R-NLCG with truncated preconditioned Richardson search direction*). It is crucial to note that this acceleration technique is only heuristically justified. Nevertheless, in practice, we observed faster convergence for the problem at hand.

3.4.5 Preconditioned Riemannian Trust-Region method

Analogously to preconditioned R-NLCG, the *Preconditioned Riemannian Trust-Region method* is Riemannian Trust-Region on $\mathcal{M}$ endowed with the metric $\langle \cdot, \cdot \rangle_{\mathcal{P}_X}$. We will now examine how preconditioning, i.e., changing the metric, impacts on the model, the trust-region subproblem, and the convergence guarantees.

The model If we denote $\text{Hess}_{\mathcal{P}_X} \hat{f}_X(0)$ the Hessian at $0 \in T_X \mathcal{M}$ of $\hat{f}_X = f \circ R_X$ with respect to the metric induced by $\mathcal{P}_X$, it holds

$$\text{Hess}_{\mathcal{P}_X} \hat{f}_X(0) = \mathcal{P}_X^{-1} \text{Hess} \hat{f}_X(0).$$

Note that although a map of the type $\mathcal{P}_X^{-1} \text{Hess} \hat{f}_X(0)$ may not appear to be symmetric, it is indeed symmetric with respect to $\langle \cdot, \cdot \rangle_{\mathcal{P}_X}$. Considering a model Hessian H_X approximating $\text{Hess} \hat{f}_X(0)$, a natural choice for an approximation of the preconditioned Hessian $\text{Hess}_{\mathcal{P}_X} \hat{f}_X(0)$ is given by

$$H_{X, \mathcal{P}_X} = \mathcal{P}_X^{-1} H_X. \quad (3.26)$$

With this choice, the preconditioned model $m_X : T_X \mathcal{M} \rightarrow T_X \mathcal{M}$ is

$$\begin{aligned} m_X(\xi) &= f(X) + \langle \text{grad}_{\mathcal{P}_X} f(X), \xi \rangle_{\mathcal{P}_X} + \frac{1}{2} \langle H_{X, \mathcal{P}_X}[\xi], \xi \rangle_{\mathcal{P}_X} \\ &= f(X) + \langle \mathcal{P}_X^{-1} \text{grad} f(X), \xi \rangle_{\mathcal{P}_X} + \frac{1}{2} \langle \mathcal{P}_X^{-1} H_X[\xi], \xi \rangle_{\mathcal{P}_X} \\ &= f(X) + \langle \text{grad} f(X), \xi \rangle_X + \frac{1}{2} \langle H_X[\xi], \xi \rangle_X, \end{aligned}$$

i.e., the model remains unchanged. This is reasonable since the pullback $\hat{f}_X = f \circ R_X$ is independent of the metric.

The Trust-Region subproblem and tCG The Trust-Region subproblem becomes

$$\min_{\xi \in T_X \mathcal{M}} m_X(\xi) \quad \text{subject to} \quad \|\xi\|_{\mathcal{P}_X} \leq \Delta. \quad (3.27)$$

The only change is in the trust region, which is now a ball in the norm induced by $\mathcal{P}_X$, instead of the norm $\|\cdot\|_X$. To solve (3.27), we employ the truncated Conjugate Gradient (tCG) method in the metric induced by $\mathcal{P}_X$, i.e., preconditioned tCG. Therefore, similar to preconditioned CG, an effective preconditioner can substantially speed up the iterative resolution of the TRs. Moreover, preconditioned tCG can be expressed applying only $\mathcal{P}_X^{-1}$, once per iteration. We refer the reader to [CGT00, Algorithm 7.5.1] for the details.

Convergence We aim to establish that the assumptions of Propositions 3.9, 3.11 and 3.12 result in analogous convergence guarantees for preconditioned R-TR, when employing $H_{X, \mathcal{P}_X} = \mathcal{P}_X^{-1} H_X$ as the model Hessian. This conclusion is derived from the following two lemmas

Lemma 3.15. *Let T be a finite-dimensional Euclidean space equipped with the inner product $\langle \cdot, \cdot \rangle$ and the corresponding norm $\|\cdot\|$. Let $P : T \rightarrow T$ be a symmetric positive definite linear operator (with respect to $\langle \cdot, \cdot \rangle$). Denote $\langle \cdot, \cdot \rangle_P$ the inner product induced by P and $\|\cdot\|_P$ the corresponding induced norm. Let us denote $\|\cdot\|$ and $\|\cdot\|_P$ also the operator norms induced by the vector norms $\|\cdot\|$ and $\|\cdot\|_P$, respectively. For any $g \in T$ and any linear operator $H : T \rightarrow T$, the following holds:*

- i. $\|P^{-1}g\|_P = \|P^{-\frac{1}{2}}g\| \in [\|g\|/\sqrt{\|P\|}, \|g\| \cdot \sqrt{\|P^{-1}\|}]$;
- ii. $\|P^{-1}H\|_P = \|P^{-\frac{1}{2}}HP^{-\frac{1}{2}}\| \leq \|P^{-1}\| \|H\|$;
- iii. $\|HP\|_P = \|P^{\frac{1}{2}}HP^{\frac{1}{2}}\| \leq \|P\| \|H\|$.

Proof. To prove i, observe that, by definition,

$$\|P^{-1}g\|_P^2 = \langle P^{-1}g, PP^{-1}g \rangle = \langle P^{-\frac{1}{2}}g, P^{-\frac{1}{2}}g \rangle = \|P^{-\frac{1}{2}}g\|^2.$$

Using the Courant-Fischer minimax theorem [GV13, Theorem 8.1.2]

$$\frac{\|P^{-\frac{1}{2}}g\|^2}{\|g\|^2} = \frac{\langle g, P^{-1}g \rangle}{\langle g, g \rangle} \in [\lambda_{\min}(P^{-1}), \lambda_{\max}(P^{-1})] = [1/\lambda_{\max}(P), \lambda_{\max}(P^{-1})],$$

and, once again using the Courant-Fischer minimax theorem, one can argue that $\|P\| = \lambda_{\max}(P)$ and $\|P^{-1}\| = \lambda_{\max}(P^{-1})$.

To prove ii, by the definition of induced norm

$$\begin{aligned} \|P^{-1}H\|_P &= \sup_{x \in T \setminus \{0\}} \frac{\|P^{-1}Hx\|_P}{\|x\|_P} = \sup_{x \in T \setminus \{0\}} \frac{\|P^{-\frac{1}{2}}Hx\|}{\|P^{\frac{1}{2}}x\|} \stackrel{y=P^{\frac{1}{2}}x}{=} \sup_{y \in T \setminus \{0\}} \frac{\|P^{-\frac{1}{2}}HP^{-\frac{1}{2}}y\|}{\|y\|} \\ &= \|P^{-\frac{1}{2}}HP^{-\frac{1}{2}}\| \leq \|P^{-\frac{1}{2}}\|^2 \|H\| = \|P^{-1}\| \|H\|, \end{aligned}$$

and a similar argument yields iii. $\square$

Lemma 3.16. *Let $\mathcal{P}$ be a preconditioner on a Riemannian manifold $\mathcal{M}$. The maps $X \mapsto \|\mathcal{P}_X\|_X$ and $X \mapsto \|\mathcal{P}_X^{-1}\|_X$ are continuous from $\mathcal{M}$ to $\mathbb{R}$, where $\|\cdot\|_X$ denotes the operator norm induced by $\langle \cdot, \cdot \rangle_X$.*

Proof. Let $\bar{X} \in \mathcal{M}$ be an arbitrary point on the manifold. We can pass to coordinates with respect to an orthonormal local frame around $\bar{X}$. In coordinates, the preconditioner becomes (locally) a smooth map $X \mapsto P(X) \in \text{Sym}(\dim(\mathcal{M}))$ such that $P(X) \succ 0$. By continuity of the inverse and of matrix norms, it is now clear that the maps $X \mapsto \|\mathcal{P}_X\|_X$ and $X \mapsto \|\mathcal{P}_X^{-1}\|_X$ are continuous. $\square$

Applying Lemma 3.15 with $T = T_X\mathcal{M}$, $P = \mathcal{P}_X$ and $H = H_X$, if (3.12) holds, then

$$\|\mathcal{P}_{X_k}^{-1}H_{X_k}\|_{\mathcal{P}_{X_k}} \leq c_0 \|\mathcal{P}_{X_k}^{-1}\|_{X_k}.$$

If the iterates $\{X_k\}_{k=0}^{+\infty}$ are contained in a bounded set, then $\{\|\mathcal{P}_{X_k}^{-1}\|_{X_k}\}_{k=0}^{+\infty}$ is lower-bounded by continuity of $X \mapsto \|\mathcal{P}_X^{-1}\|_X$. Consequently, Proposition 3.9 holds for preconditioned R-TR under the same assumptions. Similarly, leveraging the continuity of $X \mapsto \|\mathcal{P}_X^{-1}\|_X$ and $X \mapsto \|\mathcal{P}_X\|_X$, which are bounded in a sufficiently small neighborhood of X_* , ensures that the local convergence guarantees in Proposition 3.11 hold under the same hypotheses.

Regarding the local convergence order, if (3.13) holds, then

$$\begin{aligned} \left\| \mathcal{P}_{X_k}^{-1} \mathbf{H}_{X_k} - \text{Hess}_{\mathcal{P}_{X_k}} \hat{f}_{X_k}(0) \right\|_{\mathcal{P}_{X_k}} &= \left\| \mathcal{P}_{X_k}^{-1} \left(\mathbf{H}_{X_k} - \text{Hess} \hat{f}_{X_k}(0) \right) \right\|_{\mathcal{P}_{X_k}} \leq \\ &\leq \beta_H \|\mathcal{P}_{X_k}^{-1}\|_{X_k} \|\text{grad} f(X_k)\|_{X_k} \leq \beta_H \|\mathcal{P}_{X_k}^{-1}\|_{X_k} \|\mathcal{P}_{X_k}\|_{X_k}^{\frac{1}{2}} \|\text{grad}_{\mathcal{P}_{X_k}} f(X_k)\|_{\mathcal{P}_{X_k}}, \end{aligned} \quad (3.28)$$

and Proposition 3.12 also holds under the same assumptions.

3.4.6 Preconditioners of the form $\mathcal{P}_X = \text{Proj}_X \circ \mathcal{P} \circ \text{Proj}_X$

Consider a Riemannian submanifold $\mathcal{M}$ of a Euclidean space $(\mathcal{E}, \langle \cdot, \cdot \rangle)$. Given a symmetric positive definite linear operator $\mathcal{P} : \mathcal{E} \rightarrow \mathcal{E}$, i.e., a preconditioner on $\mathcal{E}$, a valid Riemannian preconditioner on $\mathcal{M}$ can be obtained as

$$\mathcal{P}_X := \text{Proj}_X \circ \mathcal{P} = \text{Proj}_X \circ \mathcal{P} \circ \text{Proj}_X : \text{T}_X \mathcal{M} \rightarrow \text{T}_X \mathcal{M}. \quad (3.29)$$

Notice that the rightmost projection in (3.29) is there just for "cosmetic" purposes, to emphasize symmetry of $\mathcal{P}_X$. The metric induced by $\mathcal{P}_X$ on the tangent $\text{T}_X \mathcal{M}_r$ is

$$\langle \xi, \eta \rangle_{\mathcal{P}_X} = \langle \text{Proj}_X(\mathcal{P}\xi), \eta \rangle^{\eta \in \text{T}_X \mathcal{M}_r} \langle \mathcal{P}\xi, \eta \rangle = \langle \xi, \eta \rangle_{\mathcal{P}} \quad \forall \xi, \eta \in \text{T}_X \mathcal{M},$$

that is, it coincides with the metric induced by $\mathcal{P}$. Therefore, considering $\mathcal{M}$ endowed with the metric induced by $\mathcal{P}_X$ is equivalent to considering $\mathcal{M}$ as a Riemannian submanifold of $(\mathcal{E}, \langle \cdot, \cdot \rangle_{\mathcal{P}})$.

R-TR with projected euclidean Hessian To employ preconditioned R-TR with projected Euclidean Hessian as the model Hessian, two approaches may be considered:

1. Compute the projected Euclidean Hessian considering $\mathcal{M}$ as a Riemannian submanifold of $(\mathcal{E}, \langle \cdot, \cdot \rangle)$. Then, take $\mathbf{H}_{X, \mathcal{P}_X} = \mathcal{P}_X^{-1} \circ \text{Proj}_X \circ \text{Hess} \bar{f}(X)$ as the model Hessian in the metric induced by $\mathcal{P}_X$.
2. Compute the projected Euclidean Hessian considering $\mathcal{M}$ as a Riemannian submanifold of $(\mathcal{E}, \langle \cdot, \cdot \rangle_{\mathcal{P}})$, i.e., take the model Hessian $\mathbf{H}_X^{\mathcal{P}} = \text{Proj}_X^{\mathcal{P}} \circ \mathcal{P}^{-1} \circ \text{Hess} \bar{f}(X)$, where $\text{Proj}_X^{\mathcal{P}}$ denotes the orthogonal projection onto the tangent space with respect to the metric induced by $\mathcal{P}$.

We claim that the two approaches are equivalent, meaning the two R-TR models coincide. Denoting $\bar{\mathbf{H}} = \text{Hess} \bar{f}(X)$, for all $\xi \in \text{T}_X \mathcal{M}$ it holds:

$$\begin{aligned} \langle \mathbf{H}_{X, \mathcal{P}_X}[\xi], \xi \rangle_{\mathcal{P}_X} &= \langle \text{Proj}_X(\bar{\mathbf{H}}[\xi]), \xi \rangle = \langle \bar{\mathbf{H}}[\xi], \xi \rangle = \langle \mathcal{P}^{-1}(\bar{\mathbf{H}}[\xi]), \xi \rangle_{\mathcal{P}} \\ &= \left\langle \text{Proj}_X^{\mathcal{P}} \left(\mathcal{P}^{-1}(\bar{\mathbf{H}}[\xi]) \right), \xi \right\rangle_{\mathcal{P}} = \langle \mathbf{H}_X^{\mathcal{P}}[\xi], \xi \rangle_{\mathcal{P}} = \langle \mathbf{H}_X^{\mathcal{P}}[\xi], \xi \rangle_{\mathcal{P}_X}, \end{aligned}$$

which proves our claim.

Why the Riemannian truncated preconditioned Richardson is not a change of metric Let $f : \mathcal{M} \rightarrow \mathbb{R}$ be a smooth objective function and denote as $\bar{f}$ any smooth extension of f . Considering $\mathcal{M}$ as a Riemannian submanifold of $(\mathcal{E}, \langle \cdot, \cdot \rangle_{\mathcal{P}})$, the search direction of preconditioned R-GD can be computed as

$$\xi_k = -\text{grad}_{\mathcal{P}} f(X_k) = -\text{Proj}_X^{\mathcal{P}}(\text{grad}_{\mathcal{P}} \bar{f}(X_k)) = -\text{Proj}_X^{\mathcal{P}}(\mathcal{P}^{-1} \text{grad} \bar{f}(X_k)).$$

This should be compared with the search direction of the Riemannian truncated preconditioned Richardson iteration

$$\xi_k = -\text{Proj}_X(\mathcal{P}^{-1} \text{grad} \bar{f}(X_k)).$$

Notably, the only difference is in the projection to the tangent. Due to this "wrong projection", the Riemannian truncated preconditioned Richardson iteration cannot be interpreted as a change of metric on the manifold. Indeed, suppose that such a metric exists, and denote it as $\langle \cdot, \cdot \rangle_X$. Then, for all $(X, \xi) \in \text{T}\mathcal{M}$

$$\langle \text{Proj}_X(\text{grad} \bar{f}(X)), \xi \rangle = Df(X)[\xi] = \langle \text{Proj}_X(\mathcal{P}^{-1} \text{grad} \bar{f}(X)), \xi \rangle_X. \quad (3.30)$$

However, for a general positive definite preconditioner $\mathcal{P}$, it could happen that $\text{Proj}_X(\mathcal{P}^{-1} \text{grad} \bar{f}(X))$ is zero even if $\text{Proj}_X(\text{grad} \bar{f}(X))$ is nonzero. An explicit counterexample follows.

Let $\mathcal{M} = \mathcal{M}_r$ be the manifold of rank- r matrices in $\mathbb{R}^{n \times n}$, and let $X = \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \in \mathcal{M}$. By Proposition 2.26

$$\text{T}_X \mathcal{M} = \begin{bmatrix} \mathbb{R}^{r \times r} & \mathbb{R}^{r \times (n-r)} \\ \mathbb{R}^{(n-r) \times r} & 0 \end{bmatrix} \quad \text{and} \quad \text{Proj}_X(Y) = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & 0 \end{bmatrix},$$

where the block structure of Y is the same as X . Let $E \in \mathbb{R}^{n \times n}$ be any symmetric positive definite matrix such that $E_{21}^\top E_{21} \neq 0$, and let us define the positive definite linear operator $\mathcal{P}X = EXE$. If we define

$$\bar{f}(X) = \langle X, C \rangle, \quad \text{where } C = \mathcal{P} \left(\begin{bmatrix} 0 & 0 \\ 0 & I_{n-r} \end{bmatrix} \right) = \begin{bmatrix} E_{21}^\top E_{21} & E_{21}^\top E_{22} \\ E_{22}^\top E_{21} & E_{22}^\top E_{22} \end{bmatrix},$$

then $\text{Proj}_X(\text{grad} \bar{f}(X)) = \text{Proj}_X(C) \neq 0$ (the upper-left block is nonzero), while

$$\text{Proj}_X(\mathcal{P}^{-1} \text{grad} \bar{f}(X)) = \text{Proj}_X(\mathcal{P}^{-1} C) = \text{Proj}_X \left(\begin{bmatrix} 0 & 0 \\ 0 & I_{n-r} \end{bmatrix} \right) = 0.$$

We remark that the Riemannian truncated preconditioned Richardson iteration remains a valid practical alternative. Although it may not provide the same convergence guarantees as preconditioned methods in the Riemannian sense, it proves to be a viable choice, especially when it is computationally more efficient to invert $\mathcal{P}$ (possibly approximately) and then project, rather than inverting $\mathcal{P}_X$. This will become evident later

when $\mathcal{P}$ is a generalized Sylvester operator $\mathcal{P}X = AXD + EXB$, and approximating the inverse of $\mathcal{P}$ is relatively inexpensive with a few iterations of the factored Alternating Direction Implicit (fADI) method, while inverting $\mathcal{P}_X$ requires rather cumbersome and computationally expensive methods. Further details are deferred to Sections 5.2 and 5.3.

Chapter 4

Cost function and convergence for multiterm linear matrix equations

In this chapter, we detail the application of Riemannian optimization techniques to the approximate resolution of multiterm linear matrix equations. We begin by explaining the reasons for preferring the cost function $f(X) = \frac{1}{2} \langle X, \mathcal{A}X \rangle - \langle X, F \rangle$, which stems from the reformulation of equation (1.1) as a distance minimization problem in $\mathcal{A}$ -norm, over minimizing the Frobenius norm of the residual. Furthermore, we provide explicit formulas for efficiently computing the cost, the Riemannian gradient, and the Riemannian Hessian. An initialization strategy for the line-search backtracking procedure is also proposed. Additionally, we discuss the properties of critical points and the convergence guarantees associated with the cost function f . Finally, we outline how fixed-rank Riemannian optimization can lead to a Riemannian Rank Adaptive Method (RRAM) that alternates between fixed-rank optimization and rank updates.

4.1 Cost function, gradient and Hessian

4.1.1 Cost function

Since $\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ is symmetric positive definite, it induces an inner product $\langle X, Y \rangle_{\mathcal{A}} = \langle \mathcal{A}X, Y \rangle$ on $\mathbb{R}^{m \times n}$, with corresponding $\mathcal{A}$ -induced norm $\|X\|_{\mathcal{A}} = \sqrt{\langle X, X \rangle_{\mathcal{A}}}$. We will refer to $\|\cdot\|_{\mathcal{A}}$ as $\mathcal{A}$ -norm or *energy norm*. Let $X_{\star} = \mathcal{A}^{-1}F \in \mathbb{R}^{m \times n}$ be the exact solution of (1.1), then the squared distance in $\mathcal{A}$ -norm of a point $X \in \mathbb{R}^{m \times n}$ from $X_{\star}$ is

$$\begin{aligned} \tilde{f}(X) &= \|X - X_{\star}\|_{\mathcal{A}}^2 = \|X\|_{\mathcal{A}}^2 - 2 \langle X, X_{\star} \rangle_{\mathcal{A}} + \|X_{\star}\|_{\mathcal{A}}^2 \\ &= \langle X, \mathcal{A}X \rangle - 2 \langle X, \mathcal{A}X_{\star} \rangle + \|X_{\star}\|_{\mathcal{A}}^2 = \langle X, \mathcal{A}X \rangle - 2 \langle X, F \rangle + \|X_{\star}\|_{\mathcal{A}}^2. \end{aligned}$$

Therefore, minimizing $\|X - X_{\star}\|_{\mathcal{A}}$ is equivalent to minimizing the cost function

$$\bar{f}(X) = \frac{1}{2} \langle X, \mathcal{A}X \rangle - \langle X, F \rangle, \quad X \in \mathbb{R}^{m \times n}. \quad (4.1)$$

4.1. Cost function, gradient and Hessian

On $\mathcal{E} = \mathbb{R}^{m \times n}$, the unique global minimizer of (4.1) is the exact solution $X_\star$. Similarly, on $\mathcal{M}_r$ we can consider minimization of the cost function defined by restriction of (4.1)

$$f := \bar{f}|_{\mathcal{M}_r} : \mathcal{M}_r \rightarrow \mathbb{R}, \quad f(X) = \frac{1}{2} \langle X, \mathcal{A}X \rangle - \langle X, F \rangle. \quad (4.2)$$

Any (local) minimizer of (4.2) on $\mathcal{M}_r$ is (locally) the point of the manifold closest, in $\mathcal{A}$ -norm, to the exact solution $X_\star$. This is in a sense the best solution of rank r , if the error is measured in the energy norm. One may first ask whether a global minimum on $\mathcal{M}_r$ exists and whether this is also the best solution of rank less than or equal to r . The following result answers this question, and extends [VV10, Proposition 3.2].

Proposition 4.1. *Let $\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ be a linear symmetric positive definite operator. For $F \in \mathbb{R}^{m \times n}$, let $X_\star = \mathcal{A}^{-1}F$ be the exact solution of the linear equation $\mathcal{A}X = F$. Define $\bar{f} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ as $\bar{f}(X) = \frac{1}{2} \langle X, \mathcal{A}X \rangle - \langle X, F \rangle$. Then:*

- i. *if $\text{rank}(X_\star) \geq r$, there exists a minimizer of $\bar{f}$ on $\mathcal{M}_r$. Moreover, any minimizer $\hat{X}$ of $\bar{f}$ on the set of matrices of rank lower than or equal to r has rank exactly r .*
- ii. *if $\text{rank}(X_\star) < r$ there is no minimizer of $\bar{f}$ on $\mathcal{M}_r$.*

Proof. First, let us prove i. Let $\tilde{f}(X) = \|X - X_\star\|_{\mathcal{A}}^2$, since $\tilde{f}(X) = 2\bar{f}(X) + \|X_\star\|_{\mathcal{A}}^2$, replacing $\bar{f}$ with $\tilde{f}$ does not change the minimizers (and their existence). Denote $S = \{X \in \mathbb{R}^{m \times n} : \text{rank}(X) \leq r\}$. Since S is closed, a minimum $\hat{X}$ of $\tilde{f}$ on S exists: it is a point of S nearest to $X_\star$, in $\mathcal{A}$ -norm. Suppose that $\text{rank}(\hat{X}) < r$, then any rank-one perturbation of $\hat{X}$ is still in S and must therefore have a function value greater than or equal to that of $\hat{X}$. Hence, for any rank-one matrix Y and for every $\varepsilon > 0$

$$0 \leq \frac{1}{\varepsilon} \left(\tilde{f}(\hat{X} + \varepsilon Y) - \tilde{f}(\hat{X}) \right) = \varepsilon \|Y\|_{\mathcal{A}}^2 - 2\langle X_\star - \hat{X}, Y \rangle_{\mathcal{A}}.$$

Writing the same equation with $-Y$ instead of Y we get $|\langle X_\star - \hat{X}, Y \rangle_{\mathcal{A}}| \leq \frac{\varepsilon}{2} \|Y\|_{\mathcal{A}}^2$, and by arbitrariness of ε it holds

$$\langle X_\star - \hat{X}, Y \rangle_{\mathcal{A}} = 0 \quad \text{for all } Y \in \mathbb{R}^{m \times n} \text{ s.t. } \text{rank}(Y) = 1.$$

Since the Span of rank-one matrices is the whole $\mathbb{R}^{m \times n}$, it must be $X_\star = \hat{X}$. This contradicts the hypothesis $\text{rank}(X_\star) \geq r$.

To prove ii, observe that since $X_\star \notin \mathcal{M}_r$ then $\tilde{f}(X) > 0 = \tilde{f}(X_\star)$ for all $X \in \mathcal{M}_r$. To conclude, it suffices to construct a sequence $\{X_k\}_{k=1}^{+\infty} \subset \mathcal{M}_r$ such that $X_k \rightarrow X_\star$. Denote $p = \text{rank}(X_\star) < r$ and consider an SVD decomposition $X_\star = U\Sigma V^\top$, where $U \in \text{St}(m, r)$, $V \in \text{St}(n, r)$ and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_p, 0, \dots, 0)$. We can define $X_k := U \text{diag}(\sigma_1, \dots, \sigma_p, \frac{1}{k}, \dots, \frac{1}{k}) V^\top$ and the thesis follows. $\square$

Remark 4.2. *Result i tells us that if $\text{rank}(X_\star) \geq r$ we can minimize on $\mathcal{M}_r$ and still get the best approximation of rank less than and equal to r , not just of rank exactly r . In*

contrast, it tells us that the minimization problem on $\mathcal{M}_r$ is ill-posed if $\text{rank}(X_\star) < r$. However, as the proof shows, on $\mathcal{M}_r$ we can find arbitrarily accurate approximations of $X_\star$, even though these will tend to be numerically rank-deficient.

4.1.1.1 Why not to use the squared Frobenius norm of the residual

Another cost function one can naturally think of is the squared Frobenius norm of the residual, i.e., $\bar{f}(X) = \frac{1}{2} \|\mathcal{A}X - F\|_F^2$. This cost function does not require the operator $\mathcal{A}$ to be symmetric positive definite, but in fact it implicitly leads back to the positive definite case with cost function (4.1), through the normal equations. Indeed, it holds

$$\begin{aligned} \bar{f}(X) &= \frac{1}{2} \|\mathcal{A}X - F\|_F^2 = \frac{1}{2} \langle \mathcal{A}X, \mathcal{A}X \rangle - \langle \mathcal{A}X, F \rangle + \frac{1}{2} \|F\|_F^2 \\ &= \frac{1}{2} \langle X, \mathcal{A}^\top \mathcal{A}X \rangle - \langle X, \mathcal{A}^\top F \rangle + \text{const.} \end{aligned}$$

The operator $\mathcal{A}^\top \mathcal{A}$ is always symmetric positive semidefinite. If we further assume that $\mathcal{A}$ is invertible, $\mathcal{A}^\top \mathcal{A}$ is also positive definite. Then, minimizing the squared Frobenius norm is equivalent to applying the energy norm based approach to the normal equations

$$(\mathcal{A}^\top \mathcal{A})X = \mathcal{A}^\top F. \quad (4.3)$$

In case $\mathcal{A}$ is nonsymmetric or indefinite, the use of normal equations is a standard approach, which has already been applied to solving multiterm linear matrix equations, e.g., in [KS15]. The drawback is that the condition number gets squared, accentuating the need for effective preconditioners. However, if $\mathcal{A}$ has a dominant Sylvester-like term, denoted as $\mathcal{P}$, the dominant term in $\mathcal{A}^\top \mathcal{A}$ takes the form of $\mathcal{P}^\top \mathcal{P}$, losing its Sylvester-like nature due to cross-terms and thus becoming more difficult to use as a preconditioner.

In conclusion, when $\mathcal{A}$ is symmetric positive definite there is no valid reason for using a cost function that relies on the squared Frobenius norm of the residual. While this is necessary if $\mathcal{A}$ is nonsymmetric or indefinite, we can still see it as an energy norm based approach, applied to normal equations.

4.1.1.2 Implementation aspects and complexity

The evaluation of the cost function $f(X)$ (4.2) at $X = U\Sigma V^\top \in \mathcal{M}_r$ can be carried out efficiently because

$$\begin{aligned} f(X) &= \frac{1}{2} \langle X, \mathcal{A}X \rangle - \langle X, F \rangle = \frac{1}{2} \sum_{i=1}^{\ell} \langle X, A_i X B_i^\top \rangle - \langle X, F \rangle \\ &= \frac{1}{2} \sum_{i=1}^{\ell} \langle U\Sigma V^\top, A_i U\Sigma V^\top B_i^\top \rangle - \langle U\Sigma V^\top, F_L F_R^\top \rangle \\ &= \frac{1}{2} \sum_{i=1}^{\ell} \langle \Sigma(V^\top B_i V), (U^\top A_i U)\Sigma \rangle - \langle (F_L^\top U)\Sigma, (F_R^\top V) \rangle. \end{aligned}$$

4.1. Cost function, gradient and Hessian

Since they will also be useful for gradient and Hessian calculations, we store for all $i = 1, \dots, \ell$ the matrices $A_i U \in \mathbb{R}^{m \times r}$, $B_i V \in \mathbb{R}^{n \times r}$, $U^\top A_i U \in \mathbb{R}^{r \times r}$, $V^\top B_i V \in \mathbb{R}^{r \times r}$, as well as $(F_L^\top U) \in \mathbb{R}^{r_F \times r}$, $(F_R^\top V) \in \mathbb{R}^{r_F \times r}$.

The memory requirements are a $\mathcal{O}(\ell r(m+n))$. In addition to the cost of $\mathcal{O}(r_F r(n+m))$ for computing $F_L^\top U$ and $F_R^\top V$, the dominant cost in the evaluation of $f(X)$ is computing the matrices $U^\top A_i U$ and $V^\top B_i V$ for all $i = 1, \dots, \ell$. For each i , this requires r matvecs with A_i to compute $A_i U$, r matvecs with B_i to compute $B_i V$, and finally $\mathcal{O}(r^2(n+m))$ flops to compute $U^\top (A_i U)$ and $V^\top (B_i V)$. Assuming that the matrices A_i and B_i are sparse for all i 's, so that the complexity of each matvec is $\mathcal{O}(m)$ and $\mathcal{O}(n)$ flops, respectively, the total cost for each evaluation of f is $\mathcal{O}(r(\ell r + r_F)(n+m))$ flops.

4.1.2 Gradient

The (Euclidean) gradient of $\bar{f}$ (4.1) at $X \in \mathbb{R}^{m \times n}$ is

$$\text{grad} \bar{f}(X) = \mathcal{A}X - F, \quad (4.4)$$

and if $X = U\Sigma V^\top \in \mathcal{M}_r$ it can be written in factorized form as

$$\text{grad} \bar{f}(X) = YW^\top := \begin{bmatrix} A_1 U \Sigma & \cdots & A_\ell U \Sigma & -F_L \end{bmatrix} \begin{bmatrix} B_1 V & \cdots & B_\ell V & F_R \end{bmatrix}^\top. \quad (4.5)$$

The Riemannian gradient of f (with respect to the Frobenius metric) can be obtained as

$$\text{grad} f(X) = \text{Proj}_X(\text{grad} \bar{f}(X)) = \text{Proj}_X(\mathcal{A}X - F). \quad (4.6)$$

As discussed in Proposition 2.27, to efficiently compute the representation of the Riemannian gradient as a triplet (M, U_p, V_p) , it suffices to efficiently calculate the products $\text{grad} \bar{f}(X)V$ and $\text{grad} \bar{f}(X)^\top U$. It holds

$$\begin{aligned} \text{grad} \bar{f}(X)V &= \sum_{i=1}^{\ell} (A_i U) \Sigma (V^\top B_i V)^\top - F_L (F_R^\top V), \\ \text{grad} \bar{f}(X)^\top U &= \sum_{i=1}^{\ell} (B_i V) \Sigma (U^\top A_i U)^\top - F_R (F_L^\top U), \end{aligned}$$

with all terms in parentheses that have already been calculated and stored during the evaluation of $f(X)$. The dominant cost is given by the calculation of the products $(A_i U) \Sigma (V^\top B_i V)^\top$ and $(B_i V) \Sigma (U^\top A_i U)^\top$ for all i 's, and of the products $F_R (F_L^\top U)$, $F_L (F_R^\top V)$. Since Σ is diagonal, this gives a total complexity of $\mathcal{O}(r(\ell r + r_F)(n+m))$ flops. Note how this is the same asymptotic complexity of evaluating f itself at X . We already anticipate that also Hessian-vector products have a similar complexity.

4.1.3 Hessian

The (Euclidean) Hessian of $\bar{f}$ (4.1) is $\text{Hess}\bar{f}(X) = \mathcal{A}$, i.e.

$$\text{Hess}\bar{f}(X) : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}, \quad \text{Hess}\bar{f}(X)[H] = \mathcal{A}H. \quad (4.7)$$

Given $X = U\Sigma V^\top \in \mathcal{M}_r$ and $\xi \in \text{T}_X \mathcal{M}_r$, $\xi \doteq (M, U_\xi, V_\xi)$, by Proposition 2.28 for the Riemannian Hessian it holds

$$\begin{aligned} \text{Hess}f(X)[\xi] = & \text{Proj}_X(\mathcal{A}\xi) + \\ & + \left(P_U^\perp(\mathcal{A}X - F)V_\xi\Sigma^{-1} \right) V^\top + U \left(P_V^\perp(\mathcal{A}X - F)^\top U_\xi\Sigma^{-1} \right)^\top. \end{aligned} \quad (4.8)$$

Moreover, as already discussed, to efficiently compute the representation of $\text{Hess}f(X)[\xi]$, it suffices to efficiently calculate the products ZV_ξ , $Z^\top U_\xi$, $\dot{Z}V$ and $\dot{Z}^\top U$, where $Z = \text{grad}\bar{f}(X)$ and $\dot{Z} = \text{Hess}\bar{f}(X)[\xi]$ for short. It holds

$$\begin{aligned} \dot{Z}V &= \sum_{i=1}^{\ell} [A_i(UM_\xi) + A_iU_\xi] (V^\top B_iV)^\top + (A_iU) [(B_iV_\xi)^\top V] \\ \dot{Z}^\top U &= \sum_{i=1}^{\ell} [B_i(VM_\xi^\top) + B_iV_\xi] (U^\top A_iU)^\top + (B_iV) [(A_iU_\xi)^\top U] \\ ZV_\xi &= \sum_{i=1}^{\ell} (A_iU)\Sigma [(B_iV)^\top V_\xi] - F_L(F_R^\top V_\xi), \\ Z^\top U_\xi &= \sum_{i=1}^{\ell} (B_iV)\Sigma [(A_iU)^\top U_\xi] - F_R(F_L^\top U_\xi), \end{aligned}$$

with A_iU , $U^\top A_iU$, B_iV , $V^\top B_iV$ that have already been calculated and stored during the evaluation of $f(X)$, for all $i = 1, \dots, \ell$.

Excluding the $2r$ matvecs products with A_i for computing A_iU_ξ , $A_i(UM_\xi)$ and the $2r$ matvecs products with B_i for computing B_iV_ξ , $B_i(VM_\xi^\top)$ for each i , the computational complexity is dominated by products of dense matrices that have a total cost of $\mathcal{O}(r(\ell r + r_F)(n + m))$ flops. Assuming sparsity of the matrices A_i and B_i for all i 's, the cost for each Hessian-vector product is $\mathcal{O}(r(\ell r + r_F)(n + m))$ flops. To compute the projected Euclidean Hessian, it suffices to calculate the products $\dot{Z}V$ and $\dot{Z}^\top U$, with a computational complexity of $\mathcal{O}(\ell r^2(n + m))$ flops. Finally, note that if $B_i = B_i^\top$ and $A_i = A_i^\top$, the cost of computing the Riemannian Hessian can be reduced by observing that $(A_iU)^\top U_\xi = [(A_iU_\xi)^\top U]^\top$ and $(B_iV)^\top V_\xi = [(B_iV_\xi)^\top V]^\top$.

4.2 Line-search initialization

Since the cost function f (4.2) is a strongly convex quadratic, following [Van13; KSV16], an initialization strategy for Armijo backtracking (Algorithm 3) is obtained by conducting an exact line search on the tangent space, neglecting the retraction. For a descent

4.3. Critical points and convergence

direction $\xi_k \in T_{X_k}\mathcal{M}_r$, the initial step-size is given by

$$\bar{\alpha}_k = -\frac{\langle \text{grad}\bar{f}(X_k), \xi_k \rangle}{\langle \mathcal{A}\xi_k, \xi_k \rangle} = -\frac{\langle \text{grad}f(X_k), \xi_k \rangle}{\langle \text{Proj}_{X_k}(\mathcal{A}\xi_k), \xi_k \rangle}. \quad (4.9)$$

In our numerical experiments this proved to be an excellent line-search initialization, requiring almost no backtracking.

Since the Riemannian gradient is already computed, the primary computational cost is the calculation of $\text{Proj}_{X_k}(\mathcal{A}\xi_k)$, i.e., the projected Euclidean gradient at ξ_k . This incurs a cost of $\mathcal{O}(\ell r^2(n+m))$ flops.

4.3 Critical points and convergence

We proceed to analyze the properties of critical points and the convergence of Riemannian optimization algorithms when utilizing the cost function (4.2).

4.3.1 Critical points and residual

In the context of Riemannian optimization, our goal is to attain critical points. Thus, it is important to understand the connection between being a critical point of f on $\mathcal{M}_r$ and serving as an approximate solution to the generalized Sylvester equation (1.1). Being a critical point is metric-independent. Therefore, considering $\mathcal{M}_r$ as a Riemannian submanifold of $\mathbb{R}^{m \times n}$ with the Frobenius metric, we get that $X = U\Sigma V^\top \in \mathcal{M}_r$ is a critical point for f if and only if

$$\text{Proj}_X(\text{grad}\bar{f}(X)) = \text{Proj}_X(\mathcal{A}X - F) = 0,$$

meaning the projection to the tangent space of the residual $R = R(X) := \mathcal{A}X - F$ is zero. As per Proposition 2.27, this condition is equivalent to

$$\begin{cases} U^\top RV = 0 \\ (I_m - UU^\top)RV = 0 \\ (I_n - VV^\top)R^\top U = 0 \end{cases} \iff \begin{cases} U^\top R = 0 \\ RV = 0 \end{cases}.$$

Notably, being a critical point imposes a stronger condition than the Galerkin condition on the residual $U^\top RV = 0$ (or its one-sided counterparts $U^\top R = 0$ and $RV = 0$), typically employed in Galerkin methods for standard Lyapunov and Sylvester equations [Sim16] and multiterm linear matrix equations [SSS16; PSS17; KT10; KS15].

However, it is crucial to note that the vanishing projection of the residual onto the tangent space (or its very small norm) does not ensure anything about the normal component of the residual, which, in general, can be arbitrarily large. While empirical observations suggest that optimizing over matrices of rank r , with r at least as large as the numerical rank of the exact solution $X_\star$, typically results in $\|\text{grad}f(X)\| \rightarrow 0$

implying $\|R\| = \|\mathcal{A}X - F\| \rightarrow 0$, there are no theoretical guarantees that this will occur. This limitation is particularly significant also because computing the Frobenius norm of the residual is computationally demanding, making it impractical to monitor at each iteration during the algorithm's execution. In fact, although when computing the Riemannian gradient we already compute $R = \mathcal{A}X - F = YW^\top$ through the factors $Y \in \mathbb{R}^{m \times (\ell r + r_F)}$ and $W \in \mathbb{R}^{n \times (\ell r + r_F)}$ in (4.5), computing the norm of R in a numerically stable manner requires performing the thin QR factorization of Y and W . This incurs a cost of approximately $2(n+m)(\ell r + r_F)^2$ flops, which scales quadratically with $(\ell r + r_F)$ and can significantly slow down the iteration when ℓ is large.

We briefly mention that an alternative is to estimate the residual rather than calculating it exactly. Notably, $\|R\|_F^2 = \text{tr}(R^\top R)$ and $\|R\|_2^2 = \lambda_{\max}(R^\top R)$, and the cost of multiplication by R and $R^\top$ is linear in $(\ell r + r_F)$. Consequently, norm and trace estimation methods, either iterative or stochastic, can be employed to efficiently estimate the Frobenius norm or 2-norm of the residual [Mey+21; PCK22; GT18; BK21; Hoc13; Hig02]. This estimate can then serve as a stopping criterion, with a more accurate calculation used at the end of the execution to verify the achieved accuracy.

4.3.2 Local convergence

Regarding local convergence, we note that if the exact solution $X_\star = \mathcal{A}^{-1}F$ belongs to $\mathcal{M}_r$ it is a non-degenerate minimizer of f on $\mathcal{M}_r$. Specifically, $\text{grad}\bar{f}(X_\star) = 0$ and, by Proposition 2.7, the Riemannian Hessian coincides with the projected Euclidean Hessian, which is always positive definite. Therefore, both R-GD and R-TR exhibit local convergence to $X_\star$, with the local convergence order of R-TR being at least quadratic if the assumptions of Proposition 3.12 are satisfied.

4.3.2.1 Quadratic convergence of R-TR with projected Euclidean Hessian

It is known that (3.13) is satisfied if the model Hessian H_X is the Riemannian Hessian. We now prove that, for the cost function (4.2), this condition also holds for the projected Euclidean Hessian in a neighborhood of the exact solution $X_\star \in \mathcal{M}_r$. Consequently, this ensures local quadratic convergence to $X_\star$ for R-TR with projected Euclidean Hessian.

Proposition 4.3. *Let $f : \mathcal{M}_r \rightarrow \mathbb{R}$ be the cost function (4.2), where $\mathcal{M}_r$ is equipped with the Frobenius metric. Suppose $X_\star = \mathcal{A}^{-1}F \in \mathcal{M}_r$. Then (3.13) holds with the model Hessian being the projected Euclidean Hessian. Specifically, there exists a neighborhood $\mathcal{U} \subset \mathcal{M}_r$ of $X_\star$ and a constant $\beta_H > 0$ such that for all $X \in \mathcal{U}$:*

$$\|\text{Proj}_X(\text{Hess}\bar{f}(X)) - \text{Hess}(f \circ R_X)(0)\|_X \leq \beta_H \|\text{grad}f(X)\|_X,$$

Proof. As discussed in Section 3.3.2, there exists a neighborhood $\mathcal{U}' \subset \mathcal{M}_r$ of $X_\star$ and a constant $\beta'_H > 0$ such that $\|\text{Hess}f(X) - \text{Hess}(f \circ R_X)(0)\|_X \leq \beta'_H \|\text{grad}f(X)\|_X$ for all

$X \in \mathcal{U}'$. Therefore, for all $X \in \mathcal{U}'$:

$$\begin{aligned} & \|\text{Proj}_X(\text{Hess}\bar{f}(X)) - \text{Hess}(f \circ R_X)(0)\|_X \\ & \leq \|\text{Proj}_X(\text{Hess}\bar{f}(X)) - \text{Hess}f(X)\|_X + \|\text{Hess}f(X) - \text{Hess}(f \circ R_X)(0)\|_X \\ & \leq \left\| \mathcal{D}_\xi \left(\text{Proj}_X^\perp(\text{grad}\bar{f}(X)) \right) \right\|_X + \beta'_H \|\text{grad}f(X)\|_X. \end{aligned}$$

By Proposition 2.30 and observing that $\text{grad}\bar{f}(X) = \mathcal{A}X - F = \mathcal{A}(X - X_\star)$, one obtains

$$\begin{aligned} \left\| \mathcal{D}_\xi \left(\text{Proj}_X^\perp(\text{grad}f(X)) \right) \right\|_X & \leq \frac{\sigma_{\max}(\text{Proj}_X^\perp(\text{grad}f(X)))}{\sigma_r(X)} \leq \frac{\|\text{Proj}_X^\perp(\mathcal{A}(X - X_\star))\|_F}{\sigma_r(X)} \\ & \leq \frac{\|\mathcal{A}(X - X_\star)\|_F}{\sigma_r(X)} \leq \frac{\|\mathcal{A}\|_F \cdot \|X - X_\star\|_F}{\sigma_r(X)} \leq \frac{\|\mathcal{A}\|_F \cdot \text{dist}(X, X_\star)}{\sigma_r(X)}, \end{aligned}$$

where $\|\mathcal{A}\|_F$ denotes the operator norm induced by the Frobenius norm. Up to restricting the neighborhood $\mathcal{U}'$ to a smaller neighborhood $\mathcal{U}''$, we can suppose $\sigma_r(X) \geq \frac{1}{2}\sigma_r(X_\star)$. Since $X_\star \in \mathcal{M}_r$ is a non-degenerate local minimizer of f , by [AMS08, Lemma 7.4.8], there exists a neighborhood $\mathcal{U} \subseteq \mathcal{U}''$ and constants $c_1 > c_0 > 0$ such that, for all $X \in \mathcal{U}$,

$$c_0 \cdot \text{dist}(X, X_\star) \leq \|\text{grad}f(X)\|_X \leq c_1 \cdot \text{dist}(X, X_\star).$$

Therefore, in $\mathcal{U}$ the thesis holds with $\beta_H = \beta'_H + \frac{2\|\mathcal{A}\|_F}{c_0 \cdot \sigma_r(X_\star)}$. $\square$

4.3.3 Global convergence with regularized cost function

To obtain global convergence guarantees, as per Propositions 3.7 and 3.9, we need to ensure that the iterates remain within a compact subset of $\mathcal{M}_r$. Since both R-GD and R-TR are descent methods, all iterates stay within the sublevel set at the initial point

$$\mathcal{L}_{X_0} = \{X \in \mathcal{M}_r : f(X) \leq f(X_0)\} = \{X \in \mathcal{M}_r : \|X - X_\star\|_{\mathcal{A}} \leq \|X_0 - X_\star\|_{\mathcal{A}}\}.$$

While all sublevel sets are bounded, they might not be closed because $\mathcal{M}_r$ itself is not closed. This occurs, in particular, if $\text{rank}(X_\star) < r$, as each sublevel set contains points that are arbitrarily close to $X_\star \notin \mathcal{M}_r$. This provides insights into the issues that may arise: iterates of R-GD or R-TR must accumulate somewhere in

$$\overline{\mathcal{L}}_{X_0} = \{X \in \mathbb{R}^{m \times n} : \text{rank}(X) \leq r, \|X - X_\star\|_{\mathcal{A}} \leq \|X_0 - X_\star\|_{\mathcal{A}}\} \supset \mathcal{L}_{X_0},$$

but it might happen that they accumulate around a point with rank strictly lower than r (and in a sense we hope this to happen if $\text{rank}(X_\star) < r$).

While using the cost function f may not guarantee that the iterates of R-GD and R-TR stay within a compact subset of $\mathcal{M}_r$, we prove that this is achievable by applying R-GD and R-TR with a regularized cost function. A similar approach has been considered in [Van13] for matrix completion problems. To keep away from matrices of rank smaller than r , we employ regularization with the norm of the pseudo-inverse, resulting in the

cost function

$$g = \bar{g}|_{\mathcal{M}_r}, \quad \bar{g}(X) = \bar{f}(X) + \frac{1}{2}\mu^2\|X^\dagger\|_{\mathbb{F}}^2.$$

Although the gradient and Hessian of $\bar{g}$ can be computed using [GP73, Theorem 4.3], these details are omitted as, in practice, we will still use the original cost function f . The sublevel sets of $\bar{g}$ are both bounded and closed, because being in the level set of $\bar{g}$ imposes a bound on the smallest singular value. Specifically, if $g(X) \leq c$ then

$$\begin{aligned} c &\geq g(X) = f(X) + \frac{1}{2}\mu^2\|X^\dagger\|_{\mathbb{F}}^2 = \frac{1}{2} \left(\|X - X_\star\|_{\mathcal{A}}^2 - \|X_\star\|_{\mathcal{A}}^2 + \mu^2\|X^\dagger\|_{\mathbb{F}}^2 \right) \\ &\geq \frac{1}{2} \left(\mu^2\|X^\dagger\|_{\mathbb{F}}^2 - \|X_\star\|_{\mathcal{A}}^2 \right) \end{aligned}$$

and, consequently,

$$\sigma_{\min}(X) \geq \frac{1}{\sqrt{\|X^\dagger\|_{\mathbb{F}}^2}} \geq \frac{\mu}{\sqrt{2c + \|X_\star\|_{\mathcal{A}}^2}}.$$

While any small $\mu^2 > 0$ theoretically guarantees global convergence, in practice, we will still employ the original objective function f (i.e., $\mu = 0$). Notably, in our numerical experiments, using random initialization we consistently observed convergence of the gradient norm to zero. This approach, where convergence is proven for a regularized cost function, yet optimization is performed without regularization justifying it by the fact that μ^2 can be taken small at will, is a common practice in the literature. Examples of this approach can be found in works such as [Van13; BA15; BM03].

4.4 Rank adaptivity

Up to this point, our focus has been on Riemannian optimization at a fixed rank r . However, a notable weakness of this approach is that, in practical applications, the decay of the singular values of the exact solution is rarely known a priori, and determining a suitable value of r can be challenging. If r is set too low, the exact solution cannot be adequately approximated. On the other hand, if r is set too high, the cost of optimization unnecessarily grows in both time and memory and, additionally, the fact that iterates become numerically rank deficient can cause numerical issues when computing the Riemannian Hessian.

This motivates the development of rank-adaptive techniques that combine fixed-rank Riemannian optimization with rank updates during the optimization process. Rank-adaptive methods have been developed previously in, e.g., [GA22; Zho+16; UV14; UV15; KSV16; Tan+14; Mis+14; KSU14], often combined with optimization over the real algebraic variety of bounded-rank matrices. This allows for better global convergence guarantees, since the matrices of rank at most r are a closed [Zho+16; SU15]. We limit ourselves to a more heuristic approach, which simply allows us not to need to fix the rank a priori and enables a fairer comparison with other methods in the literature for solving multiterm matrix equations.

Our Riemannian Rank-Adaptive Method (RRAM) for multiterm linear matrix equations is summarized in Algorithm 7. It alternates between fixed-rank optimization and rank updates. Starting from an initial iterate X_0 of rank $r = r_0$, we run Riemannian optimization with fixed rank r , employing any of the algorithms outlined in Chapter 3 (line 4). If the current iterate becomes numerically rank-deficient, i.e., if for a user-defined tolerance $\varepsilon_\sigma \geq 0$ it holds $\sigma_r^2 / \sum_{i=1}^r \sigma_i^2 < \varepsilon_\sigma^2$, the iterate is truncated to rank $r_- = \max\{k : \sum_{i=k+1}^r \sigma_i^2 / \sum_{i=1}^r \sigma_i^2 \geq \varepsilon_\sigma^2\}$, and the optimization restarts with rank r_- (lines 6 to 8). Upon convergence of the fixed-rank optimization, we calculate the norm of the residual (line 12). If it remains excessively high, we increment the rank for Riemannian optimization to $r_+ = r + r_{\text{up}}$, utilizing the previous solution $X_k \in \mathcal{M}_r$ as a warm start for Riemannian optimization on $\mathcal{M}_{r_+}$ (lines 14 and 15).

Algorithm 7 Riemannian Rank-Adaptive Method (RRAM)

Parameters: Tolerance for truncation $\varepsilon_\sigma \in]0, 1[$, update rank r_{up} , algorithm for fixed-rank Riemannian optimization.

Input: Initial point $X_0 \in \mathcal{M}_{r_0}$, tolerance on the relative residual $\text{tol} > 0$

```

1: Set  $r = r_0$ ,  $\text{res} = \|\mathcal{A}X_0 - F\|_F / \|F\|_F$ ,  $k = 0$ 
2: while  $\text{res} > \text{tol}$  do
3:   while fixed-rank optimization does not converge do
4:     Get  $X_{k+1} = U\Sigma V^\top \in \mathcal{M}_r$   $\triangleright$  One step of Riemannian opt. at rank  $r$ 
5:     if (fixed-rank optimization did not converge) and  $(\sigma_r^2 / \sum_{i=1}^r \sigma_i^2 < \varepsilon_\sigma^2)$  then
6:        $r_- = \max\{k : \sum_{i=k+1}^r \sigma_i^2 / \sum_{i=1}^r \sigma_i^2 \geq \varepsilon_\sigma^2\}$ 
7:        $X_{k+1} = U(:, 1 : r_-)\Sigma(1 : r_-, 1 : r_-)V(:, 1 : r_-)^\top$ 
8:        $r = r_-$ 
9:     end if
10:     $k = k + 1$ 
11:  end while
12:   $\text{res} = \|\mathcal{A}X_k - F\|_F / \|F\|_F$ 
13:  if  $\text{res} > \text{tol}$  then
14:    From  $X_k$  compute a "warm start" for optimization with rank  $r_+ = r + r_{\text{up}}$ 
15:     $r = r_+$ 
16:  end if
17: end while

```

Rank increase via normal correction Denote the current iterate as $X \in \mathcal{M}_r$. Following the approach outlined in [GA22; UV15; UV14], for computing a warm-start with an increased rank $r_+ = r + r_{\text{up}}$, we employ a normal correction. In other words, we conduct a line-search along the truncation at rank r_{up} of the component of the negative Euclidean gradient $-\text{grad}\bar{f}(X) = -(\mathcal{A}X - F)$ orthogonal to the tangent space. This yields the update $X \leftarrow X + \alpha_* Y_*$, where

$$Y_* \in \arg \min_{\text{rank}(Y)=r_{\text{up}}} \left\| Y - \text{Proj}_X^\perp(F - \mathcal{A}X) \right\|. \quad (4.10)$$

The search direction (4.10) is well-defined if $\text{Proj}_X^\perp(F - \mathcal{A}X)$ has rank at least r_{up} . If this is not the case, we add random components orthogonal to the tangent until we reach the desired rank r_{up} .

As per [GA22, Proposition 2], $X + \alpha_* Y_*$ is guaranteed to have a rank exactly $r + r_{\text{up}}$ for any $\alpha_* > 0$. Given that the cost function f is a strongly convex quadratic, the step-size α_* is determined through an exact line-search as

$$\alpha_* = -\frac{\langle \text{grad} \bar{f}(X), Y_* \rangle}{\langle \mathcal{A}Y_*, Y_* \rangle} = \frac{\|Y_*\|_{\text{F}}^2}{\langle \mathcal{A}Y_*, Y_* \rangle}.$$

As emphasized in [UV15; UV14], this update step can be viewed as an approximate gradient descent step in the ambient space, truncated at rank r_{up} . Specifically, if X is a critical point of $f : \mathcal{M}_r \rightarrow \mathbb{R}$, then the Euclidean gradient $\text{grad} f(X) = \mathcal{A}X - F$ belongs to the normal space, and the search direction Y_* coincides with the truncation at rank r_{up} of the negative gradient.

Implementation aspects and complexity Let $X = U\Sigma V^\top$, and let $R = \mathcal{A}X - F$ denote the residual. The computation of the rank update involves determining the rank- r_{up} TSVD of $\text{Proj}_X^\perp(R) = (I_m - UU^\top)R(I_n - VV^\top)$. In the computation of the Riemannian gradient at X , we already calculate $R = \mathcal{A}X - F = YW^\top$ with $Y \in \mathbb{R}^{m \times (\ell r + r_F)}$ and $W \in \mathbb{R}^{n \times (\ell r + r_F)}$. Hence, the cost of computing the factorization

$$\text{Proj}_X^\perp(R) = \underbrace{\left[Y - U(U^\top Y) \right]}_{\tilde{Y}} \underbrace{\left[W - V(W^\top V)^\top \right]}_{\tilde{W}}^\top$$

is $\mathcal{O}(r(m+n)(\ell r + r_F))$. However, computing the thin SVD of $\text{Proj}_X^\perp(R)$ involves the thin QR factorization of $\tilde{Y}$ and $\tilde{W}$, incurring a cost of approximately $2(m+n)(\ell r + r_F)^2$ flops, quadratic in $(\ell r + r_F)$. It is important to note that these calculations can be performed only once and should not be repeated when calculating the residual norm, which can be written as

$$\|R\|_{\text{F}}^2 = \|\text{grad} f(X)\|_{\text{F}}^2 + \|\text{Proj}_X^\perp(R)\|_{\text{F}}^2.$$

An alternative to achieve a truncation cost linear in $(\ell r + r_F)$ would be to employ randomized TSVD [YGL18; HMT11].

While not explicitly included in the pseudocode for simplicity, in fixed-rank optimization we employ an additional stopping criterion heuristic to partially reduce stagnation. Specifically, given a parameter $\gamma \in (0, 1]$, we halt fixed-rank optimization (and possibly increase the rank) if

$$\|\text{grad} f(X_k)\|_{X_k} \geq \gamma \cdot \max \left\{ \|\text{grad} f(X_{k-1})\|_{X_{k-1}}, \dots, \|\text{grad} f(X_{k-p})\|_{X_{k-p}} \right\},$$

indicating insufficient decrease in the gradient norm, with a maximum "patience" of p

iterations.

Regarding the rank-decrease step, it is worth noting that alternative strategies have been explored in the literature, e.g., based on relative gap between successive singular values [GA22] or the ratio with the largest singular value [Zho+16].

Chapter 5

Preconditioners for multiterm linear matrix equations

The goal of this chapter is to develop preconditioners for optimizing the cost function (4.2) on the manifold of fixed-rank matrices. As discussed earlier, effective preconditioners are typically derived from an approximation of the Riemannian Hessian. From Section 4.1.3, we know that the Riemannian Hessian of f

$$\text{Hess}f(X)[\xi] = \text{Proj}_X(\mathcal{A}\xi) + \left(P_{\tilde{U}}^\perp(\mathcal{A}X - F)V_\xi\Sigma^{-1} \right) V^\top + U \left(P_V^\perp(\mathcal{A}X - F)^\top U_\xi\Sigma^{-1} \right)^\top,$$

is composed of two parts: the projection of the Euclidean Hessian, and a term related to the curvature of the manifold. If the condition number of Σ^{-1} is not excessively large (i.e., X is not numerically rank-deficient), most of the ill-conditioning of $\text{Hess}f(X)$ can be expected to come from $\mathcal{A}$ itself. Therefore, a suitable preconditioner would be the projected Euclidean Hessian

$$\mathcal{A}_X = \text{Proj}_X \circ \mathcal{A} \circ \text{Proj}_X. \quad (5.1)$$

However, in general, solving equations with $\mathcal{A}_X$ may not be feasible, especially if $\mathcal{A}$ is a multiterm Sylvester operator with three or more terms. Similarly to the approximate Newton method in [KSV16, § 4.1], given a positive definite preconditioner $\mathcal{P}$ for $\mathcal{A}$, as a second approximation step we replace $\mathcal{A}$ with $\mathcal{P}$ in (5.1) and consider the positive definite Riemannian preconditioner

$$\mathcal{P}_X = \text{Proj}_X \circ \mathcal{P} \circ \text{Proj}_X. \quad (5.2)$$

All preconditioners we use follow this pattern, and the observations in Section 3.4.6 apply. The convergence guarantees discussed in Section 4.3 extend to the preconditioned case. Additionally, we can also employ the Riemannian truncated preconditioned Richardson iteration described in Section 3.4.2.

Note that R-TR with projected Euclidean Hessian was introduced precisely to cir-

cumvent this second approximation step in which $\mathcal{A}$ is replaced by $\mathcal{P}$. In fact, the search direction of preconditioned R-GD with preconditioner $\mathcal{A}_X$ is $\xi = -\mathcal{A}_X^{-1} \text{grad} f(X)$, resembling the search direction of the approximate Newton method in [KSV16], where the Riemannian Hessian is approximated by the projected Euclidean Hessian $\text{Proj}_X \circ \text{Hess} \bar{f}(X) = \mathcal{A}_X$. While we are unable to apply exactly $\mathcal{A}_X^{-1}$ for a generic $\mathcal{A}$, one could employ iterative methods, possibly preconditioned, to solve the corresponding linear system. This mirrors the approximate solution of the trust-region subproblem with model Hessian $H_X = \text{Proj}_X \circ \text{Hess} \bar{f}(X) = \mathcal{A}_X$ using (preconditioned) tCG. There is therefore a close connection between R-GD with preconditioner $\mathcal{A}_X$, i.e., the approximate Newton method from [KSV16], and R-TR with projected Euclidean Hessian.

The preconditioner $\mathcal{P}$ is typically derived from a dominant term in $\mathcal{A}$, and has a Sylvester-like structure. Therefore, our ultimate goal is to consider preconditioners of the form

$$\mathcal{P}X = AXD + EXB \quad \text{where } A, B, D, E \succ 0. \quad (5.3)$$

Extending beyond two terms in $\mathcal{P}$ seems unlikely to lead to a computationally efficient preconditioner to invert. Indeed, even a preconditioner of the form (5.3) with $E \neq I_m$ or $D \neq I_n$ poses significant challenges, as we will see. To move towards using preconditioners derived from Sylvester operators like (5.3), we proceed step by step. Initially, we consider $\mathcal{P}X = EXD$. Subsequently, we extend the techniques in [VV10, § 7.2] from Lyapunov to Sylvester operators $\mathcal{P}X = AX + XB$. By combining these two types of preconditioners, we can then address the case $\mathcal{P}X = AXD + EXB$. Finally, we propose an ADI-like iteration to approximate the inverse of $\mathcal{P}_X$.

5.1 $\mathcal{P}X = EXD$ where $E, D \succ 0$

In the pursuit of running preconditioned Riemannian optimization on $\mathcal{M}_r$ endowed with the metric induced by $\mathcal{P}X = EXD$ — which coincides with the metric induced by the Riemannian preconditioner $\mathcal{P}_X = \text{Proj}_X \circ \mathcal{P} \circ \text{Proj}_X$ — two alternatives are available. First, as outlined in Sections 3.4.4 and 3.4.5, one can execute preconditioned methods solely through the action of $\mathcal{P}_X^{-1}$ while still leveraging all the Riemannian optimization tools from Section 2.1. As a second alternative, $\mathcal{M}_r$ can be directly considered as a Riemannian submanifold of $\mathbb{R}^{m \times n}$ endowed with the metric induced by $\mathcal{P}$, and all Riemannian optimization tools can be developed within this metric. In this section, both approaches will be considered, emphasizing the similarities and differences. We remark that, although the second approach may initially appear more intricate, it proves particularly useful when preconditioning with $\mathcal{P}X = AXD + EXB$.

5.1.1 Applying $\mathcal{P}_X^{-1}$

Applying $\mathcal{P}_X^{-1}$ to $\eta \in \text{T}_X \mathcal{M}_r$ is equivalent to solving

$$\text{Proj}_X(E\xi D) = \eta \quad \xi \in \text{T}_X \mathcal{M}_r. \quad (5.4)$$

Let $\eta \doteq (M_\eta, U_\eta, V_\eta)$ and $\xi \doteq (M_\xi, U_\xi, V_\xi)$, by Proposition 2.27 the linear equation (5.4) is equivalent to

$$U_\eta + UM_\eta = (EU)M_\xi(V^\top DV) + EU_\xi(V^\top DV) + (EU)V_\xi^\top(DV), \quad (5.5a)$$

$$V_\eta + VM_\eta^\top = (DV)M_\xi^\top(U^\top EU) + DV_\xi(U^\top EU) + (DV)U_\xi^\top(EU), \quad (5.5b)$$

$$M_\eta = (U^\top E)U_\xi(V^\top DV) + (U^\top EU)V_\xi^\top(V^\top D)^\top + (U^\top EU)M_\xi(V^\top DV). \quad (5.5c)$$

From (5.5a) and (5.5b) one obtains

$$E^{-1}(U_\eta + UM_\eta) = U \left[M_\xi(V^\top DV) + V_\xi^\top DV \right] + U_\xi(V^\top DV),$$

$$D^{-1}(V_\eta + VM_\eta^\top) = V \left[M_\xi^\top(U^\top EU) + U_\xi^\top EU \right] + V_\xi(U^\top EU),$$

and therefore U_ξ and V_ξ can be computed as

$$\begin{aligned} U_\xi &= P_U^\perp E^{-1}(U_\eta + UM_\eta)(V^\top DV)^{-1}, \\ V_\xi &= P_V^\perp D^{-1}(V_\eta + VM_\eta^\top)(U^\top EU)^{-1}. \end{aligned} \quad (5.6)$$

Then one can plug the solutions into (5.5c) to obtain M_ξ

$$M_\xi = (U^\top EU)^{-1} \left[M_\eta - (U^\top E)U_\xi(V^\top DV) - (U^\top EU)V_\xi^\top(V^\top D)^\top \right] (V^\top DV)^{-1}. \quad (5.7)$$

Implementation aspects and complexity In addition to the r linear systems with the sparse matrices E and D , the dominant costs involve solving m linear systems with the dense $r \times r$ matrix $U^\top EU$ and n linear systems with the dense $r \times r$ matrix $V^\top DV$. After a Cholesky decomposition of $U^\top EU$ and $V^\top DV$, incurring a cost of $\mathcal{O}(r^3)$ flops, the overall complexity of the linear systems with $U^\top EU$ and $V^\top DV$ amounts to $\mathcal{O}(r^2(m+n))$ flops. Notably, this asymptotic complexity is the same as that of a retraction or a vector transport. Assuming optimal complexities of $\mathcal{O}(m)$ for system resolution with E and $\mathcal{O}(n)$ for system resolution with D , the total complexity remains $\mathcal{O}(r^2(m+n))$ operations. For enhanced efficiency in the solution of the linear systems, it is advisable to precompute a Cholesky factorization of the matrices E and D , possibly after reordering to avoid excessive fill-in.

The one-sided case If $E = I_m$ or $D = I_n$, the derivations above simplify. For instance, considering the case $D = I_n$, we obtain from (5.5a) and (5.5b) alone:

$$M_\xi = U^\top E^{-1}(UM_\eta + U_\eta), \quad (5.8a)$$

$$U_\xi = P_U^\perp E^{-1}(UM_\eta + U_\eta), \quad (5.8b)$$

$$V_\xi = V_\eta(U^\top EU)^{-1}, \quad (5.8c)$$

and one can verify that (5.5c) is also satisfied. The asymptotic complexity remains $\mathcal{O}(r^2 \cdot (m+n))$, but the number of flops is roughly halved. Observe that the computation

of $E^{-1}(UM_\eta + U_\eta)$ needs to be executed only once. Similar derivations are obtained when $E = I_m$.

5.1.2 Geometry with the metric induced by $\mathcal{P}X = EXD$

We now consider the manifold of rank- r matrices $\mathcal{M}_r$ as a Riemannian submanifold of $\mathbb{R}^{m \times n}$ equipped with the metric induced by $\mathcal{P}$. The metric induced by $\mathcal{P}$ is denoted as $\langle \cdot, \cdot \rangle_{\mathcal{P}}$. Within this metric, we use the notation $\text{Proj}_X^{\mathcal{P}}$ for the orthogonal projector onto the tangent space $T_X \mathcal{M}_r$, $P_{\mathcal{M}_r}^{\mathcal{P}}$ for the metric projection onto the manifold, $R^{\mathcal{P}}$ for the metric projection retraction, $\text{grad}_{\mathcal{P}}$ for the gradient and $\text{Hess}_{\mathcal{P}}$ for the Hessian (whether Euclidean or Riemannian will be clear from the domain of the function).

5.1.2.1 The Cholesky decomposition of $\mathcal{P}$

Given the Cholesky decompositions $D = C_D^{\top} C_D$ and $E = C_E^{\top} C_E$, the Cholesky decomposition of $\mathcal{P}$ is expressed as

$$\mathcal{P} = C^{\top} C, \quad CX = C_E X C_D^{\top}. \quad (5.9)$$

Observe that $\mathcal{C} : \mathcal{M}_r \rightarrow \mathcal{M}_r$ is a (linear) diffeomorphism. The following lemmas will prove useful.

Lemma 5.1. $T_{\mathcal{C}X} \mathcal{M}_r = \mathcal{C}(T_X \mathcal{M}_r)$ for all $X \in \mathcal{M}_r$.

Proof. Since $\mathcal{C}$ is an invertible linear map, it suffices to show that $\mathcal{C}T_X \mathcal{M}_r \subseteq T_{\mathcal{C}X} \mathcal{M}_r$ and equality follows by a dimensionality argument. Let $\xi \in T_X \mathcal{M}_r$, then there exists a smooth curve $\gamma : \mathbb{R} \rightarrow \mathcal{M}_r$ such that $\gamma(0) = X$, $\gamma'(0) = \xi$. Let $\alpha := \mathcal{C} \circ \gamma : \mathbb{R} \rightarrow \mathcal{M}_r$, then α is a smooth curve on $\mathcal{M}_r$ passing through $\mathcal{C}X$ at time 0, hence $\alpha'(0) = \mathcal{C}\gamma'(0) = \mathcal{C}\xi \in T_{\mathcal{C}X} \mathcal{M}_r$. $\square$

Lemma 5.2. $\mathcal{C}^{-1}P_{\mathcal{M}_r}(X) = P_{\mathcal{M}_r}^{\mathcal{P}}(\mathcal{C}^{-1}X)$ for all $X \in \mathbb{R}^{m \times n}$ sufficiently close to the manifold, so that both terms are well-defined.

Proof.

$$\begin{aligned} \mathcal{C}^{-1}P_{\mathcal{M}_r}(X) &= \mathcal{C}^{-1} \arg \min_{Y \in \mathcal{M}_r} \|X - Y\| = \mathcal{C}^{-1} \arg \min_{Y \in \mathcal{M}_r} \|\mathcal{C}(\mathcal{C}^{-1}X - \mathcal{C}^{-1}Y)\| = \\ &= \arg \min_{Z \in \mathcal{M}_r} \|\mathcal{C}^{-1}X - Z\|_{\mathcal{P}} = P_{\mathcal{M}_r}^{\mathcal{P}}(\mathcal{C}^{-1}X), \end{aligned}$$

where we used the change of variables $Z = \mathcal{C}^{-1}Y$. $\square$

Lemma 5.3. $\mathcal{C}^{-1}\text{Proj}_{\mathcal{C}X}(Y) = \text{Proj}_X^{\mathcal{P}}(\mathcal{C}^{-1}Y)$ for all $X \in \mathcal{M}_r$, $Y \in \mathbb{R}^{m \times n}$.

Proof.

$$\begin{aligned} \mathcal{C}^{-1}\text{Proj}_{\mathcal{C}X}(Y) &= \mathcal{C}^{-1} \arg \min_{\xi \in \mathcal{T}_{\mathcal{C}X}\mathcal{M}_r} \|Y - \xi\| = \arg \min_{\eta \in \mathcal{C}^{-1}\mathcal{T}_{\mathcal{C}X}\mathcal{M}_r} \|Y - \mathcal{C}\eta\|_{\mathcal{P}} \stackrel{\text{Lemma 5.1}}{=} \\ &= \arg \min_{\eta \in \mathcal{T}_X\mathcal{M}_r} \|Y - \mathcal{C}\eta\| = \arg \min_{\eta \in \mathcal{T}_X\mathcal{M}_r} \|\mathcal{C}^{-1}Y - \eta\|_{\mathcal{P}} = \text{Proj}_X^{\mathcal{P}}(\mathcal{C}^{-1}Y). \end{aligned}$$

□

5.1.2.2 QR and SVD decomposition with non-standard inner products

In Section 2.1, where $\mathcal{M}_r$ is endowed with the Frobenius metric, the Riemannian optimization tools heavily rely on QR and SVD decompositions. To adapt them to the case of the $\mathcal{P}$ -induced metric, a generalization of the QR and SVD decomposition is required.

Definition 5.4. Let $T \in \mathbb{R}^{k \times k}$ be positive definite. A matrix $Q \in \mathbb{R}^{k \times k}$ is T -orthogonal if $Q^\top T Q = I_k$. A matrix $Q \in \mathbb{R}^{k \times r}$, $r \leq k$, has T -orthonormal columns if $Q^\top T Q = I_r$.

Proposition 5.5 (T -QR decomposition). Let X be in $\mathbb{R}^{m \times n}$ and let $T \in \mathbb{R}^{m \times m}$ be positive definite. Then there exists a T -orthogonal matrix $\tilde{Q} \in \mathbb{R}^{m \times m}$ and an upper triangular matrix $R \in \mathbb{R}^{m \times n}$ such that $X = \tilde{Q}R$.

Proof. Let $T = C^\top C$ be the Cholesky decomposition of T . Let $CX = QR$ be the QR decomposition of CX . Then $X = \tilde{Q}R$ where R is upper triangular and $\tilde{Q} = C^{-1}Q$ is such that

$$\tilde{Q}^\top T \tilde{Q} = Q^\top C^{-\top} T C^{-1} Q = Q^\top Q = I_m.$$

□

Proposition 5.6 (E, D -Singular Value Decomposition [Van76]). Let X be in $\mathbb{R}^{m \times n}$, $m \geq n$, and let $E \in \mathbb{R}^{m \times m}$ and $D \in \mathbb{R}^{n \times n}$ be positive definite. Then, there exists an E -orthogonal $\tilde{U} \in \mathbb{R}^{m \times m}$, a D -orthogonal $\tilde{V} \in \mathbb{R}^{n \times n}$ and a diagonal $\tilde{\Sigma} = \text{diag}(\tilde{\sigma}_1, \dots, \tilde{\sigma}_p) \in \mathbb{R}^{m \times n}$, where $p = \min\{m, n\}$ and $\tilde{\sigma}_1 \geq \dots \geq \tilde{\sigma}_p \geq 0$, such that $X = \tilde{U} \tilde{\Sigma} \tilde{V}^\top$.

Proof. Let $D = C_D^\top C_D$ and $E = C_E^\top C_E$ be the Cholesky decomposition of D and E . Let $C_E X C_D^\top = U \Sigma V^\top$ be the SVD of $C_E X C_D^\top$. Then $X = \tilde{U} \Sigma \tilde{V}^\top$, where $\tilde{U} = C_E^{-1}U$ and $\tilde{V} = C_D^{-1}V$ are E -orthogonal and D -orthogonal, respectively. □

If $\text{rank}(X) = r < \min\{m, n\}$, the definition and existence of *thin* T -QR and E, D -SVD decompositions is straightforward. It should be noted that, although the above proofs use the Cholesky factorization, the T -QR decomposition can be computed by working with the matrix T alone. This can be achieved, for example, through a modified Gram-Schmidt process with reorthogonalization in the inner product induced by T [LBG13] or by using Householder reflections [Sha23]. Similarly, an algorithm for computing the E, D -SVD without factoring E, D can be devised from the algorithms for the E -QR and D -QR decomposition. For simplicity, however, we employed the Cholesky factorization of the matrices in our implementation.

5.1.2.3 Riemannian optimization tools

Representing points and tangent vectors To identify a point $X \in \mathcal{M}_r$ and store it in memory, it is convenient to store its thin E, D -SVD $(\tilde{U}, \tilde{\Sigma}, \tilde{V})$. Moreover, the following representation of the tangent space holds. Note the similarity with (2.16).

Proposition 5.7. *The tangent space $T_X \mathcal{M}_r$ at $X = \tilde{U} \tilde{\Sigma} \tilde{V}^\top$ is given by*

$$T_X \mathcal{M}_r = \{ \tilde{U} \tilde{M} \tilde{V}^\top + \tilde{U}_p \tilde{V}^\top + \tilde{U} \tilde{V}_p^\top : \tilde{M} \in \mathbb{R}^{r \times r}, \tilde{U}_p \in \mathbb{R}^{m \times r}, \tilde{V}_p \in \mathbb{R}^{n \times r}, \tilde{U}^\top E \tilde{U}_p = 0, \tilde{V}^\top D \tilde{V}_p = 0 \}. \quad (5.10)$$

Proof. According to Lemma 5.1, ξ belongs to $T_X \mathcal{M}_r$ if and only if $\xi = \mathcal{C}^{-1} \eta$ for some η in $T_{\mathcal{C}X} \mathcal{M}$. The SVD decomposition of $\mathcal{C}X$ is given by $\mathcal{C}X = C_E X C_D^\top = U \Sigma V^\top$, where $U = C_E \tilde{U}$ and $V = C_D \tilde{V}$. From (2.16), we can express η as

$$\eta = U M V^\top + U_p V^\top + U V_p^\top,$$

where $M \in \mathbb{R}^{r \times r}$, $U_p \in \mathbb{R}^{m \times r}$, $V_p \in \mathbb{R}^{n \times r}$, and $U^\top U_p = V^\top V_p = 0$. Consequently, $\xi = \mathcal{C}^{-1} \eta = \tilde{U} \tilde{M} \tilde{V}^\top + \tilde{U}_p \tilde{V}^\top + \tilde{U} \tilde{V}_p^\top$, with

$$\tilde{M} = M, \quad \tilde{U}_p = C_E^{-1} U_p, \quad \tilde{V}_p = C_D^{-1} V_p,$$

and it holds $\tilde{U}^\top E \tilde{U}_p = U^\top U_p = 0$, $\tilde{V}^\top D \tilde{V}_p = V^\top V_p = 0$. This shows the inclusion $\subseteq$, the opposite inclusion is obtained analogously. $\square$

Analogously to the Frobenius metric case, to store a tangent vector $\xi \in T_X \mathcal{M}_r$ it is sufficient to store the matrices $(\tilde{M}, \tilde{U}_p, \tilde{V}_p)$. It is worth noting that if $X = \tilde{U} \tilde{\Sigma} \tilde{V}^\top \in \mathcal{M}_r$ and $\xi, \xi' \in T_X \mathcal{M}_r$ are represented as $\xi \doteq (\tilde{M}, \tilde{U}_p, \tilde{V}_p)$ and $\xi' \doteq (\tilde{M}', \tilde{U}'_p, \tilde{V}'_p)$, then

$$\begin{aligned} \langle \xi, \xi' \rangle_{\mathcal{P}} &= \langle (E \tilde{U}) \tilde{M} (D \tilde{V})^\top + (E \tilde{U}_p) (D \tilde{V})^\top + (E \tilde{U}) (D \tilde{V}_p)^\top, \tilde{U} \tilde{M}' \tilde{V}^\top + \tilde{U}'_p \tilde{V}^\top + \tilde{U} \tilde{V}'_p{}^\top \rangle = \\ &= \langle \tilde{M}, \tilde{M}' \rangle + \langle E \tilde{U}_p, \tilde{U}'_p \rangle + \langle D \tilde{V}_p, \tilde{V}'_p \rangle. \end{aligned}$$

Therefore, when they are calculated as side effects of other computations, it is convenient to store also the matrices $E \tilde{U}_p$ and $D \tilde{V}_p$ in memory.

Orthogonal projectors

Proposition 5.8. *Let $X = \tilde{U} \tilde{\Sigma} \tilde{V}^\top \in \mathcal{M}_r$ and let us define*

$$P_{\tilde{U}}^{\mathcal{P}} = \tilde{U} \tilde{U}^\top E, \quad P_{\tilde{V}}^{\mathcal{P}} = \tilde{V} \tilde{V}^\top D, \quad P_{\tilde{U}}^{\perp, \mathcal{P}} = I_m - P_{\tilde{U}}^{\mathcal{P}}, \quad P_{\tilde{V}}^{\perp, \mathcal{P}} = I_n - P_{\tilde{V}}^{\mathcal{P}}.$$

The orthogonal projection of $Z \in \mathbb{R}^{m \times n}$ to $T_X \mathcal{M}_r$ with respect to the inner product induced by $\mathcal{P}X = EXD$ is given by

$$\text{Proj}_X^{\mathcal{P}}(Z) = P_{\tilde{U}}^{\mathcal{P}} Z (P_{\tilde{V}}^{\mathcal{P}})^\top + P_{\tilde{U}}^{\perp, \mathcal{P}} Z (P_{\tilde{V}}^{\mathcal{P}})^\top + P_{\tilde{U}}^{\mathcal{P}} Z (P_{\tilde{V}}^{\perp, \mathcal{P}})^\top, \quad (5.11)$$

which is represented by

$$\begin{aligned}\tilde{M} &= \tilde{U}^\top EZD\tilde{V} \\ \tilde{U}_p &= P_{\tilde{U}}^{\perp, \mathcal{P}} ZD\tilde{V} = ZD\tilde{V} - \tilde{U}\tilde{M} \\ \tilde{V}_p &= P_{\tilde{V}}^{\perp, \mathcal{P}} Z^\top E\tilde{U} = Z^\top E\tilde{U} - \tilde{V}\tilde{M}^\top.\end{aligned}\tag{5.12}$$

Proof. First, let us verify that $\text{Proj}_X^{\mathcal{P}}(Z)$, as defined in (5.11), is indeed represented by the triplet $(\tilde{M}, \tilde{U}_p, \tilde{V}_p)$ of (5.12). It holds

$$\begin{aligned}P_{\tilde{U}}^{\mathcal{P}} Z(P_{\tilde{V}}^{\mathcal{P}})^\top &= \tilde{U}\tilde{U}^\top EZD\tilde{V}\tilde{V}^\top = \tilde{U}\tilde{M}\tilde{V}^\top, \\ P_{\tilde{U}}^{\perp, \mathcal{P}} Z(P_{\tilde{V}}^{\mathcal{P}})^\top &= P_{\tilde{U}}^{\perp, \mathcal{P}} ZD\tilde{V}\tilde{V}^\top = \tilde{U}_p\tilde{V}^\top, \\ P_{\tilde{U}}^{\mathcal{P}} Z(P_{\tilde{V}}^{\perp, \mathcal{P}})^\top &= \tilde{U}\tilde{U}^\top EZ(P_{\tilde{V}}^{\perp, \mathcal{P}})^\top = \tilde{U}\tilde{V}_p^\top.\end{aligned}$$

Moreover,

$$\tilde{U}^\top EP_{\tilde{U}}^{\perp, \mathcal{P}} = 0, \quad \tilde{V}^\top DP_{\tilde{V}}^{\perp, \mathcal{P}} = 0,$$

therefore $\text{Proj}_X^{\mathcal{P}}$, as defined by (5.11), is such that $\text{Proj}_X^{\mathcal{P}}(Z) \in \mathbf{T}_X \mathcal{M}_r$ for all $Z \in \mathbb{R}^{m \times n}$ and $\text{Proj}_X^{\mathcal{P}}(\xi) = \xi$ for all $\xi \in \mathbf{T}_X \mathcal{M}_r$. To conclude, we need to show orthogonality, i.e., that

$$\langle Z - \text{Proj}_X^{\mathcal{P}}(Z), \eta \rangle_{\mathcal{P}} = 0, \quad \forall Z \in \mathbb{R}^{m \times n}, \forall \eta \in \mathbf{T}_X \mathcal{M}_r.$$

If $\eta \in \mathbf{T}_X \mathcal{M}_r$ is represented by the triplet $(\tilde{M}_\eta, \tilde{U}_\eta, \tilde{V}_\eta)$, then

$$\begin{aligned}\langle Z, \eta \rangle_{\mathcal{P}} &= \langle EZD, \tilde{U}\tilde{M}_\eta\tilde{V}^\top + \tilde{U}_\eta\tilde{V}^\top + \tilde{U}\tilde{V}_\eta^\top \rangle = \\ &= \langle \tilde{M}_\eta, \tilde{U}^\top EZD\tilde{V} \rangle + \langle \tilde{U}_\eta, EZD\tilde{V} \rangle + \langle \tilde{V}_\eta, DZ^\top E\tilde{U} \rangle = \\ &= \langle \tilde{M}_\eta, \tilde{M} \rangle + \langle \tilde{U}_\eta, E\tilde{U}_p + E\tilde{U}\tilde{M} \rangle + \langle \tilde{V}_\eta, D\tilde{V}_p + D\tilde{V}\tilde{M}^\top \rangle = \\ &= \langle \tilde{M}_\eta, \tilde{M} \rangle + \langle \tilde{U}_\eta, E\tilde{U}_p \rangle + \langle \tilde{V}_\eta, D\tilde{V}_p \rangle = \langle \text{Proj}_X^{\mathcal{P}}(Z), \eta \rangle_{\mathcal{P}}.\end{aligned}\tag{5.13}$$

□

Retraction As a retraction, we employ the metric projection retraction (with respect to the metric induced by $\mathcal{P}$). To compute it, we can establish an analogue of the Eckart-Young-Mirsky theorem for the E, D -SVD.

Proposition 5.9. *Let X be in $\mathbb{R}^{m \times n}$ and let $X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$ be its E, D -SVD where $\tilde{\Sigma} = \text{diag}(\tilde{\sigma}_1, \dots, \tilde{\sigma}_p)$ with $p = \min\{m, n\}$ and $\tilde{\sigma}_r > \tilde{\sigma}_{r+1}$, then*

$$P_{\mathcal{M}_r}^{\mathcal{P}}(X) := \arg \min_{Y \in \mathcal{M}_r} \|X - Y\|_{\mathcal{P}} = \tilde{U}(:, 1:r)\tilde{\Sigma}(1:r, 1:r)\tilde{V}(:, 1:r)^\top.\tag{5.14}$$

Proof. The SVD decomposition of $\mathcal{C}X$ is $\mathcal{C}X = U\Sigma V$, where $U = C_E\tilde{U}$, $V = C_D\tilde{V}$ and

$\Sigma = \tilde{\Sigma}$. Therefore, by Lemma 5.2

$$\begin{aligned} \mathcal{P}_{\mathcal{M}_r}(X) &= \mathcal{C}^{-1} \mathcal{P}_{\mathcal{M}_r}(\mathcal{C}X) = \mathcal{C}^{-1} \left(U(:, 1:r) \Sigma(1:r, 1:r) V(:, 1:r)^\top \right) = \\ &= \tilde{U}(:, 1:r) \tilde{\Sigma}(1:r, 1:r) \tilde{V}(:, 1:r)^\top. \end{aligned}$$

□

Hence, computing the metric projection retraction $R_X^{\mathcal{P}}(t\xi)$ is tantamount to truncating the E, D -SVD of $X + t\xi$ at rank r . The retraction is well-defined if and only if $\tilde{\sigma}_r(X + t\xi) > \tilde{\sigma}_{r+1}(X + t\xi)$, where $\tilde{\sigma}_i(X + t\xi)$ denotes the i -th singular value in the E, D -SVD of $X + t\xi$. Similar to (2.22), when the metric projection is well-defined, it can be efficiently computed by exploiting the fact that $X + t\xi$ has a rank at most $2r$. If $X = \tilde{U} \tilde{\Sigma} \tilde{V}^\top$ and $\xi \doteq (\tilde{M}, \tilde{U}_p, \tilde{V}_p)$, then $X + t\xi = [\tilde{U} \ \tilde{U}_p] \begin{bmatrix} \tilde{\Sigma} + t\tilde{M} & tI_r \\ tI_r & 0_r \end{bmatrix} [\tilde{V} \ \tilde{V}_p]^\top$. First compute

$$\begin{aligned} \begin{bmatrix} \tilde{U} & \tilde{U}_p \end{bmatrix} &= Q_U R_U \quad \text{thin } E\text{-QR decomposition,} \\ \begin{bmatrix} \tilde{V} & \tilde{V}_p \end{bmatrix} &= Q_V R_V \quad \text{thin } D\text{-QR decomposition,} \end{aligned}$$

(assuming $2r \leq \min\{m, n\}$, otherwise the procedure is slightly changed). Then, using the (standard) SVD of the $2r \times 2r$ matrix $R_U \begin{bmatrix} \tilde{\Sigma} + t\tilde{M} & tI_r \\ tI_r & 0_r \end{bmatrix} R_V = \bar{U} \bar{\Sigma} \bar{V}$, the E, D -SVD decomposition of $X + t\xi$ is, using MATLAB's notation

$$X + t\xi = \left(Q_U \bar{U}(:, 1:r) \right) \bar{\Sigma}(1:r, 1:r) \left(Q_V \bar{V}(:, 1:r) \right)^\top.$$

Finally, when computing $R_X^{\mathcal{P}}(t\xi)$ for multiple values of t and the same $(X, \xi) \in \text{TM}$ the E -QR and D -QR decompositions can be saved after the initial retraction.

Transporter As a transporter we use the orthogonal projection (with respect to the metric induced by $\mathcal{P}$) restricted to the tangent (2.13).

Gradient and Hessian: from Euclidean to Riemannian Let $f : \mathcal{M}_r \rightarrow \mathbb{R}$ be a smooth function and let $\bar{f}$ be a smooth extension of f . The Riemannian gradient of f with respect to the metric induced by $\mathcal{P}$ is

$$\text{grad}_{\mathcal{P}} f(X) = \text{Proj}_X^{\mathcal{P}} \left(\text{grad}_{\mathcal{P}} \bar{f}(X) \right) = \text{Proj}_X^{\mathcal{P}} \left(\mathcal{P}^{-1} \text{grad} \bar{f}(X) \right).$$

Let $X = \tilde{U} \tilde{\Sigma} \tilde{V}^\top$ and denote $Z = \text{grad} \bar{f}(X)$. Then, by (5.12), $\text{grad}_{\mathcal{P}} f(X) \in \text{T}_X \mathcal{M}_r$ is represented by the triplet

$$\begin{aligned} \tilde{M} &= \tilde{U}^\top E \left(\mathcal{P}^{-1} Z \right) D \tilde{V} &= \tilde{U}^\top Z \tilde{V} \\ \tilde{U}_p &= \left(\mathcal{P}^{-1} Z \right) D \tilde{V} - \tilde{U} \tilde{M} &= E^{-1} \left(Z \tilde{V} - E \tilde{U} \tilde{M} \right) \\ \tilde{V}_p &= \left(\mathcal{P}^{-1} Z \right)^\top E \tilde{U} - \tilde{V} \tilde{M}^\top &= D^{-1} \left(Z^\top \tilde{U} - D \tilde{V} \tilde{M}^\top \right). \end{aligned} \tag{5.15}$$

Observe that the matrices $E\tilde{U}_p$ and $D\tilde{V}_p$, necessary for computing inner products, are computed as intermediate steps during the calculation of $\tilde{U}_p$ and $\tilde{V}_p$. In comparison to the Riemannian gradient with respect to the Frobenius metric, computing the Riemannian gradient necessitates r additional linear systems with the matrices E and D .

For the Riemannian Hessian, the following proposition holds. Note the similarities with Proposition 2.28. Once again, the computational cost, in comparison to the Frobenius metric, increases by r system resolutions with E and D .

Proposition 5.10. *Let $f : \mathcal{M}_r \rightarrow \mathbb{R}$ be a smooth function, with smooth extension $\bar{f}$. With the same notation as in this section, consider $X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top \in \mathcal{M}_r$, $\xi \in \mathbb{T}_X\mathcal{M}_r$ represented by the triplet $(\tilde{M}_\xi, \tilde{U}_\xi, \tilde{V}_\xi)$, and let us denote $Z = \text{grad}\bar{f}(X)$, $\dot{Z} = \text{Hess}\bar{f}(X)[\xi]$. Then the Riemannian Hessian of f on $\mathcal{M}_r$ endowed with the metric induced by $\mathcal{P}X = EXD$ is represented by the triplet*

$$\begin{aligned}\tilde{M} &= \tilde{U}^\top \dot{Z} \tilde{V}, \\ \tilde{U}_p &= E^{-1}(I - E\tilde{U}\tilde{U}^\top) \left[\dot{Z}\tilde{V} + Z\tilde{V}_\xi\tilde{\Sigma}^{-1} \right], \\ \tilde{V}_p &= D^{-1}(I - D\tilde{V}\tilde{V}^\top) \left[\dot{Z}^\top\tilde{U} + Z^\top\tilde{U}_\xi\tilde{\Sigma}^{-1} \right].\end{aligned}\tag{5.16}$$

Proof. Let us denote by $J(X)$ and $J^\mathcal{P}(X)$ the Fréchet differential at X of $X \mapsto \text{Proj}_X$ and $X \mapsto \text{Proj}_X^\mathcal{P}$, respectively. By Proposition 2.7, it holds

$$\text{Hess}_\mathcal{P}f(X)[\xi] = \text{Proj}_X^\mathcal{P} \left(\text{Hess}_\mathcal{P}\bar{f}(X)[\xi] \right) + \text{Proj}_X^\mathcal{P} \left(J^\mathcal{P}(X)[\xi](\text{grad}_\mathcal{P}\bar{f}(X)) \right).$$

Denoting $\dot{Z} = \text{Hess}\bar{f}(X)[\xi]$, by Proposition 5.8 the projected Euclidean Hessian

$$\text{Proj}_X^\mathcal{P} \left(\text{Hess}_\mathcal{P}\bar{f}(X)[\xi] \right) = \text{Proj}_X^\mathcal{P} \left(\mathcal{P}^{-1} \text{Hess}\bar{f}(X) \right)$$

is represented by the triplet

$$\begin{aligned}\tilde{M} &= \tilde{U}^\top E(\mathcal{P}^{-1}\dot{Z})D\tilde{V} = \tilde{U}^\top \dot{Z}D, \\ \tilde{U}_p &= (I - \tilde{U}\tilde{U}^\top E)(\mathcal{P}^{-1}\dot{Z})D\tilde{V} = E^{-1}(I - E\tilde{U}\tilde{U}^\top)\dot{Z}\tilde{V}, \\ \tilde{V}_p &= (I - \tilde{V}\tilde{V}^\top D)(\mathcal{P}^{-1}\dot{Z})^\top E\tilde{U} = D^{-1}(I - D\tilde{V}\tilde{V}^\top)\dot{Z}^\top\tilde{U}.\end{aligned}\tag{5.17}$$

To derive an expression for $J^\mathcal{P}(X)[\xi]$, observe that by Lemma 5.3 it holds $\text{Proj}_X^\mathcal{P} = \mathcal{C}^{-1} \circ \text{Proj}_{\mathcal{C}X} \circ \mathcal{C}$, and differentiating with respect to X yields

$$J^\mathcal{P}(X)[\xi] = \mathcal{C}^{-1} \circ J(\mathcal{C}X)[\mathcal{C}\xi] \circ \mathcal{C}.$$

Hence, applying Lemma 5.3 once again,

$$\text{Proj}_X^\mathcal{P} \left(J^\mathcal{P}(X)[\xi] \left(\text{grad}_\mathcal{P}\bar{f}(X) \right) \right) = \mathcal{C}^{-1} \text{Proj}_{\mathcal{C}X} \left(J(\mathcal{C}X)[\mathcal{C}\xi] \left(\mathcal{C}^{-\top} \text{grad}\bar{f}(X) \right) \right).$$

Suppose that $X = \tilde{U}\tilde{\Sigma}\tilde{V}^\top$ and $\xi \doteq (\tilde{M}_\xi, \tilde{U}_\xi, \tilde{V}_\xi)$, then $\mathcal{C}X = U\Sigma V^\top$ where $\Sigma = \tilde{\Sigma}$, $U = C_E\tilde{U}$, $V = C_D\tilde{V}$, and $\mathcal{C}\xi \doteq (M_\xi, U_\xi, V_\xi)$ where $M_\xi = \tilde{M}_\xi$, $U_\xi = C_E\tilde{U}_\xi$, $V_\xi =$

$C_D \tilde{V}_\xi$. Similarly, if $\eta := \text{Proj}_X^\mathcal{P} \left(J^\mathcal{P}(X)[\xi] \left(\text{grad}_\mathcal{P} \bar{f}(X) \right) \right)$ is represented by the triplet $(\tilde{M}_\eta, \tilde{U}_\eta, \tilde{V}_\eta)$, then

$$\text{Proj}_{\mathcal{C}X} \left(J(\mathcal{C}X)[\mathcal{C}\xi] \left(\mathcal{C}^{-\top} \text{grad} \bar{f}(X) \right) \right) = \mathcal{C}\eta \doteq (M_\eta, U_\eta, V_\eta)$$

with $M_\eta = \tilde{M}_\eta$, $U_\eta = C_E \tilde{U}_\eta$ and $V_\eta = C_D \tilde{V}_\eta$. By Proposition 2.28 it holds

$$\begin{aligned} M_\eta &= 0, \\ U_\eta &= P_U^\perp \left(\left(\mathcal{C}^{-\top} \text{grad} \bar{f}(X) \right) V_\xi \Sigma^{-1} \right), \\ V_\eta &= P_V^\perp \left(\left(\mathcal{C}^{-\top} \text{grad} \bar{f}(X) \right)^\top U_\xi \Sigma^{-1} \right). \end{aligned}$$

Reverting the change of variables yields

$$\begin{aligned} \tilde{U}_\eta &= C_E^{-1} P_U^\perp \left(\left(\mathcal{C}^{-\top} \text{grad} \bar{f}(X) \right) V_\xi \Sigma^{-1} \right) = C_E^{-1} P_U^\perp \left(C_E^{-\top} Z C_D^{-1} V_\xi \Sigma^{-1} \right) \\ &= (C_E^{-1} P_U^\perp C_E^{-\top}) Z \tilde{V}_\xi \Sigma^{-1} = (E^{-1} - \tilde{U} \tilde{U}^\top) Z \tilde{V}_\xi \Sigma^{-1} \\ &= E^{-1} (I - E \tilde{U} \tilde{U}^\top) Z \tilde{V}_\xi \Sigma^{-1}, \end{aligned}$$

and similarly

$$\tilde{V}_\eta = D^{-1} (I - D \tilde{V} \tilde{V}^\top) Z^\top \tilde{U}_\xi \Sigma^{-1}.$$

This concludes the proof. $\square$

5.1.2.4 Line-search initialization and rank adaptivity

When employing the tools for $\mathcal{M}_r$ as an embedded submanifold of $(\mathbb{R}^{m \times n}, \langle \cdot, \cdot \rangle_\mathcal{P})$, minor adjustments are required in the calculation of the initial step-size in Armijo backtracking and in the rank-update step of the RRAM.

The initial step-size $\bar{\alpha}_k$ for the Armijo backtracking procedure remains the same as in (4.9), as it is obtained neglecting the retraction. The only modification lies in how it is computed: substitute the Riemannian gradient and the projected Euclidean Hessian with those in the metric induced by $\mathcal{P}$, and employ $\langle \cdot, \cdot \rangle_\mathcal{P}$ instead of $\langle \cdot, \cdot \rangle$.

To adjust the rank-increase step in RRAMs, we need to employ the orthogonal projection with respect to the metric induced by $\mathcal{P}$ of the Euclidean gradient with respect to the metric induced by $\mathcal{P}$. This requires truncating at rank r_{up} the E, D -SVD of $\text{grad}_\mathcal{P} f(X) = \mathcal{P}^{-1} \text{grad} f(X) = E^{-1} (AX - F) D^{-1}$. Using the same notation as in (4.5), this entails the computation of $E^{-1} Y$ and $D^{-1} W$, as well as the use of the E -QR and D -QR decompositions in place of the standard QR-decomposition.

5.1.3 Difference between the two approaches

Although the metrics induced by $\mathcal{P}_X$ and $\mathcal{P}$ coincide, it is not the same to run Riemannian optimization algorithms in the following two ways:

5.2. $\mathcal{P}X = AXD + EXB$ where $A, B, D, E \succ 0$

- (i) employing the tools for $\mathcal{M}_r$ as an embedded submanifold of $(\mathbb{R}^{m \times n}, \langle \cdot, \cdot \rangle)$ and leveraging the action of $\mathcal{P}_X^{-1}$, as detailed in Sections 3.4.4 and 3.4.5; or
- (ii) using the tools for $\mathcal{M}_r$ as an embedded submanifold of $(\mathbb{R}^{m \times n}, \langle \cdot, \cdot \rangle_{\mathcal{P}})$, developed in Section 5.1.2.

A key distinction between these approaches lies in the retraction used. In (i), the metric projection retraction with respect to the Frobenius metric is employed, while (ii) uses metric projection retraction with respect to the metric induced by $\mathcal{P}$. This distinction affects Riemannian GD, altering the curve along which line search is executed, despite the search direction remaining the same (i.e., the negative preconditioned Riemannian gradient).

Additionally, the transporters differ due to the different metric used for orthogonal projection onto the tangent. This marks a second difference between the two approaches in Riemannian NLCG, impacting both the transportation of the previous descent direction to the current tangent space and the computation of β_k .

For Riemannian trust-region, the first difference is in how the candidate next step is obtained by retracting the vector obtained from approximately solving the TRs (line 4 of Algorithm 6). Additionally, since the retraction changes, the pullback of the objective function to the tangent space also changes. If approach (i) is utilized with the model Hessian derived as in (3.26) from the Riemannian Hessian in the Frobenius metric, the resulting model m_X is the second-order Taylor expansion of $\hat{f}_X = f \circ R_X$. Conversely, if approach (ii) is employed with the model Hessian being the Riemannian Hessian in the metric induced by $\mathcal{P}$, the resulting model is the second-order Taylor expansion of $\hat{f}_X^{\mathcal{P}} = f \circ \text{Retr}_X^{\mathcal{P}}$. In general, these two models differ. On the other hand, if the projected Euclidean Hessian is employed as the model Hessian in both approaches, there is no difference in m_X , as discussed in Section 3.4.6.

5.2 $\mathcal{P}X = AXD + EXB$ where $A, B, D, E \succ 0$

We now investigate preconditioners $\mathcal{P}$ that exhibit a Sylvester-like structure. As a preliminary step, we consider the simpler case of $\mathcal{P}X = AX + XB$ with $A, B \succ 0$. By combining the latter with the tools outlined in Section 5.1.2, we will subsequently address the more general scenario of $\mathcal{P}X = AXD + EXB$ with $A, B, D, E \succ 0$.

5.2.1 The case $\mathcal{P}X = AX + XB$

Let us consider the Sylvester operator $\mathcal{P}X = AX + XB$ where $A, B \succ 0$. The only viable method for conducting Riemannian optimization with such a preconditioner is through the action of $\mathcal{P}_X^{-1}$. Applying $\mathcal{P}_X^{-1}$ to a tangent vector $\eta \in T_X \mathcal{M}$ involves solving the projected Sylvester equation

$$\text{Proj}_X(A\xi + \xi B) = \eta \quad \xi \in T_X \mathcal{M}_r. \quad (5.18)$$

The solution of (5.18) has been addressed in [VV10, § 7.2] for Lyapunov operators and positive semidefinite fixed-rank matrices. Additionally, the more general scenario of tensors in the Tucker format and Laplacian-like preconditioners was considered in [KSV16, § 4.2]. While the tensor case in [KSV16, § 4.2] encompasses (5.18), it is computationally more convenient to extend the techniques outlined in [VV10, § 7.2] — specifically designed for the matrix case — to Sylvester operators and general fixed-rank matrices, as explained below.

Let $\eta \doteq (M_\eta, U_\eta, V_\eta)$ and $\xi \doteq (M_\xi, U_\xi, V_\xi)$. According to (2.19), the linear system (5.18) is equivalent to

$$U_\eta = P_U^\perp (A\xi + \xi B)V, \quad V_\eta = P_V^\perp (A\xi + \xi B)^\top U, \quad M_\eta = U^\top (A\xi + \xi B)V,$$

which can be simplified and rearranged as

$$\begin{aligned} (P_U^\perp A)U_\xi + U_\xi(V^\top BV) &= U_\eta - (P_U^\perp AU)M_\xi, \\ (P_V^\perp B)V_\xi + V_\xi(U^\top AU) &= V_\eta - (P_V^\perp BV)M_\xi^\top, \\ (U^\top AU)M_\xi + M_\xi(V^\top BV) &= M_\eta - (U^\top A)U_\xi - ((V^\top B)V_\xi)^\top. \end{aligned} \tag{5.19}$$

Taking the eigenvalue decompositions

$$V^\top BV = Q_U \Lambda Q_U^\top, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_r), \quad U^\top AU = Q_V K Q_V^\top, \quad K = \text{diag}(k_1, \dots, k_r)$$

and with the change of coordinates

$$\begin{aligned} \bar{U}_\xi &= U_\xi Q_U, & \bar{U}_\eta &= U_\eta Q_U, & \bar{U} &= U Q_V, \\ \bar{V}_\xi &= V_\xi Q_V, & \bar{V}_\eta &= V_\eta Q_V, & \bar{V} &= V Q_U, \\ \bar{M}_\xi &= Q_V^\top M_\xi Q_U, & \bar{M}_\eta &= Q_V^\top M_\eta Q_U, \end{aligned}$$

the system (5.19) is equivalent to

$$(P_U^\perp A)\bar{U}_\xi + \bar{U}_\xi \Lambda = \bar{U}_\eta - (A\bar{U} - \bar{U}K)\bar{M}_\xi, \tag{5.20a}$$

$$(P_V^\perp B)\bar{V}_\xi + \bar{V}_\xi K = \bar{V}_\eta - (B\bar{V} - \bar{V}\Lambda)\bar{M}_\xi^\top, \tag{5.20b}$$

$$K\bar{M}_\xi + \bar{M}_\xi \Lambda = \bar{M}_\eta - (\bar{U}^\top A)\bar{U}_\xi - ((\bar{V}^\top B)\bar{V}_\xi)^\top, \tag{5.20c}$$

together with the orthogonality constraints $\bar{V}_\xi^\top \bar{V} = \bar{U}_\xi^\top \bar{U} = 0_r$. Reading equation (5.20a) on the i -th column yields

$$(P_U^\perp A)\bar{U}_\xi(:, i) + \lambda_i \bar{U}_\xi(:, i) = \bar{U}_\eta(:, i) - (A\bar{U} - \bar{U}K)\bar{M}_\xi(:, i) \quad \text{s.t.} \quad \bar{U}^\top \bar{U}_\xi(:, i) = 0,$$

which is equivalent to the saddle point system

$$\begin{bmatrix} A + \lambda_i I & \bar{U} \\ \bar{U}^\top & 0 \end{bmatrix} \begin{bmatrix} \bar{U}_\xi(:, i) \\ y_u \end{bmatrix} = \begin{bmatrix} \bar{U}_\eta(:, i) - (A\bar{U} - \bar{U}K)\bar{M}_\xi(:, i) \\ 0 \end{bmatrix} \tag{5.21}$$

with $y_u \in \mathbb{R}^r$ the dual variable. Since U and $A + \lambda_i I$ are full-rank, the above system is nonsingular. Given that r is expected to be relatively low, an explicit solution can be obtained by inverting the Schur complement. It holds

$$\bar{U}_\xi(:, i) = w_u^{(i)} - F_u^{(i)} \bar{M}_\xi(:, i) \quad (5.22)$$

where

$$\begin{aligned} S_u^{(i)} &= -\bar{U}^\top (A + \lambda_i I_m)^{-1} \bar{U} \in \mathbb{R}^{r \times r}, \\ w_u^{(i)} &= \left(I_m + (A + \lambda_i I_m)^{-1} \bar{U} (S_u^{(i)})^{-1} \bar{U}^\top \right) (A + \lambda_i I_m)^{-1} \bar{U}_\eta(:, i), \\ F_u^{(i)} &= \left(I_m + (A + \lambda_i I_m)^{-1} \bar{U} (S_u^{(i)})^{-1} \bar{U}^\top \right) (A + \lambda_i I_m)^{-1} (A \bar{U} - \bar{U} K). \end{aligned} \quad (5.23)$$

Similarly, from (5.20b) one can write

$$\bar{V}_\xi(:, i) = w_v^{(i)} - F_v^{(i)} \bar{M}_\xi(i, :)^T \quad (5.24)$$

where expressions for $w_v^{(i)}, F_v^{(i)}$ are analogues of the ones for $w_u^{(i)}, F_u^{(i)}$. Let us denote

$$\begin{aligned} W_u &= \begin{bmatrix} w_u^{(1)} & \dots & w_u^{(r)} \end{bmatrix}, & \Lambda^{(i)} &= \lambda_i I_r - (\bar{U}^\top A) F_u^{(i)}, \\ W_v &= \begin{bmatrix} w_v^{(1)} & \dots & w_v^{(r)} \end{bmatrix}, & K^{(i)} &= k_i I_r - (\bar{V}^\top B) F_v^{(i)}. \end{aligned}$$

Substituting the previously obtained expressions in (5.20c) yields

$$\left[\Lambda^{(1)} M_\xi(:, 1) \mid \dots \mid \Lambda^{(r)} M_\xi(:, r) \right] + \begin{bmatrix} M_\xi(1, :)(K^{(1)})^\top \\ \vdots \\ M_\xi(r, :)(K^{(r)})^\top \end{bmatrix} = R, \quad (5.25)$$

where

$$R = \bar{M}_\eta - (\bar{U}^\top A) W_u - ((\bar{V}^\top B) W_v)^\top. \quad (5.26)$$

Let $\Pi_{r,r} \in \mathbb{R}^{r^2 \times r^2}$ be the perfect shuffle matrix. Vectorizing (5.25) we obtain the $r^2 \times r^2$ system

$$F \text{vec}(\bar{M}_\xi) = \text{vec}(R), \quad F = \left(\begin{bmatrix} \Lambda^{(1)} & & \\ & \ddots & \\ & & \Lambda^{(r)} \end{bmatrix} + \Pi_{r,r} \begin{bmatrix} K^{(1)} & & \\ & \ddots & \\ & & K^{(r)} \end{bmatrix} \Pi_{r,r} \right). \quad (5.27)$$

Once we solve for $\bar{M}_\xi$, we can obtain $\bar{U}_\xi, \bar{V}_\xi$ using (5.22) and (5.24) respectively, and finally, revert the change of coordinates to obtain $U_\xi = \bar{U} Q_U^\top$, $V_\xi = \bar{V} Q_V^\top$ and $M_\xi = Q_V \bar{M}_\xi Q_U^\top$.

Implementation aspects and complexity The two dominant costs are the assembly of $w_u^{(i)}, w_v^{(i)}, F_u^{(i)}, F_v^{(i)}$ for $i = 1, \dots, r$, as described in (5.23), and the solution of the

$r^2 \times r^2$ system (5.27).

Concerning (5.23), for each i , we need to apply $(A + \lambda_i I_m)^{-1}$ to $\bar{U}$, $(A\bar{U} - \bar{U}K)$ and $\bar{U}_\eta(:, i)$. If we denote by $c_A(m)$ the complexity of applying $(A + \lambda_i I_m)^{-1}$ to a vector in $\mathbb{R}^m$, the cost is $\mathcal{O}(c_A(m)r)$. Then, computing the Cholesky factorization of $S_u^{(i)}$ and solving by backward and forward substitutions $r+1$ systems with matrix $S_u^{(i)}$ gives a cost $\mathcal{O}(r^3)$. In total, the cost of assembling the system (5.27) is $\mathcal{O}(c_A(m)r^2 + c_B(n)r^2) + \mathcal{O}(r^4)$ (where $c_B(n)$ denotes the complexity of applying $(B + k_i I_n)^{-1}$ to a vector in $\mathbb{R}^n$). Usually $r \ll m, n$, and the dominating term is $\mathcal{O}(c_A(m)r^2 + c_B(n)r^2)$. Assuming optimal complexity $c_A(m) = \mathcal{O}(m)$ and $c_B(n) = \mathcal{O}(n)$, the cost is $\mathcal{O}(mr^2 + nr^2) + \mathcal{O}(r^4)$ operations, which is the same asymptotic complexity as computing a retraction or a transporter if $r = \mathcal{O}(\sqrt{m+n})$. Moreover, the construction of $w_u^{(i)}, w_v^{(i)}, F_u^{(i)}, F_v^{(i)}$ for different i 's is embarrassingly parallel.

Solving the $r^2 \times r^2$ system (5.27) with a direct method results in a $\mathcal{O}(r^6)$ cost in general, which is feasible only for relatively small ranks. For larger ranks, it is advisable to use an iterative solver like CG, with the structure of F that can be exploited to apply it with cost $\mathcal{O}(r^3)$.

Finally, if we wish to solve (5.18) for a fixed X and multiple right-hand-sides $\eta \in T_X \mathcal{M}_r$, as it happens when preconditioning tCG, after the first iteration the subsequent system assembly can be carried out with a lower cost. Indeed, the system matrix F remains the same, and we only need to compute the right-hand side R in (5.26). If we store the matrices $(A + \lambda_i I_m)^{-1} \bar{U}$, $(A + \lambda_i I_m)^{-1} (A\bar{U} - \bar{U}K)$, $(B + k_i I_n)^{-1} \bar{V}$, and $(B + k_i I_n)^{-1} (B\bar{V} - \bar{V}\Lambda)$, for all $i = 1, \dots, r$, given that the Cholesky factorizations of $S_u^{(i)}$ and $S_v^{(i)}$ have already been computed, computing W_u and W_v costs $\mathcal{O}(c_A(m)r + c_B(n)r + r^3)$ flops. The matrix products in (5.26) contribute an extra cost of $\mathcal{O}(nr^2 + mr^2)$. Consequently, the overall complexity of subsequent system assembly is $\mathcal{O}(c_A(m)r + c_B(n)r) + \mathcal{O}(nr^2 + mr^2)$. Despite maintaining an asymptotic complexity of $\mathcal{O}(nr^2 + mr^2)$ if $c_A(m) = \mathcal{O}(m)$, $c_B(n) = \mathcal{O}(n)$ and $r = \mathcal{O}(\sqrt{m+n})$, the constants hidden by big- $\mathcal{O}$ notation are substantially lower, and the system assembly is considerably faster in practice. The downside is that memory requirements increase from $\mathcal{O}(r(n+m))$ to $\mathcal{O}(r^2(n+m))$.

5.2.2 The case $\mathcal{P}X = AXD + EXB$

Let us consider a preconditioner $\mathcal{P}X = AXD + EXB$, where $A, B, D, E \succ 0$. When attempting to solve the linear system

$$\text{Proj}_X(A\xi D + E\xi B) = \eta \quad \xi \in T_X \mathcal{M}_r, \quad (5.28)$$

for a given $\eta \in T_X \mathcal{M}_r$, using an approach similar to the one in the previous section appears impractical. Indeed, due to the presence of matrices on both sides of ξ , reading the equations on the components (M_η, U_η, V_η) does not yield decoupled equations for U_ξ and V_ξ . We propose to resort to the Sylvester case in Section 5.2.1 by a change of metric.

Recall that in preconditioning using $\mathcal{P}$, the goal is to employ Riemannian optimization algorithms in the metric induced by $\mathcal{P}$. We can write

$$\begin{aligned} \mathcal{P} &= \mathcal{B}\tilde{\mathcal{P}} \quad \text{where} \quad \mathcal{B}X = EXD, \\ \tilde{\mathcal{P}}X &= \mathcal{B}^{-1}\mathcal{P}X = E^{-1}AX + XBD^{-1}. \end{aligned} \quad (5.29)$$

The linear operator $\tilde{\mathcal{P}}$ is a symmetric positive definite linear operator with respect to the metric induced by $\mathcal{B}$. Moreover, for all $X, Y \in \mathbb{R}^{m \times n}$ it holds

$$\langle X, Y \rangle_{\mathcal{P}} = \langle X, \mathcal{P}Y \rangle = \langle X, \mathcal{B}\tilde{\mathcal{P}}Y \rangle = \langle X, \tilde{\mathcal{P}}Y \rangle_{\mathcal{B}},$$

i.e., the metric induced by $\mathcal{P}$ (using as a base metric the Frobenius metric) coincides with the metric induced by $\tilde{\mathcal{P}}$ using as a base metric the metric induced by $\mathcal{B}$. Therefore, to run Riemannian optimization in the metric induced by $\mathcal{P}$, we can use the tools developed in Section 5.1.2 for the manifold of fixed-rank matrices with the metric induced by $\mathcal{B}$ and run algorithms through the action of $\tilde{\mathcal{P}}_X^{-1}$. Applying $\tilde{\mathcal{P}}_X^{-1}$ to a vector $\eta \in T_X\mathcal{M}_r$ requires solving

$$\text{Proj}_X^{\mathcal{B}}(\tilde{\mathcal{P}}\xi) = \text{Proj}_X^{\mathcal{B}}(E^{-1}A\xi + \xi BD^{-1}) = \eta \quad \xi \in T_X\mathcal{M}_r. \quad (5.30)$$

The advantage, compared to (5.28), is that now $\mathcal{B}^{-1}\mathcal{P}X = E^{-1}AX + XBD^{-1}$ is a Sylvester operator, enabling us to employ techniques akin to those in the previous section. However, we want to avoid explicitly assembling the matrices $E^{-1}A$ and BD^{-1} .

Using the notation from Section 5.1.2, if $\eta \doteq (\tilde{M}_\eta, \tilde{U}_\eta, \tilde{V}_\eta)$ and $\xi \doteq (\tilde{M}_\xi, \tilde{U}_\xi, \tilde{V}_\xi)$, by (5.12) and doing appropriate simplifications, the linear system (5.30) can be rearranged as

$$\begin{aligned} (I_m - E\tilde{U}\tilde{U}^\top)A\tilde{U}_\xi + E\tilde{U}_\xi(\tilde{V}^\top B\tilde{V}) &= E\tilde{U}_\eta - (A\tilde{U} - E\tilde{U}(\tilde{U}^\top A\tilde{U}))\tilde{M}_\xi, \\ (I_m - D\tilde{V}\tilde{V}^\top)B\tilde{V}_\xi + D\tilde{V}_\xi(\tilde{U}^\top A\tilde{U}) &= D\tilde{V}_\eta - (B\tilde{V} - D\tilde{V}(\tilde{V}^\top B\tilde{V}))\tilde{M}_\xi^\top, \\ (\tilde{U}^\top A\tilde{U})\tilde{M}_\xi + \tilde{M}_\xi(\tilde{V}^\top B\tilde{V}) &= \tilde{M}_\eta - (\tilde{U}^\top A)\tilde{U}_\xi - ((\tilde{V}^\top B)\tilde{V}_\xi)^\top, \end{aligned} \quad (5.31)$$

along with orthogonality constraints $\tilde{U}_\xi E\tilde{U} = \tilde{V}_\xi D\tilde{V} = 0_r$. At this point, one can follow a procedure analogous to that in Section 5.2.1. Take the eigenvalue decompositions

$$\tilde{V}^\top B\tilde{V} = Q_U \Lambda Q_U^\top, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_r), \quad \tilde{U}^\top A\tilde{U} = Q_V K Q_V^\top, \quad K = \text{diag}(k_1, \dots, k_r)$$

and rewrite (5.31) using the change of coordinates

$$\begin{aligned} \bar{U}_\xi &= \tilde{U}_\xi Q_U, & \bar{U}_\eta &= \tilde{U}_\eta Q_U, & \bar{U} &= \tilde{U} Q_U, \\ \bar{V}_\xi &= \tilde{V}_\xi Q_V, & \bar{V}_\eta &= \tilde{V}_\eta Q_V, & \bar{V} &= \tilde{V} Q_V, \\ \bar{M}_\xi &= Q_V^\top \tilde{M}_\xi Q_U, & \bar{M}_\eta &= Q_V^\top \tilde{M}_\eta Q_U. \end{aligned}$$

Reading the resulting system for $\bar{U}_\xi$ on the i -th column, together with the orthogonality

5.3. Alternating-Direction Implicit (ADI) iteration

constrains $\bar{U}_\xi^\top E \bar{U} = 0_r$, yields the saddle point system

$$\begin{bmatrix} A + \lambda_i E & E \bar{U} \\ (E \bar{U})^\top & 0 \end{bmatrix} \begin{bmatrix} \bar{U}_\xi(:, i) \\ y_u \end{bmatrix} = \begin{bmatrix} E \bar{U}_\eta(:, i) - (A \bar{U} - E \bar{U} K) \bar{M}_\xi(:, i) \\ 0 \end{bmatrix}. \quad (5.32)$$

Using the Schur complement one obtains

$$\bar{U}_\xi(:, i) = w_u^{(i)} - F_u^{(i)} \bar{M}_\xi(:, i)$$

where the resulting $w_u^{(i)}$ and $F_u^{(i)}$ are now given by

$$\begin{aligned} S_u^{(i)} &= -(E \bar{U})^\top (A + \lambda_i E)^{-1} (E \bar{U}) \in \mathbb{R}^{r \times r}, \\ w_u^{(i)} &= \left(I_m + (A + \lambda_i E)^{-1} (E \bar{U}) (S_u^{(i)})^{-1} (E \bar{U})^\top \right) (A + \lambda_i E)^{-1} \bar{U}_\eta(:, i), \\ F_u^{(i)} &= \left(I_m + (A + \lambda_i E)^{-1} (E \bar{U}) (S_u^{(i)})^{-1} (E \bar{U})^\top \right) (A + \lambda_i E)^{-1} (A \bar{U} - E \bar{U} K). \end{aligned} \quad (5.33)$$

Similar expressions can be derived for $\bar{V}_\xi$, $w_v^{(i)}$ and $F_v^{(i)}$. The rest of the assembly of the $r^2 \times r^2$ system for $\text{vec}(\tilde{M}_\xi)$ is entirely identical to Section 5.2.1. It is important to note that there is no need to explicitly compute the matrices $E^{-1}A$ and $D^{-1}B$; instead, matrices of the pencils $A + \lambda E$ and $B + kD$ are used.

Implementation aspects and complexity The analysis of computational costs mirrors that in Section 5.2.1. Specifically, it is sufficient to substitute the cost $c_A(m)$, related to the application of $(A + \lambda_i I_m)^{-1}$, with the cost $c_{(A,E)}(m)$ for the application of $(A + \lambda_i E)^{-1}$; similarly the cost $c_{(B,D)}(n)$ for the application of $(B + k_i D)^{-1}$ replaces $c_B(n)$. In many applications, optimal complexities $c_{(A,E)}(m) = \mathcal{O}(m)$ and $c_{(B,D)}(n) = \mathcal{O}(n)$ are achieved, such as when matrices A, B, E, D are banded, or sparse with a suitable sparsity pattern, and the total cost for system assembly remains $\mathcal{O}(mr^2 + nr^2) + \mathcal{O}(r^4)$ operations.

The considerations from Section 5.2.1 regarding the computational cost of solving the final $r^2 \times r^2$ system remain applicable. Furthermore, akin to the previous section, the computational cost of subsequent applications of the preconditioner can be reduced when solving (5.30) for a fixed X and multiple right-hand-sides $\eta \in \text{TXM}_r$.

5.3 Alternating-Direction Implicit (ADI) iteration

The effectiveness of preconditioned methods crucially depends on the efficient application of the preconditioner's inverse. This principle holds true both in the linear algebra interpretation of the term preconditioning and within the domain of Riemannian optimization. Typically, as a preconditioner accelerates convergence in terms of iterations, there is a corresponding increase in the computational expense when applying its inverse. Consequently, when a symmetric positive definite linear operator is identified as a suitable preconditioner, a pragmatic and often more efficient approach involves opting for an approximate inverse. This strategy aims to strike the delicate balance between

5.3. Alternating-Direction Implicit (ADI) iteration

computational efficiency and the benefits of preconditioning.

If the preconditioner takes the form of a Sylvester operator $\mathcal{P}X = AXD + EXB$ with $A, B, D, E \succ 0$, applying $\mathcal{P}^{-1}$ entails solving a Sylvester equation. An effective strategy for approximating the inverse is the Alternating Direction Implicit (ADI) iteration [BLT09; TW18; Wac08; Wac13]. In this section, we first introduce the ADI iteration, which will prove valuable in preconditioning the resolution of multiterm matrix equations through Conjugate Gradient with Truncation (Algorithm 1) and the Riemannian truncated preconditioned Richardson iteration (Section 3.4.2). Following this, we then formulate an ADI-like iteration for solving the projected Sylvester equation (5.28). This iteration aims to approximate the inverse of the corresponding Riemannian preconditioner $\mathcal{P}_X = \text{Proj}_X \circ \mathcal{P} \circ \text{Proj}_X$, serving as an (approximate) preconditioner for Riemannian methods.

5.3.1 ADI iteration for Sylvester equations

The Alternating Direction Implicit (ADI) method was first introduced in [PR55] for the solution of elliptic and parabolic differential equation. In the linear algebra context, ADI is an iterative method that addresses the solution of linear systems where the system matrix A is SPD and can be written as a sum of two SPD commuting matrices H and V [Wac13]. In such a case, often referred as the *ADI model problem*, rigorous theoretical convergence analysis is available and rapid convergence of ADI can be guaranteed [Wac13]. However, ADI is often employed even when H and V do not commute, although convergence analysis becomes less precise [Var00].

It was later observed in [EW86; Wac88] that a Sylvester operator $\mathcal{P}X = AX + XB$ can be regarded as the sum of two commuting linear operators: left-multiplication by A and right-multiplication by B . This observation makes it suitable for the application of ADI. Note that if $A, B \succ 0$, symmetry and positive definiteness are guaranteed. However, the SPD requirement can be relaxed to only requiring that A and B are N -stable, i.e., their eigenvalues have positive real part [Wac88; Wac08]. The ADI iteration for the solution of $AX + XB = F$ is given by:

$$\begin{aligned} (A - q_j I)X^{(j-\frac{1}{2})} &= F - X^{(j-1)}(B + q_j I), \\ X^{(j)}(B + p_j I) &= F - (A - p_j I)X^{(j-\frac{1}{2})}, \end{aligned} \tag{5.34}$$

where we made a slight abuse of notation indicating with I the identity matrix of size n or m . An initial guess X_0 is required, typically set to $X_0 = 0$, and the tuples $\{(p_j, q_j)\}_{j=1}^k$ are the *shift parameters*. The choice of shift parameters significantly influences the convergence of ADI.

5.3. Alternating-Direction Implicit (ADI) iteration

5.3.1.1 ADI as a fixed-point iteration

Observing that $(A - p_j I)$, $(A - q_j I)$, $(A - p_j I)^{-1}$ and $(A - q_j I)^{-1}$ commute, and the same holds for shifts of B , the two-step iteration (5.34) can be expressed as

$$X^{(j)} = (A - q_j I)^{-1} \left[(p_j - q_j)F + (A - p_j I)X^{(j-1)}(B + q_j I) \right] (B + p_j I)^{-1}. \quad (5.35)$$

Although it is crucial to employ different shifts at each iteration for a faster convergence of ADI (see Section 5.3.1.2) it is noteworthy that if the same shift parameters $(p_j, q_j) = (p, q)$ are used at each iteration, then (5.35) can be viewed as a fixed-point iteration resulting from a rewriting of the Sylvester operator. Indeed, as observed in [Smi68; Wac88], given shift parameters $p, q \in \mathbb{R}$ such that $p \neq q$, the Sylvester operator $\mathcal{P}X = AX + XB$ can be rewritten as

$$\mathcal{P}X = \frac{1}{p - q} \left(\underbrace{(A - qI)X(B + pI)}_{\mathcal{G}(X)} - \underbrace{(A - pI)X(B + qI)}_{\mathcal{N}(X)} \right). \quad (5.36)$$

This splitting suggests the fixed-point iteration

$$(A - qI)X^{(j)}(B + pI) = (p - q)F + (A - pI)X^{(j-1)}(B + qI) \quad (5.37)$$

which is equivalent to (5.35) with shift parameters $(p_j, q_j) = (p, q)$. Assuming that the matrices A and B are diagonalizable (for instance, if $A, B \succ 0$), the convergence speed is governed by

$$\rho(\mathcal{G}^{-1}\mathcal{N}) = \max_{\substack{\lambda_A \in \Lambda(A) \\ \lambda_B \in \Lambda(B)}} \left| \frac{(\lambda_A - p)(\lambda_B + q)}{(\lambda_A - q)(\lambda_B + p)} \right|,$$

where $\rho(\cdot)$ denotes the spectral radius and $\Lambda(\cdot)$ the spectrum.

5.3.1.2 Convergence of ADI and computing the shift parameters

Let $X_\star$ denote the exact solution of the Sylvester equation $AX + XB = F$. The rewriting in (5.35) yields an error equation for the ADI iterates, as given in [BLT09; TW18]:

$$X_\star - X^{(k)} = r_k(A)(X_\star - X^{(0)})r_k(-B)^{-1}, \quad r_k(z) = \prod_{j=1}^k \frac{z - p_j}{z - q_j}. \quad (5.38)$$

Assuming that A and B are diagonalizable, with eigendecompositions $A = V_A \Lambda_A V_A^{-1}$ and $B = V_B \Lambda_B V_B^{-1}$, (5.38) leads to the bound on the approximation error $\|X - X^{(k)}\|_F$:

$$\frac{\|X_\star - X^{(k)}\|_F}{\|X_\star - X^{(0)}\|_F} \leq \|r_k(A)\|_2 \cdot \|r_k(-B)^{-1}\|_2 \leq \kappa_2(V_A) \kappa_2(V_B) \frac{\sup_{z \in \Lambda(A)} |r_k(z)|}{\inf_{z \in \Lambda(-B)} |r_k(z)|}. \quad (5.39)$$

Note that in the case of interest, where A and B are symmetric positive definite, $\kappa_2(V_A) = \kappa_2(V_B) = 1$. As the spectra $\Lambda(A)$ and $\Lambda(B)$ are often unknown and ex-

5.3. Alternating-Direction Implicit (ADI) iteration

pensive to estimate numerically, one replaces them with closed disjoint sets $E \supseteq \Lambda(A)$ and $F \supseteq \Lambda(-B)$. The Wachspress optimal ADI shifts are then defined by the zeros and poles of the rational function that achieves the *Zolotarev number* $Z_k(E, F)$

$$Z_k(E, F) = \inf_{\substack{r_k = h_k/t_k \\ \deg(h_k) = \deg(t_k) = k}} \frac{\sup_{z \in E} |r_k(z)|}{\inf_{z \in F} |r_k(z)|}.$$

Explicit bounds for $Z_k(E, F)$ are known for specific choices of E and F , see [Wac88; BT17; TW18; BT19] and the references therein. As a rule of thumb, $Z_k(E, F)$ rapidly approaches zero as k increases when E and F are well-separated [Sta92]. For symmetric positive definite matrices A and B , taking real intervals $E = [a, b]$ and $F = [c, d]$, an explicit bound on the Zolotarev number is derived in [BT19] as

$$Z_k([a, b], [c, d]) \leq 4 \left[\exp \left(\frac{\pi^2}{2 \log(16\gamma)} \right) \right]^{-2k}, \quad \gamma := \frac{|c - a||d - b|}{|c - b||d - a|}.$$

The corresponding optimal ADI poles can be computed using elliptic integrals of the first kind, as detailed in [EW91; BMS07; Wac13].

To compute the Wachspress optimal ADI shifts, knowledge of the intervals $[a, b]$ and $[c, d]$ is necessary, corresponding to bounds on the spectra of A and B . In practical applications, theoretical bounds are usually difficult to obtain, leading to the need for numerical estimation of extreme eigenvalues. This can be achieved through methods such as power iteration and inverse iteration, or by an Arnoldi process [GV13]. Alternatively, heuristic strategies can be employed to compute suboptimal parameters, such as the ones proposed in [Pen99; Pen00b]. Additional heuristics are available in [KBS14; BMS07] and related references.

5.3.1.3 ADI iteration for generalized Sylvester equations

The ADI iteration can be extended to solve generalized Sylvester equations $AXD + EXB = F$. Assuming invertibility of E and D , the generalized Sylvester equation can be (formally) rewritten as:

$$E^{-1}AX + XBD^{-1} = E^{-1}FD^{-1} \quad (5.40)$$

Following (5.35), an ADI iteration for (5.40) can be expressed as:

$$(A - q_j E)X^{(j)}(B + p_j D) = (p_j - q_j)F + (A - p_j E)X^{(j-1)}(B + q_j D) \quad (5.41)$$

For fixed shift parameters $(p_j, q_j) = (p, q)$, (5.41) can be viewed as a fixed-point iteration derived from the rewriting of the generalized Sylvester operator as

$$AXD + EXB = \frac{1}{p - q} \left((A - qE)X(B + pD) - (A - pE)X(B + qD) \right). \quad (5.42)$$

5.3. Alternating-Direction Implicit (ADI) iteration

It is important to note that inverting E and D is just a formal step and is not necessary during the algorithm's application, where only the matrix pencils (A, E) and (B, D) are used. For the convergence analysis and the calculation of ADI shift parameters, it suffices to replace $\Lambda(A)$ with $\Lambda(E^{-1}A) = \Lambda(A, E)$, and similarly $\Lambda(B)$ with $\Lambda(B, D)$.

5.3.2 Factored ADI (fADI) iteration

The factored ADI (fADI) iteration is equivalent to the ADI iteration with initialization $X^{(0)} = 0$, but computes $X^{(j)}$ implicitly by obtaining a low-rank factorization of it. It was initially proposed for stable Lyapunov equations [Pen99; LW02; Pen00b] and then extended to the Sylvester case [BLT09; TW18]. We describe it here for solving the generalized Sylvester equation $AXD^\top + EXB^\top = F$. Let $F = MN^\top$ with $M \in \mathbb{R}^{m \times \rho}$ and $N \in \mathbb{R}^{n \times \rho}$, after k iterations of fADI $X^{(k)}$ is expressed as $X^{(k)} = W^{(k)}D^{(k)}Y^{(k)\top}$, where

$$\begin{aligned} D^{(k)} &= \text{blkdiag}((p_1 - q_1)I_\rho, (p_2 - q_2)I_\rho, \dots, (p_k - q_k)I_\rho), \\ W^{(k)} &= \begin{bmatrix} \hat{W}^{(1)} & \hat{W}^{(2)} & \dots & \hat{W}^{(k)} \end{bmatrix}, \quad Y^{(k)} = \begin{bmatrix} \hat{Y}^{(1)} & \hat{Y}^{(2)} & \dots & \hat{Y}^{(k)} \end{bmatrix}, \\ \hat{W}^{(1)} &= (A - q_1 E)^{-1}M, \quad \hat{W}^{(j+1)} = (A - q_{j+1}E)^{-1}(A - p_j E)\hat{W}^{(j)}, \\ \hat{Y}^{(j)} &= (B + p_1 D)^{-1}N, \quad \hat{Y}^{(j+1)} = (B + p_{j+1}D)^{-1}(B + q_j D)\hat{Y}^{(j)}. \end{aligned} \quad (5.43)$$

Each fADI step increases the rank of the tentative solution $X^{(j)}$ by ρ . Notably, the dominant computational cost in each iteration is the solution of ρ linear systems with matrices $(A - q_j E)$ and $(B + p_j D)$, in contrast to n and m as in the ADI iteration (5.41). If $\rho \ll n, m$, that is, if the right-hand-side F is low-rank, the computational cost of fADI is significantly lower than that of ADI.

5.3.3 ADI-like iteration on the tangent space (tangADI)

Our objective is to develop an ADI-like iteration for solving the projected generalized Sylvester equation

$$\text{Proj}_X(A\xi D + E\xi B) = \eta \quad \xi \in \text{T}_X \mathcal{M}_r, \quad (5.44)$$

where $\eta \in \text{T}_X \mathcal{M}_r$ is given and $A, B, D, E \succ 0$. Starting from the simpler case of $E = I_m$ and $D = I_n$, one might consider applying the ADI iteration using the splitting

$$\text{Proj}_X(A\xi + \xi B) = \underbrace{\text{Proj}_X(A\xi)}_{\mathcal{H}_X(\xi)} + \underbrace{\text{Proj}_X(\xi B)}_{\mathcal{V}_X(\xi)}.$$

However, due to the projection in between, the symmetric positive definite linear operators $\mathcal{H}_X, \mathcal{V}_X : \text{T}_X \mathcal{M}_r \rightarrow \text{T}_X \mathcal{M}_r$ do not commute. Despite this, we can still leverage the interpretation of ADI as a fixed-point iteration. By the linearity of the projection, the rewriting in (5.42) extends to (5.44). Thus, we define the following iteration, which

5.3. Alternating-Direction Implicit (ADI) iteration

we will refer to as the *tangADI* iteration:

$$\text{Proj}_X\left((A - q_j E)\xi^{(j)}(B + p_j D)\right) = \text{Proj}_X\left((A - p_j E)\xi^{(j-1)}(B + q_j D)\right) + (p_j - q_j)\eta. \quad (5.45)$$

While theoretical justification and convergence results are established only for the fixed-point iteration, that is, repeating the same shifts parameters $(p_j, q_j) = (p, q)$, a faster convergence is often observed in practice when employing multiple shift parameters, akin to the standard ADI iteration. Note that to recover the implicitly defined ξ_j in (5.45), techniques discussed in Section 5.1.1 can be employed.

5.3.3.1 Convergence and computing the shift parameters

We express the tangADI iteration (5.45) as

$$\tilde{\mathcal{G}}_j(\xi^{(j)}) = \tilde{\mathcal{N}}_j(\xi^{(j-1)}) + (p_j - q_j)\eta,$$

where

$$\begin{aligned} \mathcal{G}_j(X) &= (A - q_j E)X(B + p_j D), & \tilde{\mathcal{G}}_j &:= \text{Proj}_X \circ \mathcal{G}_j \circ \text{Proj}_X : \text{T}_X \mathcal{M}_r \rightarrow \text{T}_X \mathcal{M}_r, \\ \mathcal{N}_j(X) &= (A - p_j E)X(B + q_j D), & \tilde{\mathcal{N}}_j &:= \text{Proj}_X \circ \mathcal{N}_j \circ \text{Proj}_X : \text{T}_X \mathcal{M}_r \rightarrow \text{T}_X \mathcal{M}_r. \end{aligned}$$

If $\xi_\star$ denotes the exact solution of (5.44), we obtain the error equation

$$\xi_\star - \xi^{(k)} = \tilde{\mathcal{G}}_k^{-1} \tilde{\mathcal{N}}_k(\xi_\star - \xi^{(k-1)}) = \dots = \tilde{\mathcal{G}}_k^{-1} \tilde{\mathcal{N}}_k \tilde{\mathcal{G}}_{k-1}^{-1} \tilde{\mathcal{N}}_{k-1} \dots \tilde{\mathcal{G}}_1^{-1} \tilde{\mathcal{N}}_1(\xi_\star - \xi^{(0)}).$$

If the fixed-point iteration is employed, i.e., $(p_j, q_j) = (p, q)$ for all j , then convergence is governed by $\rho(\tilde{\mathcal{G}}_1^{-1} \tilde{\mathcal{N}}_1)$. By the interlacing properties of eigenvalues for definite matrix pencils [LY91; KPS14],

$$\rho(\tilde{\mathcal{G}}_1^{-1} \tilde{\mathcal{N}}_1) \leq \rho(\mathcal{G}_1^{-1} \mathcal{N}_1),$$

indicating that the convergence is at least as fast as ADI with the same pair of shifts $(p_j, q_j) = (p, q)$ used at each iteration. This ensures the convergence of tangADI in its fixed-point version. However, the question remains open regarding the convergence and speed of convergence when different shift parameters are employed at each iteration. In practice, we observed that convergence can be significantly accelerated by employing different shift parameters, choosing them as discussed in the next paragraph.

The previous observation about the convergence of the fixed-point tangADI iteration suggests using the same shift parameters for tangADI as would be used in standard ADI, without the projection to the tangent. While these parameters may be suboptimal, due to the spectrum narrowing with projection, the advantage lies in computing them once, independently of the point X . Subsequently, they can be reused to solve the projected Sylvester equation (5.44) on the tangent at different points $X \in \mathcal{M}_r$. This proves particularly useful in our case, where we aim to employ the tangADI iteration to approximate the action of $\mathcal{P}_X^{-1}$ during preconditioned Riemannian methods, with the

5.3. Alternating-Direction Implicit (ADI) iteration

current iterate X changing.

5.3.3.2 Implementation aspects and complexity

Let us now detail the implementation of the tangADI iteration implicitly defined by (5.45). Let $X = U\Sigma V^\top$ and $\eta \doteq (M_\eta, U_\eta, V_\eta)$. Denote the j -th iterate of tangADI as $\xi^{(j)} \doteq (M_j, U_j, V_j)$, and define $Z_j := (A - p_j E)\xi^{(j-1)}(B + q_j D)$. Using the techniques discussed in Section 5.1.1, we get

$$\begin{aligned} U_{j+1} &= P_U^\perp (A - q_j E)^{-1} (Z_j V + U_\eta + U M_\eta) (V^\top B V + p_j V^\top D V)^{-1} \\ V_{j+1} &= P_V^\perp (B + p_j D)^{-1} (Z_j^\top U + V_\eta + V M_\eta^\top) (U^\top A U - q_j U^\top E U)^{-1} \\ M_{j+1} &= (U^\top A U - q_j U^\top E U)^{-1} \left[U^\top Z_j V + M_\eta - U^\top A U_{j+1} (V^\top B V + p_j V^\top D V) + \right. \\ &\quad \left. + (U^\top A U - q_j U^\top E U) V_{j+1}^\top B V \right] (V^\top B V + p_j V^\top D V)^{-1}. \end{aligned}$$

Efficient computation of $Z_j V$ and $Z_j^\top U$ is possible observing that

$$\begin{aligned} \xi^{(j)} &= \begin{bmatrix} U & U_j \end{bmatrix} \begin{bmatrix} V M_j + V_j & V \end{bmatrix}^\top =: Y_j W_j^\top, \\ Z_j V &= \left[(A + p_j E) Y_j \right] \left[(B + q_j D) W_j \right]^\top V, \\ Z_j^\top U &= \left[(B + q_j D) W_j \right] \left[(A + p_j E) Y_j \right]^\top U. \end{aligned}$$

The two dominant costs of the j -th iteration are:

- The solution of r linear systems with matrices $(A - q_j E)$ and $(B + p_j D)$, with a complexity of $\mathcal{O}(r \cdot c_{(A,E)}(m) + r \cdot c_{(B,D)}(n))$.
- The solution of m linear systems with the dense matrix $(U^\top A U - q_j U^\top E U)$ and n linear systems with the dense matrix $(V^\top B V - p_j V^\top D V)$, which, after a Cholesky factorization, gives a complexity of $\mathcal{O}(r^3) + \mathcal{O}(r^2(m + n)) = \mathcal{O}(r^2(m + n))$.

Therefore, the total complexity of one tangADI step is $\mathcal{O}(r \cdot c_{(A,E)}(m) + r \cdot c_{(B,D)}(n)) + \mathcal{O}(r^2(m + n))$. As already discussed, in many applications $c_{(A,E)}(m) = \mathcal{O}(m)$ and $c_{(B,D)}(n) = \mathcal{O}(n)$, resulting in a total asymptotic complexity of $\mathcal{O}(r^2(m + n))$.

Just as tangADI approximates the action of $\mathcal{P}_X^{-1}$, fADI can serve as an approximate inverse of $\mathcal{P}X = AXD + EXB$ in the Riemannian truncated preconditioned Richardson iteration. Therefore, it is meaningful to compare the per-iteration cost of applying tangADI with the Riemannian gradient as the right-hand side to the per-iteration cost of fADI with the Euclidean gradient as the right-hand side. Observing that the Euclidean gradient (4.4) has low-rank factors with $(\ell r + r_F)$ columns, each iteration of fADI involves solving $(\ell r + r_F)$ linear systems with matrices $(A - q_j E)$ and $(B + p_j D)$, against the r of tangADI. Additionally, after k fADI steps, a matrix of rank $k(\ell r + r_F)$ needs to be projected to the tangent, resulting in an extra final projection cost of $\mathcal{O}(k(\ell r + r_F)r(m + n))$. Moreover, it is worth noting that the memory requirements

5.3. Alternating-Direction Implicit (ADI) iteration

of tangADI remain independent of the number of steps k , while for fADI they increase linearly with k . It is important to emphasize that the cost of fADI can be significantly reduced by truncating the Euclidean gradient (i.e., the residual) using a criterion similar to the one employed for truncating the residual in CG with truncation. However, in this case, the cost of truncation must also be added to the complexity.

Chapter 6

Optimizing over fixed-rank symmetric positive semidefinite matrices

In certain cases, the solution of a multiterm Lyapunov equation is guaranteed to be symmetric positive semidefinite. See, for example, [SZ70] and [Pen98, Theorem 3] for standard and generalized Lyapunov equations, respectively, and [BD11, Theorem 3.9] for the multiterm scenario. Under these conditions, it becomes meaningful to optimize directly on fixed-rank symmetric positive semidefinite matrices. This is the approach taken in [VV10] to develop an efficient solver for generalized Lyapunov equations. Most of the techniques developed in this thesis can be readily adapted to the positive semidefinite case. This chapter provides a brief outline of this adaptation.

6.1 Embedded geometry of fixed-rank symmetric positive semidefinite matrices

We provide a brief overview of the embedded geometry of fixed-rank positive semidefinite matrices. We introduce only the essential tools; for a comprehensive survey and proofs, readers are referred to [Van10, § 3].

Proposition 6.1. *The set $S_{n,r}^+ = \{X \in \mathbb{R}^{n \times n} : X = X^\top, X \succeq 0, \text{rank}(X) = r\}$ is an embedded submanifold of $\mathbb{R}^{n \times n}$ for every $0 \leq r \leq n$. Moreover, $\dim S_{n,r}^+ = nr - r(r-1)/2$.*

Using an eigenvalue decomposition one can express $S_{n,r}^+$ as follows:

$$S_{n,r}^+ = \left\{VDV^\top : V \in \text{St}(n, r), D = \text{diag}(d_1, \dots, d_r), d_1 \geq \dots \geq d_r > 0\right\}. \quad (6.1)$$

Proposition 6.2 (Tangent space). *The tangent space $T_X S_{n,r}^+$ at $X = VDV^\top \in S_{n,r}^+$ is*

6.1. Embedded geometry of fixed-rank symmetric positive semidefinite matrices

given by

$$\begin{aligned} \mathbf{T}_X \mathbf{S}_{n,r}^+ &= \left\{ \begin{bmatrix} V & V_\perp \end{bmatrix} \begin{bmatrix} H & K^\top \\ K & 0 \end{bmatrix} \begin{bmatrix} V & V_\perp \end{bmatrix}^\top : H \in \text{Sym}(r), K \in \mathbb{R}^{(n-r) \times r} \right\} \\ &= \left\{ V H V^\top + V_p V^\top + V V_p^\top : H \in \text{Sym}(r), V_p \in \mathbb{R}^{n \times r}, V_p^\top V = 0 \right\}. \end{aligned} \quad (6.2)$$

To identify a point $X \in \mathbf{S}_{n,r}^+$ and store it in memory, it is convenient to store its eigenvalue decomposition (V, D) . Once we have $X \in \mathbf{S}_{n,r}^+$ represented by (V, D) , a tangent vector $\xi \in \mathbf{T}_X \mathbf{S}_{n,r}^+$ can be represented by storing the matrices (H, V_p) . We consider $\mathbf{S}_{n,r}^+$ as an embedded submanifold of $\mathbb{R}^{n \times n}$ equipped with the Frobenius metric. With this metric, the inner product of $\dot{\xi}, \dot{\xi}' \in \mathbf{T}_X \mathbf{S}_{n,r}^+$, where $\dot{\xi} \doteq (H, V_p)$ and $\dot{\xi}' \doteq (H', V_p')$, can be computed as $\langle \dot{\xi}, \dot{\xi}' \rangle = \langle H, H' \rangle + 2\langle V_p, V_p' \rangle$.

Orthogonal projectors Consider $X = V D V^\top \in \mathbf{S}_{n,r}^+$, and denote $P_V = V V^\top$ and $P_V^\perp = I_n - P_V$. The orthogonal projection of $Z \in \mathbb{R}^{n \times n}$ onto $\mathbf{T}_X \mathbf{S}_{n,r}^+$ is expressed as

$$\text{Proj}_X(Z) = P_V \text{sym}(Z) P_V + P_V \text{sym}(Z) P_V^\perp + P_V^\perp \text{sym}(Z) P_V, \quad (6.3)$$

where $\text{sym}(Z) = (Z + Z^\top)/2$. Additionally, the projection is represented by the matrices:

$$H = V^\top \text{sym}(Z) V, \quad V_p = \text{sym}(Z) V - V H. \quad (6.4)$$

Metric projection retraction As one could expect, the metric projection retraction is not defined on the entire tangent bundle but is locally defined, for $\xi \in \mathbf{T}_X \mathbf{S}_{n,r}^+$ small enough. Specifically, the following theorem is applicable.

Theorem 6.3 ([VV10, Theorem 5.7]). *Let $X \in \mathbb{R}^{n \times n}$ be a symmetric matrix with n_+ positive and n_- negative eigenvalues. Let its eigenvalue decomposition be $X = V \text{diag}(d_1, \dots, d_n) V^\top$ with $d_1 \geq \dots \geq d_{n_+} > 0 > d_{n_+ + 1} \geq \dots \geq d_n$. The best symmetric positive semidefinite approximation of rank r in the Frobenius norm exists if and only if $n_+ \geq r$. One such minimizer is given by $\mathcal{P}_{\mathbf{S}_{n,r}^+}(X) = V \text{diag}(d_1, \dots, d_r, 0, \dots, 0) V^\top$.*

Refer to [VV10, § 5.4] for sufficient conditions on ξ for the existence of $R_X(\xi) = \mathcal{P}_{\mathbf{S}_{n,r}^+}(X + \xi)$ and details on efficient computation.

From Euclidean to Riemannian: gradient and Hessian Since we equipped $\mathbf{S}_{n,r}^+$ with the ambient space metric, the Riemannian gradient of f is obtained by projecting the Euclidean gradient of an extension $\bar{f}$. Using similar ideas as in Proposition 2.28, the following formulas for the Riemannian Hessian are derived. Let $X = V D V^\top \in \mathbf{S}_{n,r}^+$, and let $\xi \in \mathbf{T}_X \mathbf{S}_{n,r}^+$ be represented by (H_ξ, V_ξ) . Denote $S = \text{sym}(\text{grad} \bar{f}(X))$ and $\dot{S} = \text{sym}(\text{Hess} \bar{f}(X)[\xi])$. Then $\text{Hess} f(X)[\xi] \in \mathbf{T}_X \mathbf{S}_{n,r}^+$ is represented by (H, V_p) where

$$H = V^\top \dot{S} V, \quad V_p = P_V^\perp (\dot{S} V + S V_\xi D^{-1}). \quad (6.5)$$

6.2 Preconditioners

In the case of optimization on $S_{n,r}^+$ preconditioners can be developed in a manner similar to Chapter 5. Given that the projection to the tangent (6.3) is such that $\text{Proj}_X(Z) = \text{Proj}_X(\text{sym}(Z))$, it makes sense to consider only preconditioners $\mathcal{P}$ for which $(\mathcal{P}\xi)^\top = \mathcal{P}\xi$ for every symmetric matrix ξ . Therefore, given matrices $A, M \succ 0$, we consider the one-term preconditioner $\mathcal{P}X = MXM$, the Lyapunov preconditioner $\mathcal{P}X = AX + XA$ and the generalized Lyapunov preconditioner $\mathcal{P}X = AXM + MXA$.

6.2.1 $\mathcal{P}X = MXM$ where $M \succ 0$

6.2.2 Applying $\mathcal{P}_X^{-1}$

Analogously to the procedure outlined in Section 5.1.1, consider $X \in S_{n,r}^+$ and a tangent vector $\eta \in T_X S_{n,r}^+$ represented by the tuple (H_η, V_η) . The solution to the equation

$$\text{Proj}_X(M\xi M) = \eta \quad \xi \in T_X S_{n,r}^+,$$

is given by $\xi \doteq (H_\xi, V_\xi)$, where

$$\begin{aligned} V_\xi &= P_V^\perp M^{-1}(VH_\xi + V_\xi), \\ H_\xi &= (V^\top MV)^{-1} \left[M_\xi - (V^\top MV)(V_\xi^\top MV) - \left((V^\top MV)(V_\xi^\top MV) \right)^\top \right] (V^\top MV)^{-1}. \end{aligned}$$

6.2.2.1 Geometry with the metric induced by $\mathcal{P}X = MXM$

The Riemannian optimization tools for $S_{n,r}^+$ as a Riemannian submanifold of $\mathbb{R}^{n \times n}$ with the metric induced by $\mathcal{P}X = MXM$ are developed similarly to Section 5.1.2.

Storing points and tangent vectors A point $X \in S_{n,r}^+$ is represented by the thin decomposition $X = \tilde{V}\tilde{D}\tilde{V}^\top$ where $\tilde{V}^\top M\tilde{V} = I_r$ and $\tilde{D} = \text{diag}(\tilde{d}_1, \dots, \tilde{d}_r)$, $\tilde{d}_1 \geq \dots \geq \tilde{d}_r > 0$. Note that here $\tilde{d}_1, \dots, \tilde{d}_r$ are not the nonzero eigenvalues of X . They are the nonzero eigenvalues of $C_M X C_M^\top$, where $M = C_M^\top C_M$ is the Cholesky decomposition of M . Tangent vectors $\xi \in T_X S_{n,r}^+$ can be written as

$$\xi = \tilde{V}\tilde{H}\tilde{V}^\top + \tilde{V}_p\tilde{V}^\top + \tilde{V}\tilde{V}_p^\top \quad \text{where} \quad \tilde{V}_p^\top M\tilde{V} = 0, \tilde{H} = \tilde{H}^\top.$$

The inner product between $\dot{\xi}, \dot{\xi}' \in T_X S_{n,r}^+$ with $\dot{\xi} \doteq (\tilde{H}, \tilde{V}_p)$ and $\dot{\xi}' \doteq (\tilde{H}', \tilde{V}_p')$ is then computed as $\langle \dot{\xi}, \dot{\xi}' \rangle = \langle \tilde{H}, \tilde{H}' \rangle + 2\langle \tilde{V}_p, M\tilde{V}_p' \rangle$.

Orthogonal projectors Let $X = \tilde{V}\tilde{D}\tilde{V}^\top \in S_{n,r}^+$, and define $P_V^\mathcal{P} = \tilde{V}\tilde{V}^\top M$ and $P_V^{\perp, \mathcal{P}} = I_n - P_V^\mathcal{P}$. The orthogonal projection of $Z \in \mathbb{R}^{n \times n}$ onto $T_X S_{n,r}^+$, with respect to

the inner product induced by $\mathcal{P}X = MXM$, is represented by

$$\begin{aligned}\tilde{H} &= \tilde{V}^\top M \operatorname{sym}(Z) M \tilde{V} \\ \tilde{V}_p &= P_{\tilde{V}}^{\perp, \mathcal{P}} \operatorname{sym}(Z) M \tilde{V} = \operatorname{sym}(Z) M \tilde{V} - \tilde{V} \tilde{H}.\end{aligned}\tag{6.6}$$

Metric projection to $S_{n,r}^+$ Let $X \in \mathbb{R}^{n \times n}$ be a symmetric matrix with n_+ positive and n_- negative eigenvalues, and let $X = \tilde{V} \operatorname{diag}(\tilde{d}_1, \dots, \tilde{d}_n) \tilde{V}^\top$, with $\tilde{V} \in \mathbb{R}^{n \times n}$ M -orthogonal and $\tilde{d}_1 \geq \dots \geq \tilde{d}_{n_+} > 0 > \tilde{d}_{n_+ - n_- + 1} \geq \dots \geq \tilde{d}_n$. The metric projection onto $S_{n,r}^+$ in the metric induced by $\mathcal{P}X = MXM$ exists if and only if $n_+ \geq r$, and is given by $\mathcal{P}_{S_{n,r}^+}(X) = \tilde{V} \operatorname{diag}(\tilde{d}_1, \dots, \tilde{d}_r, 0, \dots, 0) \tilde{V}^\top$.

Gradient and Hessian: from Euclidean to Riemannian Let $f : S_{n,r}^+ \rightarrow \mathbb{R}$ be a smooth function with a smooth extension $\bar{f}$. Define $S = \operatorname{sym}(\operatorname{grad} \bar{f}(X))$ and $\dot{S} = \operatorname{sym}(\operatorname{Hess} \bar{f}(X)[\xi])$, where $\xi \in T_X S_{n,r}^+$ and is represented by the tuple $(\tilde{H}_\xi, \tilde{V}_\xi)$. The Riemannian gradient of f at $X = \tilde{V} \tilde{D} \tilde{V}^\top$, with respect to the metric induced by $\mathcal{P}$, is $\operatorname{grad}_{\mathcal{P}} f(X) \doteq (\tilde{H}, \tilde{V}_p)$, where

$$\tilde{H} = \tilde{V}^\top S \tilde{V}, \quad \tilde{V}_p = M^{-1} S \tilde{V} - \tilde{V} \tilde{H} = M^{-1} [S \tilde{V} - M \tilde{V} \tilde{H}].\tag{6.7}$$

The Riemannian Hessian $\operatorname{Hess}_{\mathcal{P}} f(X)[\xi]$ is represented by

$$\tilde{H} = \tilde{V}^\top \dot{S} \tilde{V}, \quad \tilde{V}_p = M^{-1} (I - M \tilde{V} \tilde{V}^\top) [\dot{S} \tilde{V} + S \tilde{V}_\xi \tilde{D}^{-1}].\tag{6.8}$$

6.2.3 $\mathcal{P}X = AXM + MXA$ where $A, M \succ 0$

The application of the Lyapunov preconditioner $\mathcal{P}X = AX + XA$ is thoroughly discussed in [VV10, § 7]. While this preconditioner is employed in the same paper as an approximation for $\mathcal{P}X = AXM + MXA$, particularly when M is as a mass matrix, experiments detailed in [VV10, § 8.2.3] suggest that if the condition number of M is not close to 1, such as in cases of non-uniform meshes, the approximation $M \approx I$ is overly simplistic and does not yield an efficient solver for generalized Lyapunov equations. Consequently, in this work, we extended our focus to generalized Lyapunov preconditioners of the form $\mathcal{P}X = AXM + MXA$.

Analogous to the discussion in Section 5.2.2, we are able to apply the preconditioner $\mathcal{P}X = AXM + MXA$ by expressing it as $\mathcal{P} = \mathcal{B} \tilde{\mathcal{P}}$, where $\mathcal{B}X = MXM$, $\tilde{\mathcal{P}}X = \mathcal{B}^{-1} \mathcal{P}X = M^{-1}AX + XAM^{-1}$, and leveraging the Riemannian optimization tools in the metric induced by $\mathcal{B}$. The detailed calculations parallel those in Section 5.2.2, and we refrain from reproducing them here.

6.2.4 tangADI

For the approximate resolution of the projected generalized Lyapunov equation $\operatorname{Proj}_X(A\xi M + M\xi A) = \eta \in T_X S_{n,r}^+$, where $\xi \in T_X S_{n,r}^+$, an ADI-like iteration on the

tangent space can be developed, similarly to Section 5.3.3. The tangADI iteration for the Lyapunov case is given by:

$$\text{Proj}_X\left((A + p_j M)\xi^{(j)}(A + p_j M)\right) = \text{Proj}_X\left((A - p_j M)\xi^{(j-1)}(A - p_j M)\right) + 2p_j\eta, \quad (6.9)$$

where for an explicit computation of $\xi^{(j)}$ one can proceed analogously to Section 6.2.2.

6.3 Rank adaptivity

To transition from fixed-rank Riemannian optimization to a rank-adaptive algorithm, we can employ an approach analogous to that outlined in Section 4.4, i.e., alternating between fixed-rank Riemannian optimization and rank updates. The only difference lies in the choice of the search direction for the rank increase step, which is determined as the projection onto the *positive semidefinite* matrices of rank r_{up} of the component orthogonal to the tangent of the negative gradient. This is well-defined if the normal component of the negative gradient has at least r_{up} positive eigenvalues. In cases where this condition is not met, random components can be added until the desired rank is attained. All other aspects of the RRAM remain unchanged.

Chapter 7

Numerical Experiments

In this chapter, we conduct an analysis of the performance of preconditioned Riemannian optimization methods on three representative model problems. The first model problem serves as a benchmark for a detailed evaluation of line-search Riemannian methods, model-Hessians for R-TR, and different preconditioners. For the second and third examples, we directly compare the performance of the most effective Riemannian optimization methods with established numerical methods for multiterm linear matrix equations.

We implemented all algorithms and preconditioners in MATLAB, R2023a. The preconditioned Riemannian methods are implemented using Manopt [Bou+13]. In R-NLCG, corresponding to Manopt’s `conjugategradient` solver, we implemented the Hestenes-Stiefel modified rule for β_k in the preconditioned case (3.24). This implementation will be included in the upcoming release of Manopt.

To assess the performance of numerical methods with a different nature, relying solely on iteration counts may be misleading, hence we often conduct a comparison based on timings. To make this comparison as fair as possible, we have taken care of optimizing the implementations and the parameters of all methods to a similar extent. The reported timings are wall times, inclusive of all necessary precomputations for preconditioning, such as calculating sparse Cholesky factorizations with reordering, spectral bounds, or ADI shifts. To compare different algorithms, we frequently examine the decrease of the relative residual in Frobenius norm, defined as

$$\text{res}(X) = \frac{\|\mathcal{A}X - F\|_F}{\|F\|_F}. \quad (7.1)$$

The computation time of $\text{res}(X_k)$ is not included in the timing of the algorithm, except for CG with truncation and RRAM update steps, both of which use it for the stopping criterion and compute it as a side effect of truncation.

All numerical experiments and timings are conducted on an Intel® Core™ i7-10750H 2.6 GHz CPU, featuring 6 cores, operating on a 64-bit Windows machine with 16 GB RAM.

7.1 Finite difference discretization of 2D PDEs with separable variables

As a first test problem, we consider a stationary diffusion equation on a square domain with Dirichlet boundary conditions

$$\begin{cases} -\nabla \cdot (k \nabla u) = f & \text{in } \Omega \\ u = g & \text{on } \partial\Omega \end{cases}, \quad (7.2)$$

where $\Omega = [0, 1] \times [0, 1]$ is the spatial domain, and the diffusion coefficient k and the source term f have separable variables

$$k(x, y) = 1 + \sum_{j=1}^{\ell_k} \alpha_j k_{j,x}(x) k_{j,y}(y), \quad f(x, y) = \sum_{j=1}^p \beta_j f_{j,x}(x) f_{j,y}(y). \quad (7.3)$$

Now, consider a finite difference discretization on a uniform mesh with mesh size $h = 1/(n+1)$. Let $U \in \mathbb{R}^{n \times n}$, where $U_{s,k} \approx u(x_s, x_k)$ and $x_i = ih$. The resulting matrix equation reads:

$$PU + UP + \sum_{j=1}^{\ell_k} \alpha_j (A_{j,x} U D_{j,y}^\top + D_{j,x} U A_{j,y}^\top) = F \quad (7.4)$$

where P is the discretization of the two-dimensional Laplacian. The other matrices on the left-hand side are defined as (using a MATLAB-like notation):

$$\begin{aligned} A_{j,z} &= \frac{1}{h^2} \text{tridiag} \left(\left[-k_{j,z}(x_{i-\frac{1}{2}}), k_{j,z}(x_{i+\frac{1}{2}}) + k_{j,z}(x_{i-\frac{1}{2}}), -k_{j,z}(x_{i+\frac{1}{2}}) \right], -1 : 1 \right), \\ D_{j,z} &= \text{diag} (k_{j,z}(x_1), k_{j,z}(x_2), \dots, k_{j,z}(x_n)), \end{aligned} \quad (7.5)$$

with $z = x, y$. The right-hand-side F is given by:

$$\begin{aligned} F &= \sum_{j=1}^p \beta_j \mathbf{f}_{j,x} \mathbf{f}_{j,y}^\top + e_1 \mathbf{b}_l^\top + e_n \mathbf{b}_r^\top + b_d e_1^\top + b_u e_n^\top, \\ [\mathbf{f}_{j,z}]_i &= f_{j,z}(x_i), \\ [b_u]_i &= k(x_i, 1 - h/2) g(x_i, 1), \quad [b_d]_i = k(x_i, h/2) g(x_i, 0), \\ [b_r]_j &= k(1 - h/2, x_j) g(1, x_j), \quad [b_l]_j = k(h/2, x_j) g(0, x_j). \end{aligned}$$

Observe that if $k > 0$ in Ω , the linear operator defined by (7.4) is symmetric positive definite. Moreover, for each j and for $z = x, y$, the matrices $A_{j,z}$ and $D_{j,z}$ are symmetric. If $k_{j,z} > 0$, then $A_{j,z}$ and $D_{j,z}$ are also positive definite.

For our numerical experiments we consider the following diffusion coefficient, source

7.1. Finite difference discretization of 2D PDEs with separable variables

term and boundary data:

$$k(x, y) = 1 + \sum_{i=1}^{\ell_k} \frac{\alpha^i}{i!} x^i y^i, \quad f(x, y) = 0, \quad g(x, y) = e^{-\alpha xy}. \quad (7.6)$$

Observe that $k(x, y)$ represents the truncated power series expansion of $e^{\alpha xy}$ around 0, and if $\ell_k = +\infty$, the exact solution becomes $u(x, y) = g(x, y) = e^{-\alpha xy}$. We set $\ell_k = 3$, $\alpha = 10$. The corresponding multiterm linear matrix equation has $\ell = 8$ terms, and the right-hand-side F has rank $r_F = 4$. The discretization mesh-size is $h = 1/(n + 1)$ with $n = 10000$.

Preconditioning A general approach to deriving preconditioners for (7.4) involves approximating the diffusion coefficient k with a separable variable term, denoted as $k_a(x, y) = a_x(x)a_y(y)$, and constructing the preconditioner based on the discretization of k_a . To be an effective preconditioner, the discretization of $k_a(x, y)$ should retain as much as possible of the ill-conditioning arising from the discretization of $k(x, y)$.

The most straightforward approximation for $k(x, y)$ is its mean, denoted as $k_a(x, y) = \bar{k} := \int_{\Omega} k(x, y) dx dy$. This results in the preconditioner $\mathcal{P}X = TX + XT$ with $T = \bar{k}P$. However, for the diffusion coefficient specified in (7.6), this mean-based preconditioner becomes less effective as α increases. This is because as α grows, the deviation of $k(x, y)$ from its mean becomes more pronounced.

Another preconditioner arises by approximating k with the dominant separable term $k_a(x, y) = \left(1 + (\sqrt{\alpha}x)^\ell\right) \left(1 + (\sqrt{\alpha}y)^\ell\right)$. This leads to a preconditioner of the form $\mathcal{P}^{(2)}X = AXD + DXA$, where A and D are defined as in (7.5). Following [Van10, § 4.6.2], a one-sided Lyapunov preconditioner can be obtained neglecting D , i.e., taking $\mathcal{P}^{(1)}X = AX + XA$. In our experiments, this one-sided Lyapunov preconditioner performed better than the mean-based one $\mathcal{P}X = TX + XT$. The reason is that neglecting the matrix D is not too harmful because it is diagonal with condition number in 2-norm bounded by $1 + \alpha^{\ell/2}$, which is relatively low and independent of mesh-size.

7.1.1 Numerical results

We aim to achieve a relative residual less than $\text{tol} = 10^{-6}$. In hindsight, a rank of $r = 15$ proves to be sufficient to meet this tolerance. Consequently, for fixed-rank Riemannian optimization, we employ a rank of $r = 15$. The initial iterate X_0 for Riemannian optimization is randomly drawn with a Frobenius norm of 1, and it is the same for all Riemannian optimization methods. We present the results of a single run, noting that the convergence trend proved to be essentially independent of the initialization.

We determine the truncation tolerances in CG with truncation (Algorithm 1) by aiming to strike a balance between consistent truncation and achieving a relative residual of tol . The following values have shown satisfactory performance. For truncation of the iterates X_k , we set a relative truncation tolerance $\epsilon_{\text{rel}} = 0.0025 \cdot \text{tol}$. To truncate

7.1. Finite difference discretization of 2D PDEs with separable variables

R_k and P_k we set $\epsilon_{\text{rel}} = 0.1 \cdot \text{tol}$ and $\epsilon_{\text{abs}} = 0.001 \cdot \text{tol}$, where the notation is the same as in (1.12). Finally, we do not truncate Q_k .

In Riemannian line-search methods, we halt optimization if the gradient norm falls below $\text{tolgradnorm} = \text{tol} \cdot \|F\|_{\text{F}}/2$. Alternatively, we stop if the step-size computed with Armijo backtracking becomes less than $\alpha_{\min} = 10^{-4}$, or if we reach 150 iterations. In the case of Riemannian optimization using the tools in the metric induced by $\mathcal{P}X = DXD$, we rescale tolgradnorm by a factor $1/\|D\|_2$, to account for all calculations being performed in the $\mathcal{P}$ -induced norm.

In Riemannian Trust-Region, we terminate optimization if the gradient norm drops below the same tolgradnorm as above or if we reach 150 iterations. All other parameters of R-TR (Algorithm 6) are set to Manopt's defaults. Specifically, we set $\rho' = 0.1$, $\kappa = 0.1$, and $\theta = 1$. The maximum trust-region radius is set to $\bar{\Delta} = \sqrt{\dim \mathcal{M}_r} = \sqrt{(m+n-r)r}$, and the initial radius is $\Delta_0 = \bar{\Delta}/8$. Additionally, we limit the number of inner iterations of tCG to 250. While this limitation might impede superlinear convergence, in our numerical experiments it is never reached when an effective preconditioner is employed.

Comparison of line-search methods and preconditioners In Figure 7.1 we compare different preconditioners applied to line-search Riemannian methods. Specifically, we consider the Lyapunov one-sided preconditioner $\mathcal{P}^{(1)}X = AX + XA$ and the two-sided one $\mathcal{P}^{(2)}X = AXD + DXA$. For the one-sided preconditioner, we present only the most effective Riemannian method, namely R-NLCG with the exact application of $(\mathcal{P}_X^{(1)})^{-1}$. For the more effective two-sided preconditioner, we compare R-GD and R-NLCG with the exact application of $(\mathcal{P}_X^{(2)})^{-1}$ (using tools from the metric induced by $X \mapsto DXD$), R-NLCG with 4 steps of tangADI as a preconditioner, and R-NLCG with Riemannian truncated preconditioned Richardson search direction, using 4 steps of fADI as a preconditioner. The number of tangADI and fADI steps has been independently optimized to minimize the convergence time. Optimal Wachspress shifts are employed in both tangADI and fADI.

The comparison of R-NLCG with the two preconditioners clearly indicates that the quality of the preconditioner strongly influences convergence. The two-sided preconditioner $\mathcal{P}^{(2)}$ results in a drastic improvement in convergence. This highlights the effectiveness of the techniques introduced in this thesis for dealing with the two-sided case. Additionally, the use of R-NLCG over R-GD contributes to significantly reducing the number of iterations, with a similar reduction observed also in time.

Both tangADI and the Riemannian truncated preconditioned Richardson with fADI as a preconditioner exhibit a slower convergence rate compared to preconditioning via the exact application $(\mathcal{P}_X^{(2)})^{-1}$. In the case of Riemannian truncated preconditioned Richardson, to reduce the cost of fADI, we incorporate truncation of the Euclidean gradient (i.e., the residual) with a criterion similar to the one used for truncating the residual in CG with truncation. However, despite this adjustment, tangADI enjoys a lower per-iteration cost on average, resulting in lower timings.

7.1. Finite difference discretization of 2D PDEs with separable variables

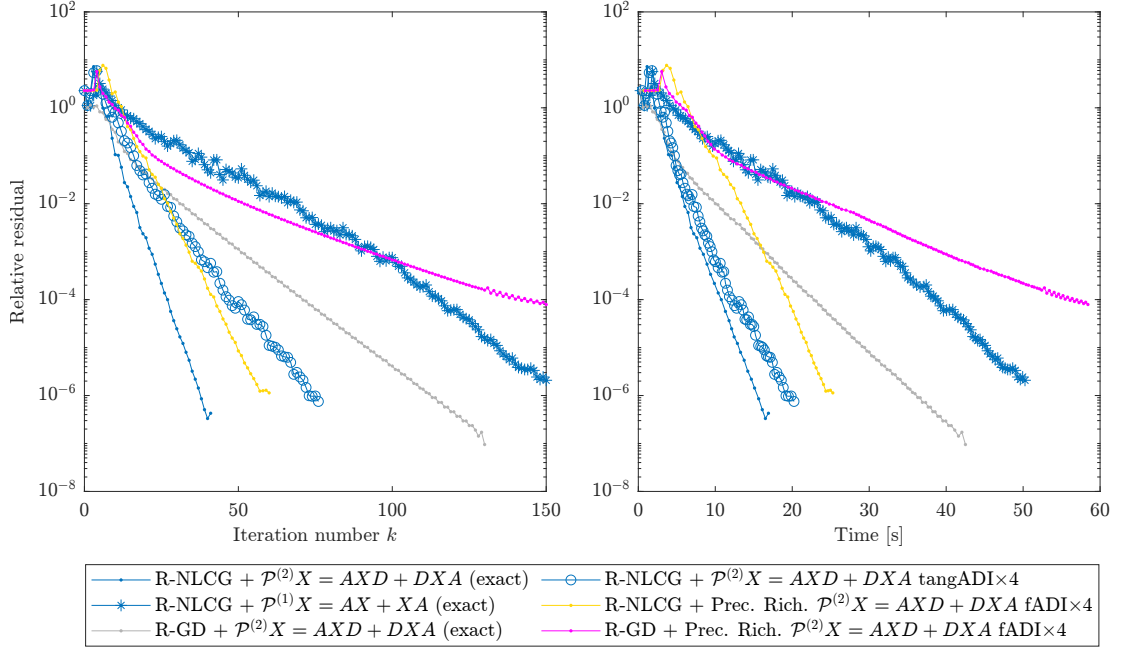


Figure 7.1: FD of 2D PDEs with separable variables, with $\alpha = 10$ and $n = 10000$. Preconditioned Riemannian line-search methods on matrices of rank $r = 15$. Lyapunov preconditioner $\mathcal{P}^{(1)}X = AX + XA$: optimization using exact $(\mathcal{P}_X^{(1)})^{-1}$. Two-sided preconditioner $\mathcal{P}^{(2)}X = AXD + DXA$: exact $(\mathcal{P}_X^{(2)})^{-1}$ (using tools in the metric induced by $X \mapsto DXD$), 4 steps of tangADI, and Riemannian truncated preconditioned Richardson search direction with 4 fADI steps. Left: relative residual vs. iterations. Right: relative residual vs. time.

Comparison of R-TR model Hessians and preconditioners We conduct a comparison of different model Hessians and preconditioners for Riemannian Trust-Region. Results are shown in Figure 7.2.

Initially, we compare the Riemannian Hessian $H_X^{(R)} = \text{Hess}f(X)$ and the projected Euclidean Hessian $H_X^{(E)} = \text{Proj}_X \circ \mathcal{A} \circ \text{Proj}_X$ as model Hessians, both utilizing the two-sided preconditioner $\mathcal{P}^{(2)}X = AXD + DXA$. Notably, the projected Euclidean Hessian exhibits convergence in a remarkably low number of iterations, while the convergence rate of R-TR with the Riemannian Hessian is considerably slower. Additionally, when employing the Riemannian Hessian, we observed that during the solution of the trust-region subproblem, tCG halts after only a few iterations, either due to detecting a negative eigenvalue or reaching the trust-region boundary, leading to continuous reduction or enlargement of the trust-region's radius.

A plausible explanation for this phenomenon is as follows: the Riemannian Hessian at $X = U\Sigma V^\top$, computed in (2.23), consists of the sum of the projected Euclidean Hessian (which is positive definite) and a curvature term dependent on the normal component of $\text{grad}\bar{f}(X) = \mathcal{A}X - F$ and on Σ^{-1} . This curvature term is always zero or indefinite, with eigenvalues characterized in Proposition 2.30, and can render the Riemannian Hessian indefinite. Suppose we chose $r = 15$ correctly, meaning the exact

7.1. Finite difference discretization of 2D PDEs with separable variables

solution $X_\star$ has a numerical rank of r . Then its r -th singular value σ_r will likely be close to 0, say $\sigma_r \approx \epsilon_{\text{mach}}$. Suppose also that the sequence generated by R-TR converges to $X_\star$. In theory, there exists a neighborhood of $X_\star$ where the Riemannian Hessian is positive definite, and where we observe local quadratic convergence. However, for the projected Euclidean Hessian to dominate the curvature term, so that the Riemannian Hessian is guaranteed to be positive definite, we need to approach $X_\star$ closely enough that the reduction in the norm of the normal residual compensates for the fact that Σ^{-1} has (at least) one entry in the order of σ_r^{-1} . If $\sigma_r \approx \epsilon_{\text{mach}}$, achieving this accuracy is challenging in practice. Moreover, as X_k approaches $X_\star$, the linear systems with Σ^{-1} become ill-conditioned, and this is responsible for the final oscillations of R-TR with Riemannian Hessian. We observed that regularizing the Hessian partially removed the oscillations but did not improve convergence.

When employing the projected Euclidean Hessian, the number of outer R-TR iterations is not much by the preconditioner. Nevertheless, employing an effective preconditioner significantly reduces the number of inner tCG iterations, consequently reducing the overall convergence time. This difference is particularly evident close to convergence, where accurate TRs resolution is required. Additionally, the superior performance of the exact preconditioner over tangADI can also be attributed to the fact that, with X fixed, subsequent applications of $(\mathcal{P}_X^{(2)})^{-1}$ are considerably less expensive.

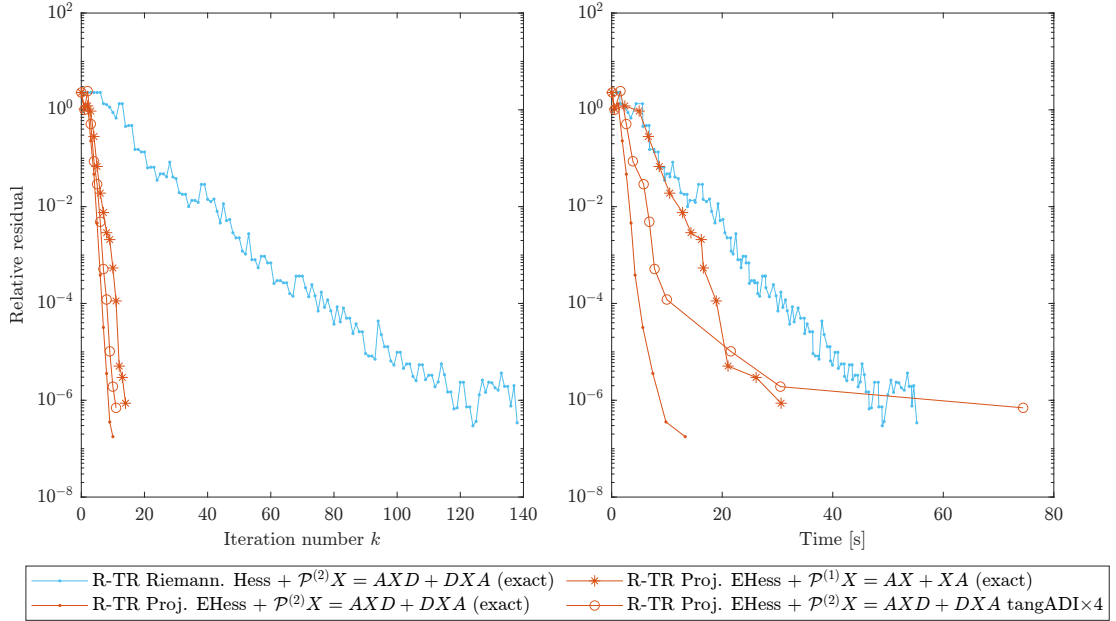


Figure 7.2: FD of 2D PDEs with separable variables, with $\alpha = 10$ and $n = 10000$. Preconditioned Riemannian Trust-Region on matrices of rank $r = 15$. Comparison of Riemannian Hessian and projected Euclidean Hessian as model Hessians for R-TR, along with different preconditioners. Lyapunov preconditioner $\mathcal{P}^{(1)}X = AX + XA$: optimization using exact $(\mathcal{P}_X^{(1)})^{-1}$. Two-sided preconditioner $\mathcal{P}^{(2)}X = AXD + DXA$: exact $(\mathcal{P}_X^{(2)})^{-1}$ (using tools in the metric induced by $X \mapsto DXD$) and 4 steps of tangADI. Left: relative residual vs. iterations. Right: relative residual vs. time.

7.1. Finite difference discretization of 2D PDEs with separable variables

Comparison of Riemannian optimization and CG with truncation Figure 7.3 presents a comparison of the best fixed-rank Riemannian methods, CG with truncation, CG with truncation and rank cap at $r = 15$, and Riemannian rank-adaptive methods. For RRAMs, the initial rank is set to $r_0 = 3$ and the rank increment to $r_{\text{up}} = 3$.

Among fixed-rank methods, the most effective method is R-TR with projected Euclidean Hessian, outperforming R-NLCG in iterations. Since the matrix equation involves $\ell = 8$ terms, in this case, approximately inverting the entire projected Euclidean Hessian $\mathcal{A}_X$ as in the R-TR step, rather than just the dominant term $\mathcal{P}_X^{(2)}$ as in preconditioned R-NLCG, proves beneficial. Moreover, the cost of the inner iterations of tCG compared to a single application of $(\mathcal{P}_X^{(2)})^{-1}$ is not excessive because subsequent applications of $(\mathcal{P}_X^{(2)})^{-1}$ are less expensive, resulting in a lower overall timing for R-TR. CG with rank cap lags behind fixed-rank Riemannian methods, underscoring the necessity of a proper Riemannian optimization approach over truncation at a fixed rank.

Among methods where the rank is not known a priori, CG with truncation demonstrates rapid convergence at a consistent rate in terms of iterations but incurs extensive computation time due to excessive growth of intermediate ranks. Riemannian rank-adaptive methods exhibit faster convergence in time, with RRAM using R-NLCG emerging as the fastest among the examined methods. Additionally, the final rank of the approximate solution is considerably lower for RRAM than for CG with truncation. However, both RRAMs exhibit oscillatory convergence, characterized by plateaus and significant fluctuations after rank changes, and the iteration count is higher than CG.

We remark that the computation time of CG with truncation could substantially benefit from the application of randomized techniques for computing the TSVD [YGL18; HMT11], which have a linear rather than quadratic cost in $(\ell r_{\text{curr}} + r_F)$, where r_{curr} is the rank of the current CG iterate. This improvement would also extend to the rank-increase steps in RRAMs. Finally, note that the oscillations observed in the initial iterations of the Riemannian methods are likely a result of random initialization and could be reduced with a more informed initialization.

Quadratic convergence of R-TR and stagnation of Riemannian truncated preconditioned Richardson Finally, we analyze the influence of the rank r on the convergence of fixed-rank Riemannian optimization.

Figure 7.4 illustrates that when a small rank is used ($r = 5$ in our case), the Riemannian truncated preconditioned Richardson iteration can stagnate and fail to reach convergence. In this case, it stops even before reaching the maximum number of iterations because the projection of the preconditioned Richardson search direction becomes an ascent direction for Riemannian optimization. This phenomenon does not occur at rank $r = 40$. Referring to the discussion in Section 3.4.6, a reasonable explanation for this lies in the fact that the error introduced by using the "wrong projection" is more evident when projecting onto a smaller tangent space.

Furthermore, when the r -th singular value of the exact solution is not too small, as it is the case for $r = 5$, one can experimentally observe the local superlinear convergence

7.2. Stochastic Galerkin matrix equations

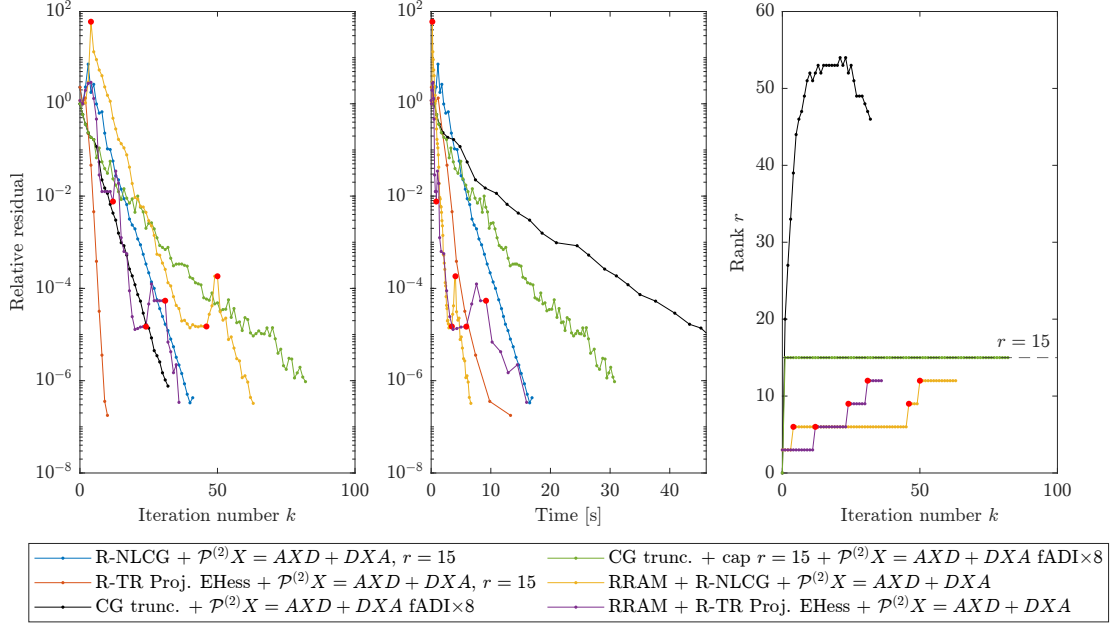


Figure 7.3: FD of 2D PDEs with separable variables, with $\alpha = 10$ and $n = 10000$. Comparison of fixed-rank Riemannian optimization methods on matrices of rank $r = 15$, CG with truncation, CG with truncation and rank cap at $r = 15$, and RRAMs. Red dots in RRAMs' plots indicate rank increases. From left to right: relative residual vs. iterations, relative residual vs. time, and rank of approximate solution vs. iterations.

of R-TR with Riemannian Hessian when approaching the solution closely enough. We remark that despite $r = 5$ being a relatively low rank, all convergent methods reach a relative residual of about $5 \cdot 10^{-5}$. In contrast, at a high rank like $r = 40$, superlinear convergence of R-TR with projected Euclidean Hessian is observed, while superlinear convergence with Riemannian Hessian cannot be experimentally observed.

7.2 Stochastic Galerkin matrix equations

We consider a parameterized steady-state diffusion equation with homogeneous Dirichlet boundary conditions, given by

$$\begin{cases} -\nabla \cdot (a(x, y) \nabla u(x, y)) = f(x) & \text{in } \Omega \times \Gamma \\ u(x, y) = 0 & \text{on } \partial\Omega \end{cases} \quad (7.7)$$

where the spatial domain is Ω , the parametric domain is $\Gamma = [-1, 1]^q$, and $a, f : \Omega \times \Gamma \rightarrow \mathbb{R}$ have the form:

$$a(x, y) = a_0 + \sum_{k=1}^q a_k(x) y_k, \quad f(x, y) = f_0(x) + \sum_{k=1}^q f_k(x) y_k. \quad (7.8)$$

7.2. Stochastic Galerkin matrix equations

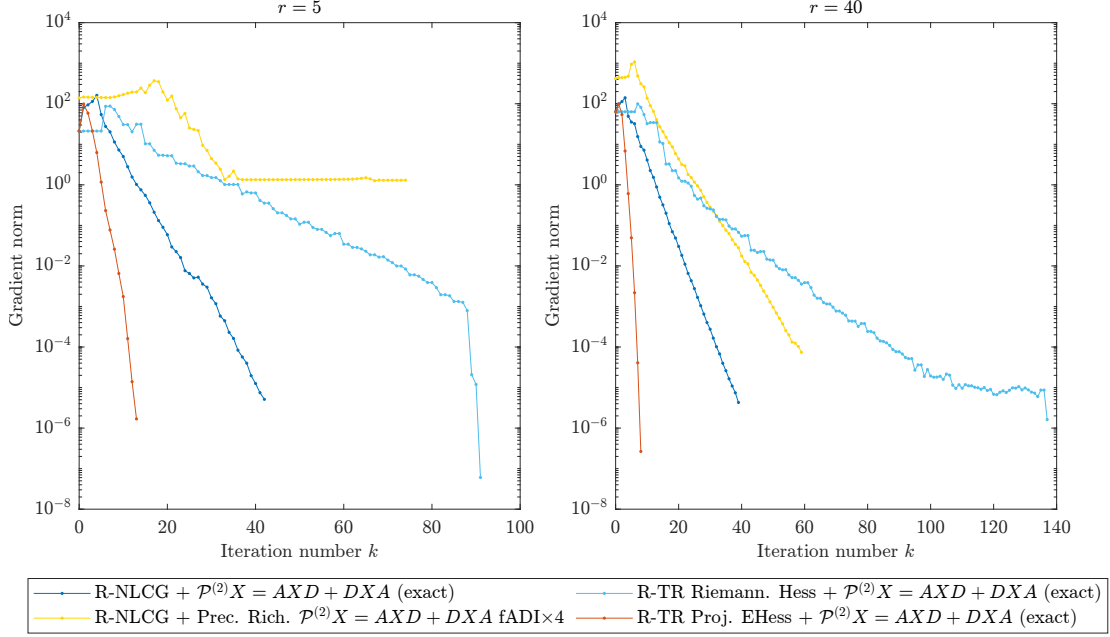


Figure 7.4: FD of 2D PDEs with separable variables, with $\alpha = 10$ and $n = 10000$. Comparison of fixed-rank Riemannian optimization methods on the manifold of matrices with ranks $r = 5$ (left) and $r = 40$ (right). The plots depict the Riemannian gradient norm versus iterations. For R-NLCG with Riemannian truncated preconditioned Richardson search direction, the Riemannian gradient and norm are computed in the Frobenius metric, while for all other methods they are calculated in the metric induced by $\mathcal{P}X = DXD$.

Parameterized PDEs of the type (7.7) emerge, for example, from the reparametrization of stochastic elliptic PDEs where random fields are approximated by their truncated Karhunen-Loève (KL) expansion. Refer to [LPS14, § 9] and [PSS17; TS07] for more details. The well-posedness of the weak formulation of (7.7) on $L^2(\Gamma, H_0^1(\Omega))$ follows from the Lax-Milgram lemma if there exist constants $a^{\min}, a^{\max} \in \mathbb{R}$ such that $0 < a^{\min} < a(x, y) < a^{\max}$ almost everywhere in $\Omega \times \Gamma$. To ensure this condition we consider the case

$$0 < \sum_{k=1}^q \|a_k\|_{\infty} < a_0. \quad (7.9)$$

Stochastic Galerkin methods [BTZ04; DBO01; LPS14] seek an approximation to the weak solution of (7.8) in finite-dimensional subspaces of $L^2(\Gamma, H_0^1(\Omega))$ constructed by tensorizing finite-dimensional subspaces of $H_0^1(\Omega)$ and $L^2(\Gamma)$. Given $V^h = \text{Span}\{\varphi_1(x), \dots, \varphi_m(x)\} \subset H_0^1(\Omega)$ and $S^p = \text{Span}\{\psi_1(y), \dots, \psi_n(y)\} \subset L^2(\Gamma)$, a finite-dimensional subspace of $L^2(\Gamma, H_0^1(\Omega))$ is constructed as

$$V^{hp} = V^h \otimes S^p := \text{Span}\{\varphi_i(x)\psi_j(y) : i = 1, \dots, m, j = 1, \dots, n\}. \quad (7.10)$$

7.2. Stochastic Galerkin matrix equations

The weak formulation of on V^{hp} yields the multiterm matrix equation

$$K_0 X G_0^\top + \sum_{k=1}^q K_k X G_k^\top = \mathbf{f}_0 \mathbf{g}_0^\top + \sum_{k=1}^q \mathbf{f}_k \mathbf{g}_k^\top, \quad (7.11)$$

where for $k = 1, \dots, q$

$$\begin{aligned} [K_0]_{s,t} &= \int_{\Omega} a_0 \nabla \varphi_s \cdot \nabla \varphi_t & [K_k]_{s,t} &= \int_{\Omega} a_k \nabla \varphi_s \cdot \nabla \varphi_t, \\ [G_0]_{s,t} &= \langle \psi_s, \psi_t \rangle & [G_k]_{s,t} &= \langle y_k \psi_s, \psi_t \rangle, \\ [\mathbf{f}_0]_s &= \int_{\Omega} f_0 \varphi_s & [\mathbf{f}_k]_s &= \int_{\Omega} f_k \varphi_s, \\ [\mathbf{g}_0]_s &= \langle \psi_s, 1 \rangle = [G_0]_{s,1} & [\mathbf{g}_k]_s &= \langle \psi_s, y_k \rangle = [G_k]_{s,1}. \end{aligned} \quad (7.12)$$

In our experiments, we choose V^h to be a finite-element space, with $\varphi_1(x), \dots, \varphi_m(x)$ being the associated FE basis functions, and S^p to be the set of multivariate polynomials of (total) degree at most k . Following [PSS17], an orthonormal basis for S^p is constructed via products of univariate Legendre polynomials.

Due to the choice of an orthonormal basis for S^p , we have $G_0 = I$. It can be proven that $G_1, \dots, G_q$ have at most two non-zero entries per row and their eigenvalues are contained in the interval $[-1, 1]$ [LPS14; EU10]. Moreover, the matrices G have an arrow-like sparsity pattern, shown in Figure 7.5, making them suitable for Cholesky decomposition with reordering. Regarding the stiffness matrices K_k , conditions (7.9) guarantee that $K_0 \succ 0$, while the matrices $K_1, \dots, K_q$ are generally indefinite since the functions $a_1, \dots, a_q$ might not be strictly positive. Nonetheless, conditions (7.9) imply that the operator $\mathcal{A}X = K_0 X G_0^\top + \sum_{i=1}^q K_i X G_i^\top$ is positive definite.

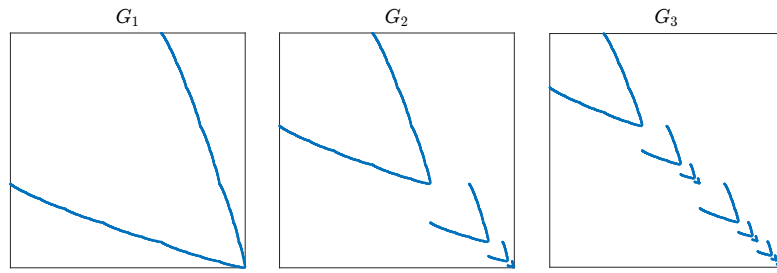


Figure 7.5: Sparsity pattern of the matrices G_k in (7.11).

Preconditioning The most straightforward preconditioner for (7.11) arises from approximating the diffusion coefficient $a(\cdot, \cdot)$ with its mean $a(x, y) \approx a_0$, resulting in

$$\mathcal{P}^{(1)} X = K_0 X G_0^\top = K_0 X. \quad (7.13)$$

The effectiveness of this preconditioner diminishes with an increasing ratio σ/μ , where $\mu = a_0$ is the expected value of the diffusion coefficient, and σ is the standard deviation

[PE09, Theorem 3.8].

An alternative preconditioner, accounting for parameter samples, is obtained in [KS11a; KT11] by approximating $a(x, y) \approx a_0 + \sum_{k=1}^q \bar{a}_k y_k$ where $\bar{a}_j = \int_{\Omega} a_j(x)$. This results in

$$\mathcal{P}^{(2)}X = K_0 X G^{\top}, \quad G = G_0 + \sum_{k=1}^q \frac{\bar{a}_k}{a_0} G_k. \quad (7.14)$$

In our experiments, this preconditioner performed nearly identically to the one proposed in [Ull10], i.e., $\mathcal{P}X = K_0 X G^{\top}$ where $G = G_0 + \sum_{k=1}^{\ell} \frac{\langle K_0, K_k \rangle}{\|K_0\|_{\mathbb{F}}^2} G_k$ is the matrix that minimizes $\|\sum_{i=0}^{\ell} G_i \otimes K_i - G \otimes K_0\|_{\mathbb{F}}$. We opted for $\mathcal{P}^{(2)}$ without a specific reason. Note that both $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ adhere to the form discussed in Section 5.1.

Whenever we invert the preconditioner $\mathcal{P}X = K_0 X G$, where $G = I_n$ if $\mathcal{P} = \mathcal{P}^{(1)}$, we need to invert the matrices K_0 and G . Therefore, it is convenient to perform a Cholesky factorization with reordering of K_0 and G as a one-time precomputation. To compute the reordering we used MATLAB's `dissect`, which employs nested dissection [KK98; Dav06]. The time spent on these precomputations is included in the overall timing of the algorithms but is negligible.

7.2.1 Numerical results

We examine Example 5.1 and Example 5.2 from [PSS17], corresponding to the test problems 5 and 2 of S-IFISS [BPS17; BPS14]. In both examples, the spatial domain Ω is a square, and we choose V^h to be the set of piecewise bilinear functions defined on a uniform finite element grid with $m = 16129$ degrees of freedom, corresponding to grid-level 7 of S-IFISS. We choose S^p to be the set of multivariate polynomials of total degree at most $p = 5$, with Legendre polynomials in $y_1, \dots, y_q$ serving as an orthonormal basis.

For both problems, we establish a target tolerance `tol` on the relative residual and define stopping criteria and parameters for CG with truncation and Riemannian optimization following the setup in Section 7.1.1. Additionally, we include in the comparison the MultiRB solver [PSS17, Algorithm 4.1], utilizing the publicly available implementation provided by the authors.

Test problem 5 of S-IFISS The problem is formulated on the square spatial domain $\Omega = [0, 1] \times [0, 1]$, with a deterministic and constant source function $f(x, y) = f_0(x) = 1$. The diffusion coefficient follows the form (7.8) with $a_0 = 1$, $a_k(x) = \gamma_k \cos(2\pi\rho_1(k)x_1) \cos(2\pi\rho_2(k)x_2)$, where $\gamma_k := 0.832 \cdot k^{-4}$ and $\rho_1(k) := k - s(k)(s(k) + 1)/2$, $\rho_2(k) := s(k) - \rho_1(k)$ with $s(k) := \lfloor -1/2 + \sqrt{1/4 + 2k} \rfloor$. This problem is extensively discussed in [Eig+14, p. 22]. Notably, the coefficients γ_k decay rapidly to zero, indicating a significantly higher weight for the initial terms. We consider $q = 9$, retaining all terms with $\gamma_k \geq 10^{-4}$, resulting in a multiterm matrix equation with $\ell = q + 1 = 10$ terms, and $m = 16129$, $n = 2002$.

We set the relative tolerance target for this problem to `tol` = 10^{-6} . Given the rapid

7.2. Stochastic Galerkin matrix equations

decay of coefficients γ_k to zero, the most efficient preconditioner proved to be the one-sided one $\mathcal{P}^{(1)}X = K_0X$. For fixed-rank Riemannian optimization, we choose the rank $r = 55$. In the case of RRAMs, the initial rank is set to $r_0 = 5$ and the rank increment to $r_{\text{up}} = 10$. Results are shown in Figure 7.6.

Among fixed-rank methods, R-TR with Projected Euclidean Hessian again exhibits the fastest convergence in terms of iterations. However, R-NLCG also converges within a relatively low number of iterations, and in terms of overall time it outperforms R-TR. This can be attributed to the fact that the inner iterations of tCG involve more Hessian-vector products than the total number of iterations required by R-NLCG to converge. Notably, when using a preconditioner of the form $\mathcal{P}^{(1)}X = K_0X$, there is almost no computational gain in applying the inverse of the Riemannian preconditioner for the same iterate X and multiple right-hand sides $\eta \in \mathcal{T}_X\mathcal{M}_r$.

Turning to rank-adaptive methods, CG with truncation demonstrates the most consistent convergence rate without excessive growth of intermediate ranks. Despite this favorable behavior, its convergence in time is slower compared to R-NLCG, whether in its fixed-rank version or the rank-adaptive variant. Note how the convergence rate of CG with truncation and fixed-rank R-NLCG are very similar. The MultiRB method proves to be non-competitive in both iterations and time, exhibiting excessive rank growth of the approximate solution.

RRAM with R-NLCG emerges as the fastest method in time, primarily due to its initial low-rank iterations that significantly reduce the residual at a minimal computational cost. However, the number of iterations is significantly higher than in CG with truncation, essentially due to numerous iterations where the relative residual stagnates while fixed-rank optimization is still performed. Removing the plateaus, the convergence rate of RRAM with R-NLCG seems very similar to that of CG with truncation. Moreover, it is interesting to note that each rank update maintains the residual nearly constant, without fluctuations, and in the initial fixed-rank optimization steps a substantial decrease in residual is promptly noticeable. On the contrary, RRAM with R-TR exhibits the slowest convergence in time, mainly attributed to the worse performance of fixed-rank RTR compared to fixed-rank R-NLCG.

Considerations regarding potential method enhancements are similar to those mentioned earlier. Specifically, both CG with truncation and the rank increase steps in RRAMs could benefit significantly from the application of randomized techniques for TSVD calculation. More importantly, the introduction of residual estimation techniques [Mey+21; PCK22; GT18; BK21] could effectively prevent stagnation in RRAMs.

Test problem 2 of S-IFISS The problem is defined on the square spatial domain $\Omega = [-1, 1] \times [-1, 1]$, with a deterministic source function $f(x, y) = (2 - x_1^2 - x_2^2)/8$. The diffusion coefficient takes the form of a truncated Karhunen-Loève expansion

$$a(x, y) = a_0 + \sigma\sqrt{3} \sum_{k=1}^q \sqrt{\lambda_k} \phi_k(x) y_k \quad (7.15)$$

7.2. Stochastic Galerkin matrix equations

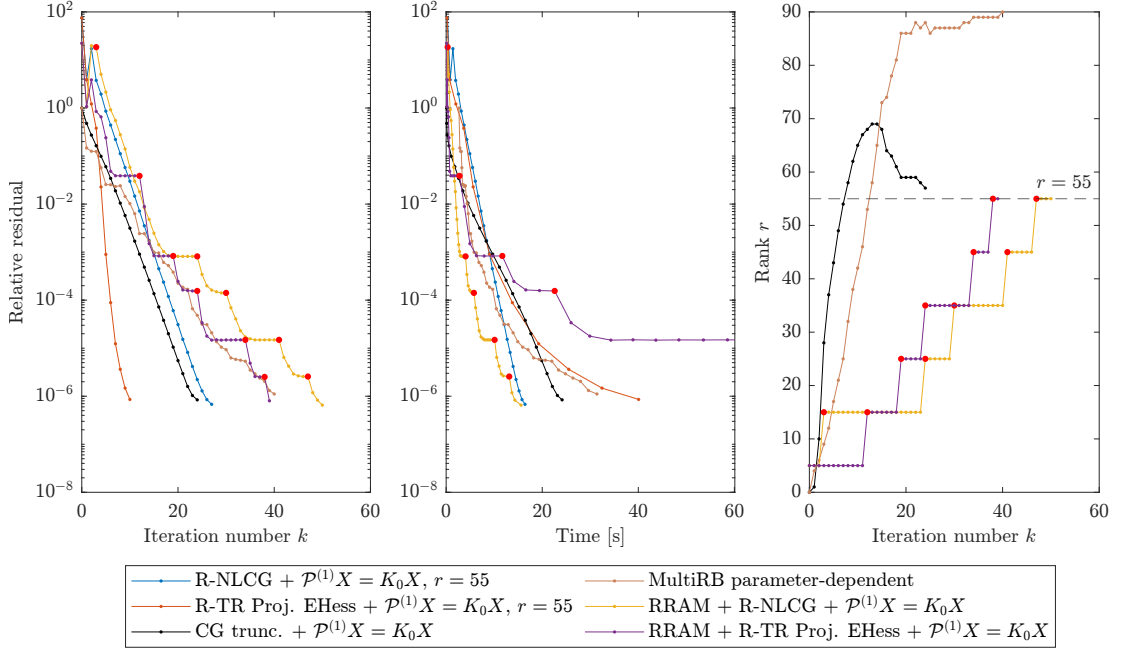


Figure 7.6: Stochastic Galerkin matrix equations, test problem 5 from *S-IFISS* ($m = 16129$, $n = 2002$). Comparison of fixed-rank Riemannian optimization methods on the matrices of rank $r = 55$, CG with truncation, MultiRB, and RRAMs. Red dots in RRAMs' plots indicate rank increases. From left to right: relative residual vs. iterations, relative residual vs. time, and rank of approximate solution vs. iterations.

where $a_0 = 1$ and $\{(\lambda_k, \phi_k)\}_{k=1}^{\infty}$ are the eigenpairs of the integral operator associated with the covariance function

$$C[a](x, \bar{x}) = \sigma^2 \exp\left(-\frac{\|x - \bar{x}\|_1}{l}\right),$$

which can be computed analytically [LPS14, § 7]. This problem is extensively discussed in [BPS14]. We set the correlation length to $l = 2$, the standard deviation to $\sigma = 0.3$, and retain the first $q = 8$ terms of the KL expansion. This results in a multiterm matrix equation with $\ell = 9$ terms, and $m = 16129$, $n = 1287$.

It is worth noting that the value $\sigma = 0.3$ corresponds to a relatively high standard deviation of the truncated diffusion coefficient, making the problem more challenging and requiring a much larger rank to adequately approximate the solution. Moreover, the eigenvalues λ_k , which determine the weighting of the terms ϕ_k in (7.15), have much slower decay to zero than the γ_k 's in the previous problem. Consequently, in this example, the most efficient preconditioner proved to be the two-sided one $\mathcal{P}^{(2)}X = K_0XG$.

We set the relative tolerance target for this problem to $\text{tol} = 10^{-5}$. For fixed-rank Riemannian optimization, we choose the rank $r = 180$. For RRAMs, the initial rank is set to $r_0 = 30$ and the rank increment to $r_{\text{up}} = 30$. Results are shown in Figure 7.7.

The behavior of fixed-rank Riemannian methods mirrors the patterns observed in the previous example, where R-TR exhibits fewer outer iterations compared to R-NLCG,

7.2. Stochastic Galerkin matrix equations

although the latter proves to be more efficient in time. The convergence rate of CG with truncation aligns with that of fixed-rank R-NLCG, but as the rank of CG iterates increases, CG iterations become slower. Note also how, since the rank is higher than in the previous example, the benefits of fixed-rank Riemannian optimization over CG with truncation are less evident. RRAM with R-NLCG demonstrates competitiveness with CG in time, but the number of iterations is considerably higher, primarily due to numerous iterations where the residual stagnates. MultiRB proves to be the most time-efficient method, especially up to a relative residual around 10^{-4} ; however, the rank of the approximate solution grows considerably.

Notably, the final rank achieved by RRAMs is significantly lower than that of MultiRB and slightly lower than that of CG. Consequently, the Riemannian rank-adaptive method with R-NLCG emerges as a competitive alternative in terms of time and final rank of the approximate solution. However, in terms of iterations, CG with truncation proves to be better. We emphasize once again that a nonrandom initialization and the application of techniques for mitigating plateaus through residual norm estimation would further improve the convergence speed of RRAMs, bringing them closer to CG in terms of iterations.

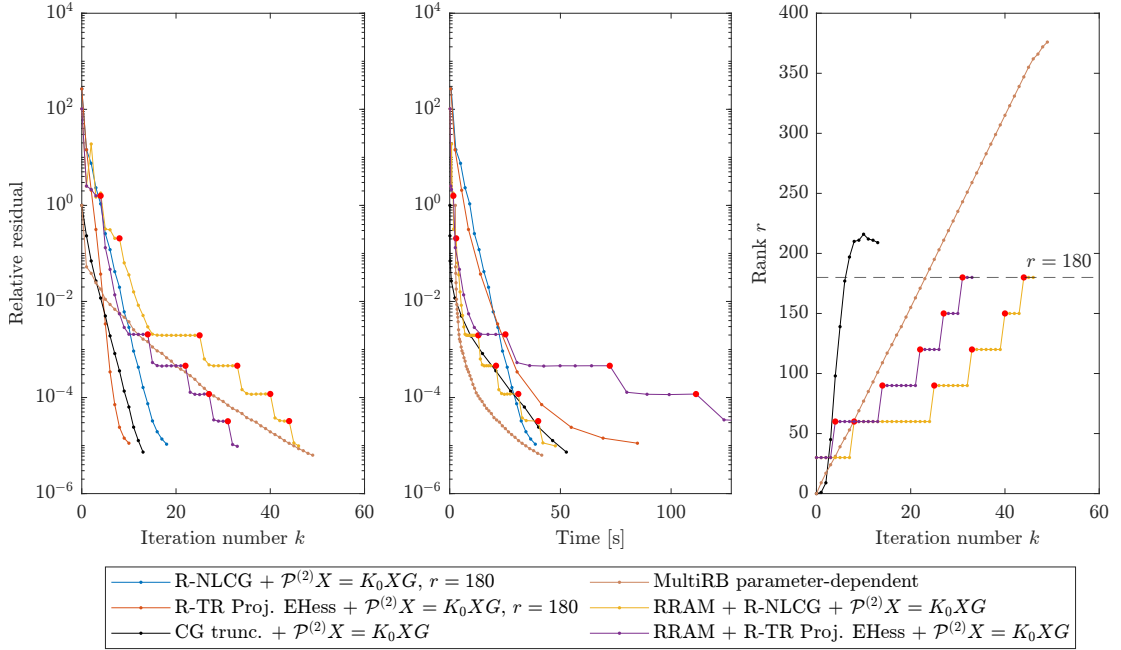


Figure 7.7: Stochastic Galerkin matrix equations, test problem 2 from S-IFISS ($m = 16129$, $n = 1287$). Comparison of fixed-rank Riemannian optimization methods on the matrices of rank $r = 180$, CG with truncation, MultiRB, and RRAMs. Red dots in RRAMs' plots indicate rank increases. From left to right: relative residual vs. iterations, relative residual vs. time, and rank of approximate solution vs. iterations.

7.3 Modified Bilinear Rail problem

We consider the bilinear reformulation of the Rail (or Steel Profile) example from the Oberwolfach collection [BS05]. This bilinear control system arises from the optimization problem of cooling steel profiles and takes the form

$$M\dot{x}(t) = (-A)x(t) + \sum_{i=1}^{\ell} N_i x(t) u_i(t) + Bu(t), \quad y(t) = Cx(t), \quad (7.16)$$

where $A, M \in \mathbb{R}^{n \times n}$ are symmetric positive definite matrices, $N_i \in \mathbb{R}^{n \times n}$ is symmetric for all $i = 1, \dots, \ell$, $B \in \mathbb{R}^{n \times \ell}$, and $C \in \mathbb{R}^{r_Y \times n}$. As discussed in [BB13; BD11], the reachability Gramian corresponding to (7.16) solves the multiterm Lyapunov equation (1.2), where $F = BB^\top$ is the right-hand side.

This example poses challenges due to the non-uniform FEM mesh, resulting in a mass matrix M with a relatively high condition number ($\kappa_2(M) \approx 350$ for the chosen refinement level). As outlined in [Van10, § 8.2.3], this renders the Lyapunov preconditioner $\mathcal{P}X = AX + XA$, obtained from the dominant term by approximating $M \approx I$, less effective. The matrices were obtained from the M-M.E.S.S. toolbox [SKB23], resulting in a multiterm linear matrix equation with size $n = 5177$ and $\ell = 6$ terms. We adjusted the matrices by modifying the constants related to the rail profile material as follows: ρ and γ_k were divided by 10^2 , λ and c were divided by 10, and u_{ext} was multiplied by 10^2 . This adjustment was necessary because otherwise the terms in N_i would be practically negligible, making the solution of the multiterm Lyapunov equation almost coincide with solving the generalized Lyapunov equation $AXM^\top + MXA^\top = BB^\top$. We verified numerically that the assumptions of [Dam08, Lemma 5.1] are satisfied, guaranteeing positive definiteness of the operator $\mathcal{A}$ and the existence of a symmetric positive semidefinite solution for any symmetric positive semidefinite right-hand-side. Furthermore, for the right-hand side of the multiterm Lyapunov equation, we considered $\tilde{B}\tilde{B}^\top$ where $\tilde{B}$ contains the first and last columns of the matrix B from the M-M.E.S.S. toolbox. Although these modifications result in the loss of the original physical meaning of the equation, they yield a challenging multiterm Lyapunov matrix equation with a nontrivial mass matrix M .

Preconditioning A preconditioner is derived from the dominant generalized Lyapunov term $\mathcal{P}^{(2)}X = AXM + MXA$. For Riemannian optimization, we compare it with the Lyapunov preconditioner $\mathcal{P}^{(1)}X = AX + XA$ from [VV10], obtained by approximating $M \approx I$.

7.3.1 Numerical results

We set a target tolerance $\text{tol} = 10^{-6}$ on the relative residual. In CG with truncation, the following values of truncation tolerances have shown satisfactory performance. For truncation of the iterates X_k , we employ a relative truncation tolerance

$\epsilon_{\text{rel}} = 10^{-4} \cdot \text{tol} = 10^{-10}$. Regarding the truncation of R_k and P_k , we set $\epsilon_{\text{rel}} = 10^{-2} \cdot \text{tol}$ and $\epsilon_{\text{abs}} = 10^{-4} \cdot \text{tol}$, using the same notation as in (1.12). Finally, we do not truncate Q_k . In Riemannian optimization, we define stopping criteria and parameters following the setup in Section 7.1.1. As an initialization strategy, we perform enough fADI steps for the generalized Lyapunov equation $AXM + MXA = \tilde{B}\tilde{B}^\top$ until we reach the desired rank. For fixed-rank Riemannian optimization, we choose the rank $r = 150$. In the case of RRAMs, the initial rank is set to $r_0 = 25$, and the rank increment to $r_{\text{up}} = 25$. Results are shown in Figure 7.8.

Note that the computational cost of applying exactly the inverse of the Riemannian preconditioned derived from $\mathcal{P}^{(2)}X = AXM + MXA$ is extremely high. To achieve convergence within a reasonable timeframe, it is more practical to employ a few steps of tangADI. Specifically, applying the exact preconditioner from Section 6.2.3 involves solving $4r + 1$ linear systems with each of the r matrices $(A + \lambda_i M)$, totaling $4r^2 + r$ linear systems of size n , plus the final $r^2 \times r^2$ system. In contrast, tangADI requires solving only r linear systems with matrix $(A + p_i M)$ per tangADI step. Given the relatively high rank $r = 150$, one iteration of R-NLCG with a few steps of tangADI is considerably more efficient than using the exact preconditioner's inverse. Using 8 steps of tangADI shows satisfactory performance, with the use of such an approximate inverse of the Riemannian preconditioner not significantly impacting the iteration count compared to the exact one, resulting in substantially lower convergence times for R-NLCG with tangADI. More precisely, a single iteration with the exact application of the preconditioner's inverse is approximately 4 times slower than the entire convergence time with tangADI.

Moreover, adopting a one-sided Lyapunov preconditioner $\mathcal{P}^{(1)}X = AX + XA$ leads to a higher iteration count and longer timings, consistent with observations in [VV10]. This is because $\kappa_2(M) \approx 350$, and consequently, the approximation $M \approx I$ is too crude to provide an effective preconditioner. The use of R-TR with projected Euclidean Hessian and tangADI preconditioner does not provide benefits in terms of outer iterations, and as a result the total execution time is higher than R-NLCG due to the Hessian-vector products in the inner tCG iterations. Note that also for tangADI there is no computational gain in applying the preconditioner for the same iterate X and multiple right-hand sides $\eta \in \text{TS}_{n,r}^+$.

Preconditioned CG with truncation converges in only 3 iterations without excessive rank growth, and is therefore the best method in terms of iterations. Despite this excellent performance of CG with truncation, fixed-rank R-NLCG with tangADI shows comparable performance in time, albeit requiring a higher number of iterations. The RRAM shows a slightly slower convergence time, but attains a lower final rank than CG with truncation. It is remarkable that even in this case, where CG with truncation shows excellent performance, Riemannian methods show a comparable performance in time, both in their fixed-rank and in their rank-adaptive variants. However, the number of iterations of the RRAM is about 10 times higher than that of CG with truncation, mainly due to numerous plateaus.

7.3. Modified Bilinear Rail problem

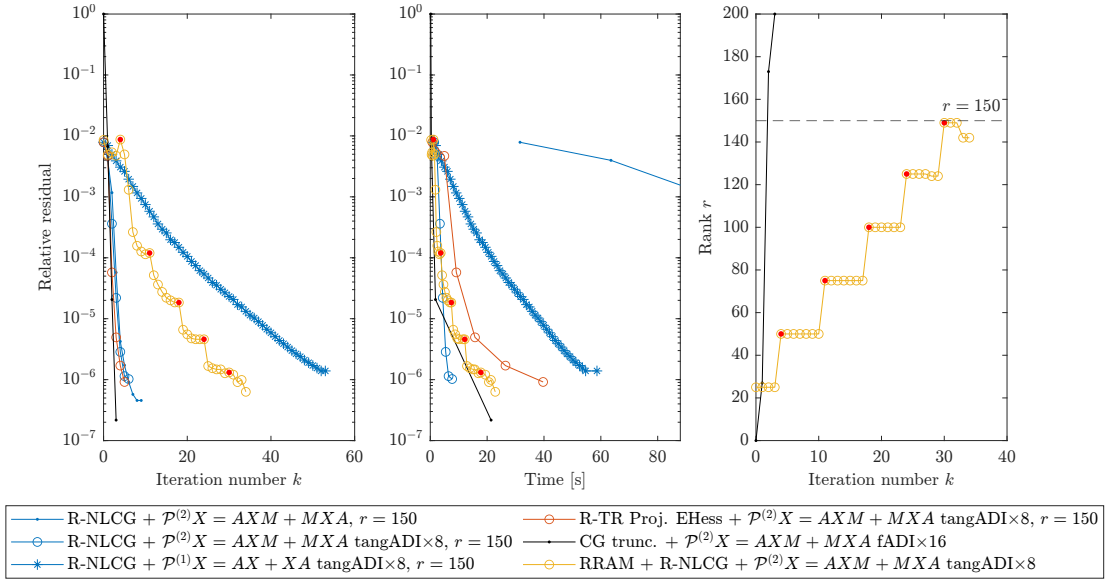


Figure 7.8: Modified Bilinear Rail problem ($n = 5177$). Comparison of fixed-rank Riemannian optimization methods on symmetric positive semidefinite matrices of rank $r = 150$, CG with truncation and RRAM. Red dots in the RRAM's plots indicate rank increases. From left to right: relative residual vs. iterations, relative residual vs. time, and rank of approximate solution vs. iterations.

Chapter 8

Conclusions

This concluding chapter summarizes the main findings of the thesis, offering a comprehensive perspective on the results and presenting insights for potential enhancements and future developments.

The central focus of this thesis was the resolution of multiterm linear matrix equations through preconditioned Riemannian optimization on low-rank matrix manifolds. Building upon earlier works, notably [Van10; KSV16], we underscored the pivotal role of preconditioning in the low-rank Riemannian approach. However, the challenge laid in identifying preconditioners that not only exhibited commendable performance, but also aligned with the low-rank structure of the manifold. This research contributed by developing effective preconditioners for multiterm linear matrix equations, providing insights on the convergence guarantees of the associated preconditioned Riemannian optimization algorithms. Specifically, we considered preconditioners derived from dominant terms in the operator $\mathcal{A}$, of the forms $\mathcal{P}X = EXD$ or $\mathcal{P}X = AXD + EXB$, where $A, B, D, E \succ 0$.

In Section 3.4, we interpreted Riemannian preconditioners as metric changes, allowing to extend the convergence guarantees of the corresponding Riemannian algorithms to the preconditioned case. Furthermore, we clarified the limitations of the Riemannian truncated preconditioned Richardson iteration in our setting, emphasizing its possible failure due to a "wrong projection" of the preconditioned Euclidean gradient, particularly on matrix manifolds with very low rank. This insight was corroborated by numerical experiments.

Motivated by the goal of utilizing the entire operator $\mathcal{A}$ for preconditioning, rather than just a dominant term $\mathcal{P}$, we proposed the use of the projected Euclidean Hessian as a model Hessian for R-TR. This choice exhibited advantages over the Riemannian Hessian, including (1) the positive definiteness of the resulting model Hessian, facilitating the resolution of the trust-region subproblem via tCG; and (2) eliminating issues related to inverting the singular values' matrix for numerically rank-deficient iterates. Numerical experiments showcased a superior performance of this approach over R-TR with the Riemannian Hessian, the method employed in [VV10] for an efficient solver for generalized Lyapunov equations. Importantly, we established in Proposition 4.3 that,

for the chosen cost function, using the projected Euclidean Hessian as a model Hessian preserves local quadratic convergence to the exact solution of the multiterm linear matrix equation, if it is on the manifold.

A contribution of this work is the development of computationally efficient methods for applying Riemannian preconditioners derived from two-sided operators $\mathcal{P}X = EXD$ and $\mathcal{P}X = AXD + EXB$. Numerical experiments demonstrated that these two-sided preconditioners can significantly accelerate convergence compared to their one-sided counterparts. Furthermore, for the computationally demanding generalized Sylvester preconditioner, a viable alternative is presented through a few iterations of the tangADI algorithm. The advantages of this approximate preconditioner are especially evident when the rank r is not negligible compared to the size of the coefficient matrices.

Lastly, by alternating between fixed-rank Riemannian optimization and rank updates, we devised a rank-adaptive method that proved competitive with established low-rank solvers for multiterm linear matrix equations, such as CG with truncation and MultiRB. Significant computational advantages of the RRAM become apparent when CG with truncation exhibits excessive growth of intermediate ranks. Additionally, in our experiments, RRAM displayed the capability to find approximate solutions of lower final rank compared to the other algorithms.

8.1 Future outlooks

One of the primary limitations of this study lies in its strong reliance on the positive definiteness assumption on $\mathcal{A}$, shared with the related works [VV10; KSV16]. However, interesting applications of multiterm linear matrix equations exist where this assumption is not met; see, e.g., [KS15; BB13; KT11; PK21]. Extending the Riemannian approach to the case where $\mathcal{A}$ is invertible but nonsymmetric or indefinite is an interesting avenue for future research. One possibility is to resort to normal equations, equivalent to minimizing the Frobenius norm of the residual. However, this approach encounters two main obstacles. Firstly, both the number of terms in the equation and the rank of the right-hand side increase by a factor ℓ , where ℓ is the number of terms in $\mathcal{A}$, leading to a proportional rise in the computational complexity of computing the cost, the gradient, and the Hessian. Secondly, and more significantly, the condition number gets squared, accentuating the need for effective preconditioners. If $\mathcal{A}$ has a generalized-Sylvester-like dominant term $\mathcal{P}$, the term $\mathcal{P}^\top \mathcal{P}$, usually a good preconditioner for $\mathcal{A}^\top \mathcal{A}$, loses its Sylvester structure due to the presence of cross-terms, needing new dedicated techniques to invert the corresponding Riemannian preconditioner $\text{Proj}_X \circ (\mathcal{P}^\top \mathcal{P}) \circ \text{Proj}_X$. Hence, applying a preconditioned low-rank Riemannian approach to nonsymmetric or indefinite cases requires further exploration and study.

Additionally, it is important to note that our work assumed the low-rank approximability of the solution, as a basis of the low-rank Riemannian approach. Although beyond the scope of this study, providing theoretical results in this regard, possibly with less restrictive hypotheses on the coefficient matrices than [BD11], would be a valuable

direction for future investigations.

For Riemannian preconditioners arising from operators like $\mathcal{P}X = EXD$ or $\mathcal{P}X = AXD + EXB$, one of the main computational costs, especially when r is not negligible compared to n and m , arises from solving linear systems with matrices E, D or $(A + \lambda E), (B + \mu D)$. Efficiently solving these linear systems is crucial to the preconditioner’s performance. The use of preconditioners as approximate inverses for these matrices could enhance the efficiency of the overall method, especially considering that the shifts are often close to each other. This aligns with observations in [Van10], and further exploration into this aspect could contribute to computational efficiency. Moreover, the convergence of tangADI when multiple shift parameters are employed remains an open question.

In this work, we mainly focused on fixed-rank Riemannian optimization, with a rank-adaptive method incorporated for fair comparison with algorithms lacking knowledge of the solution’s rank in advance. Given the satisfactory performance of RRAMs, future efforts could be directed towards improving their computational efficiency and gaining a better understanding of their convergence properties. Techniques akin to those used in [Zho+16; SU15] could be explored for convergence proofs. To gain in computational efficiency, randomized linear algebra techniques offer a promising avenue. Randomized TSVD [YGL18; HMT11] could be employed in rank-update steps. It is worth mentioning, for fair comparison, that CG with truncation would also benefit substantially from randomized TSVD in the truncation steps. Additionally, efficiently estimating the residual norm [PCK22; BK21] would allow immediate progression to the subsequent rank once stagnation is detected, mitigating the often-observed plateaus and reducing the number of iterations.

Bibliography

- [Abs+09] P.-A. Absil et al. “All roads lead to Newton: Feasible second-order methods for equality-constrained optimization”. In: (Oct. 1, 2009).
- [ABG07] P.-A. Absil, C.G. Baker, and K.A. Gallivan. “Trust-Region Methods on Riemannian Manifolds”. In: *Foundations of Computational Mathematics* 7.3 (July 1, 2007), pp. 303–330.
- [AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*: Princeton University Press, Dec. 31, 2008.
- [AM12] P.-A. Absil and Jérôme Malick. “Projection-like Retractions on Matrix Manifolds”. In: *SIAM Journal on Optimization* 22.1 (Jan. 2012), pp. 135–158.
- [ASZ02] A. C. Antoulas, D. C. Sorensen, and Y. Zhou. “On the decay rate of Hankel singular values and related issues”. In: *Systems & Control Letters* 46.5 (Aug. 15, 2002), pp. 323–342.
- [BTZ04] Ivo Babuska, Raúl Tempone, and Georgios E. Zouraris. “Galerkin Finite Element Approximations of Stochastic Elliptic Partial Differential Equations”. In: *SIAM Journal on Numerical Analysis* 42.2 (Jan. 2004), pp. 800–825.
- [BES15] Jonathan Baker, Mark Embree, and John Sabino. *Fast singular value decay for Lyapunov solutions with nonnormal coefficients*. Jan. 29, 2015. arXiv: 1410.8741 [math].
- [BG13] Jonas Ballani and Lars Grasedyck. “A projection method to solve linear systems in tensor format”. In: *Numerical Linear Algebra with Applications* 20.1 (2013), pp. 27–43.
- [BT17] Bernhard Beckermann and Alex Townsend. “On the Singular Values of Matrices with Displacement Structure”. In: *SIAM Journal on Matrix Analysis and Applications* 38.4 (Jan. 2017), pp. 1227–1248.
- [BT19] Bernhard Beckermann and Alex Townsend. “Bounds on the Singular Values of Matrices with Displacement Structure”. In: *SIAM Review* 61.2 (Jan. 2019), pp. 319–344.
- [BB13] Peter Benner and Tobias Breiten. “Low rank methods for a class of generalized Lyapunov equations and related issues”. In: *Numerische Mathematik* 124.3 (July 1, 2013), pp. 441–470.

-
- [BD11] Peter Benner and Tobias Damm. “Lyapunov Equations, Energy Functionals, and Model Order Reduction of Bilinear and Stochastic Systems”. In: *SIAM Journal on Control and Optimization* 49.2 (Jan. 2011), pp. 686–711.
 - [BLT09] Peter Benner, Ren-Cang Li, and Ninoslav Truhar. “On the ADI method for Sylvester equations”. In: *Journal of Computational and Applied Mathematics* 233.4 (Dec. 15, 2009), pp. 1035–1045.
 - [BMS07] Peter Benner, Hermann Mena, and Jens Saak. “On the Parameter Selection Problem in the Newton-ADI Iteration for Large Scale Riccati Equations”. In: *1864-0087* 29 (Nov. 26, 2007).
 - [BS05] Peter Benner and Jens Saak. “A Semi-Discretized Heat Transfer Model for Optimal Cooling of Steel Profiles”. In: *Dimension Reduction of Large-Scale Systems*. Ed. by Peter Benner, Danny C. Sorensen, and Volker Mehrmann. Lecture Notes in Computational Science and Engineering. Springer, 2005, pp. 353–356.
 - [BS13] Peter Benner and Jens Saak. “Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey”. In: *GAMM-Mitteilungen* 36.1 (2013), pp. 32–52.
 - [BPS14] Alex Bespalov, Catherine E. Powell, and David Silvester. “Energy Norm A Posteriori Error Estimation for Parametric Operator Equations”. In: *SIAM Journal on Scientific Computing* 36.2 (Jan. 2014), A339–A363.
 - [BPS17] Alex Bespalov, Catherine E. Powell, and David Silvester. *Stochastic IFISS (S-IFISS)*. Version 1.04. 2017.
 - [Bou23a] Nicolas Boumal. *An Introduction to Optimization on Smooth Manifolds*. 1st ed. Cambridge University Press, Mar. 16, 2023.
 - [Bou23b] Nicolas Boumal. *Retractions locally preserve distance*. Race to the bottom — the OPTIM@EPFL blog. Nov. 20, 2023. URL: <https://www.racetothetbottom.xyz/posts/retraction-distance/>.
 - [BA15] Nicolas Boumal and P. -A. Absil. “Low-rank matrix completion via preconditioned optimization on the Grassmann manifold”. In: *Linear Algebra and its Applications* 475 (June 15, 2015), pp. 200–239.
 - [Bou+13] Nicolas Boumal et al. *Manopt, a Matlab toolbox for optimization on manifolds*. Aug. 23, 2013. arXiv: 1308.5200[cs,math,stat].
 - [BR19] Tobias Breiten and Emil Ringh. “Residual-based iterations for the generalized Lyapunov equation”. In: *BIT Numerical Mathematics* 59.4 (Dec. 1, 2019), pp. 823–852.
 - [BK21] Zvonimir Bujanovic and Daniel Kressner. “Norm and Trace Estimation with Random Rank-one Vectors”. In: *SIAM Journal on Matrix Analysis and Applications* 42.1 (Jan. 2021), pp. 202–223.

-
- [BM03] Samuel Burer and Renato D.C. Monteiro. “A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization”. In: *Mathematical Programming* 95.2 (Feb. 1, 2003), pp. 329–357.
 - [CGT00] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. Society for Industrial and Applied Mathematics, Jan. 2000.
 - [DKM10] Wei Dai, Ely Kerman, and Olgica Milenkovic. *A Geometric Approach to Low-Rank Matrix Completion*. June 10, 2010. arXiv: 1006.2086[cs,math].
 - [Dam08] T. Damm. “Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations”. In: *Numerical Linear Algebra with Applications* 15.9 (2008), pp. 853–871.
 - [Dam04] Tobias Damm. “Rational Matrix Equations in Stochastic Control”. In: *Lecture Notes in Control and Information Sciences* 297 (Jan. 1, 2004).
 - [Dav06] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, Jan. 2006. 228 pp.
 - [DBO01] Manas K. Deb, Ivo M. Babuška, and J. Tinsley Oden. “Solution of stochastic partial differential equations using Galerkin finite element techniques”. In: *Computer Methods in Applied Mechanics and Engineering* 190.48 (Sept. 28, 2001), pp. 6359–6372.
 - [DE99] Luca Dieci and Timo Eirola. “On Smooth Decompositions of Matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 20.3 (Jan. 1999), pp. 800–819.
 - [DKS11] V. Druskin, L. Knizhnerman, and V. Simoncini. “Analysis of the Rational Krylov Subspace and ADI Methods for Solving the Lyapunov Equation”. In: *SIAM Journal on Numerical Analysis* 49.5 (Jan. 2011), pp. 1875–1898.
 - [DS11] V. Druskin and V. Simoncini. “Adaptive rational Krylov subspaces for large-scale dynamical systems”. In: *Systems & Control Letters* 60.8 (Aug. 1, 2011), pp. 546–560.
 - [Eig+14] Martin Eigel et al. “Adaptive stochastic Galerkin FEM”. In: *Computer Methods in Applied Mechanics and Engineering* 270 (Mar. 1, 2014), pp. 247–269.
 - [EW91] Nancy S. Ellner and Eugene L. Wachspress. “Alternating Direction Implicit Iteration for Systems with Complex Spectra”. In: *SIAM Journal on Numerical Analysis* 28.3 (June 1991), pp. 859–870.
 - [EW86] Nancy S. Ellner and Eugene l. Wachspress. “New ADI model problem applications”. In: *Proceedings of 1986 ACM Fall joint computer conference*. ACM ’86. IEEE Computer Society Press, Nov. 2, 1986, pp. 528–534.
 - [EU10] Oliver G. Ernst and Elisabeth Ullmann. “Stochastic Galerkin Matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (Jan. 2010), pp. 1848–1872.

- [FL18] Florian Feppon and Pierre F. J. Lermusiaux. “A Geometric Approach to Dynamical Model-Order Reduction”. In: *SIAM Journal on Matrix Analysis and Applications* 39.1 (Jan. 2018), pp. 510–538. arXiv: 1705.08521[math].
- [GA22] Bin Gao and P.-A. Absil. “A Riemannian rank-adaptive method for low-rank matrix completion”. In: *Computational Optimization and Applications* 81.1 (Jan. 1, 2022), pp. 67–90.
- [GN92] Jean Charles Gilbert and Jorge Nocedal. “Global Convergence Properties of Conjugate Gradient Methods for Optimization”. In: *SIAM Journal on Optimization* 2.1 (Feb. 1992), pp. 21–42.
- [GP73] G. H. Golub and V. Pereyra. “The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate”. In: *SIAM Journal on Numerical Analysis* 10.2 (Apr. 1973), pp. 413–432.
- [GV13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2013.
- [GT18] Serge Gratton and David Tittley-Peloquin. “Improved Bounds for Small-Sample Estimation”. In: *SIAM Journal on Matrix Analysis and Applications* 39.2 (Jan. 2018), pp. 922–931.
- [HZ06] William Hager and Hongchao Zhang. “A survey of nonlinear conjugate gradient method”. In: 2 (Jan. 2006).
- [HH09] Justin P. Haldar and Diego Hernando. “Rank-Constrained Solutions to Linear Matrix Equations Using PowerFactorization”. In: *IEEE Signal Processing Letters* 16.7 (July 2009), pp. 584–587.
- [HMT11] N. Halko, P. G. Martinsson, and J. A. Tropp. “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions”. In: *SIAM Review* 53.2 (Jan. 2011), pp. 217–288.
- [HS21] Yue Hao and Valeria Simoncini. “The Sherman–Morrison–Woodbury formula for generalized linear matrix equations and applications”. In: *Numerical Linear Algebra with Applications* 28.5 (2021), e2384.
- [HS52] M.R. Hestenes and E. Stiefel. “Methods of conjugate gradients for solving linear systems”. In: *Journal of Research of the National Bureau of Standards* 49.6 (Dec. 1952), p. 409.
- [Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 2002. 690 pp.
- [Hoc13] Michiel E. Hochstenbach. “Probabilistic Upper Bounds for the Matrix Two-Norm”. In: *Journal of Scientific Computing* 57.3 (Dec. 1, 2013), pp. 464–476.
- [HJ12] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. 2nd ed. Cambridge University Press, Oct. 22, 2012.

-
- [HGA15] Wen Huang, K. A. Gallivan, and P.-A. Absil. “A Broyden Class of Quasi-Newton Methods for Riemannian Optimization”. In: *SIAM Journal on Optimization* 25.3 (Jan. 2015), pp. 1660–1685.
 - [HGZ17] Wen Huang, K. A. Gallivan, and Xiangxiong Zhang. “Solving PhaseLift by Low-Rank Riemannian Optimization Methods for Complex Semidefinite Constraints”. In: *SIAM Journal on Scientific Computing* 39.5 (Jan. 2017), B840–B859.
 - [JNS12] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. *Low-rank Matrix Completion using Alternating Minimization*. Dec. 3, 2012. arXiv: 1212.0467[cs,math,stat].
 - [Jar+18] Elias Jarlebring et al. “Krylov methods for low-rank commuting generalized Sylvester equations”. In: *Numerical Linear Algebra with Applications* 25.6 (Dec. 2018), e2176. arXiv: 1704.02167[math].
 - [Jou+10] M. Journée et al. “Low-Rank Optimization on the Cone of Positive Semidefinite Matrices”. In: *SIAM Journal on Optimization* 20.5 (Jan. 2010), pp. 2327–2351.
 - [KK98] George Karypis and Vipin Kumar. “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs”. In: *SIAM Journal on Scientific Computing* 20.1 (Jan. 1998), pp. 359–392.
 - [KM16] Hiroyuki Kasai and Bamdev Mishra. *Low-rank tensor completion: a Riemannian manifold preconditioning approach*. May 26, 2016. arXiv: 1605.08257[cs,math,stat].
 - [KS11a] Boris N. Khoromskij and Christoph Schwab. “Tensor-Structured Galerkin Approximation of Parametric and Stochastic Elliptic PDEs”. In: *SIAM Journal on Scientific Computing* 33.1 (Jan. 2011), pp. 364–385.
 - [Kle69] D. Kleinman. “On the stability of linear stochastic systems”. In: *IEEE Transactions on Automatic Control* 14.4 (Aug. 1969), pp. 429–430.
 - [KS11b] L. Knizhnerman and V. Simoncini. “Convergence analysis of the extended Krylov subspace method for the Lyapunov equation”. In: *Numerische Mathematik* 118.3 (July 1, 2011), pp. 567–586.
 - [KPS14] Daniel Kressner, Marija Miloloža Pandur, and Meiyue Shao. “An indefinite variant of LOBPCG for definite matrix pencils”. In: *Numerical Algorithms* 66.4 (Aug. 1, 2014), pp. 681–703.
 - [KPT14] Daniel Kressner, Martin Plešinger, and Christine Tobler. “A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations”. In: *Numerical Linear Algebra with Applications* 21.5 (2014), pp. 666–684.
 - [KS15] Daniel Kressner and Petar Sirković. “Truncated low-rank methods for solving general linear matrix equations”. In: *Numerical Linear Algebra with Applications* 22.3 (2015), pp. 564–583.

- [KSU14] Daniel Kressner, Michael Steinlechner, and André Uschmajew. “Low-Rank Tensor Methods with Subspace Correction for Symmetric Eigenvalue Problems”. In: *SIAM Journal on Scientific Computing* 36.5 (Jan. 2014), A2346–A2368.
- [KSV16] Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. “Preconditioned Low-rank Riemannian Optimization for Linear Systems with Tensor Product Structure”. In: *SIAM Journal on Scientific Computing* 38.4 (Jan. 2016), A2018–A2044.
- [KT10] Daniel Kressner and Christine Tobler. “Krylov Subspace Methods for Linear Systems with Tensor Product Structure”. In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (Jan. 2010), pp. 1688–1714.
- [KT11] Daniel Kressner and Christine Tobler. “Low-Rank Tensor Krylov Subspace Methods for Parametrized Linear Systems”. In: *SIAM Journal on Matrix Analysis and Applications* 32.4 (Oct. 2011), pp. 1288–1316.
- [KBS14] Patrick Kürschner, Peter Benner, and Jens Saak. “Self-Generating and Efficient Shift Parameters in ADI Methods for Large Lyapunov and Sylvester Equations”. In: *Electronic transactions on numerical analysis ETNA* 43 (Jan. 1, 2014), pp. 142–162.
- [Lan70] Peter Lancaster. “Explicit Solutions of Linear Matrix Equations”. In: *SIAM Review* 12.4 (Oct. 1970), pp. 544–566.
- [LY91] Peter Lancaster and Qiang Ye. “Variational and numerical methods for symmetric matrix pencils”. In: *Bulletin of the Australian Mathematical Society* 43.1 (Feb. 1991), pp. 1–17.
- [Lee18] John M. Lee. *Introduction to Riemannian Manifolds*. Vol. 176. Graduate Texts in Mathematics. Springer International Publishing, 2018.
- [Lee+20] Kookjin Lee et al. *Alternating Energy Minimization Methods for Multi-term Matrix Equations*. June 15, 2020. arXiv: 2006.08531 [cs,math].
- [LBG13] Steven J. Leon, Åke Björck, and Walter Gander. “Gram-Schmidt orthogonalization: 100 years and more”. In: *Numerical Linear Algebra with Applications* 20.3 (2013), pp. 492–532.
- [LW02] Jing-Rebecca Li and Jacob White. “Low Rank Solution of Lyapunov Equations”. In: *SIAM Journal on Matrix Analysis and Applications* 24.1 (Jan. 2002), pp. 260–280.
- [LPS14] Gabriel J. Lord, Catherine E. Powell, and Tony Shardlow. *An Introduction to Computational Stochastic PDEs*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2014.
- [MPR18] Stefano Massei, Davide Palitta, and Leonardo Robol. “Solving Rank-Structured Sylvester and Lyapunov Equations”. In: *SIAM Journal on Matrix Analysis and Applications* 39.4 (Jan. 2018), pp. 1564–1590.

- [MPW18] Eleanor McDonald, Jennifer Pestana, and Andy Wathen. “Preconditioning and Iterative Solution of All-at-Once Systems for Evolutionary Partial Differential Equations”. In: *SIAM Journal on Scientific Computing* 40.2 (Jan. 2018), A1012–A1033.
- [Mey11] Gilles Meyer. “Geometric optimization algorithms for linear regression on fixed-rank matrices”. In: (Sept. 2, 2011).
- [Mey+21] Raphael A. Meyer et al. “Hutch++: Optimal Stochastic Trace Estimation”. In: *Symposium on Simplicity in Algorithms (SOSA)*. Proceedings. Society for Industrial and Applied Mathematics, Jan. 2021, pp. 142–155.
- [MIR60] L. MIRSKY. “SYMMETRIC GAUGE FUNCTIONS AND UNITARILY INVARIANT NORMS”. In: *The Quarterly Journal of Mathematics* 11.1 (Jan. 1, 1960), pp. 50–59.
- [MS14] B. Mishra and R. Sepulchre. *R3MC: A Riemannian three-factor algorithm for low-rank matrix completion*. Sept. 20, 2014. arXiv: 1306.2672[cs,math].
- [MV14] B. Mishra and B. Vandereycken. *A Riemannian approach to low-rank algebraic Riccati equations*. May 28, 2014. arXiv: 1312.4883[math].
- [MS16] Bamdev Mishra and Rodolphe Sepulchre. “Riemannian Preconditioning”. In: *SIAM Journal on Optimization* 26.1 (Jan. 2016), pp. 635–660.
- [Mis+14] Bamdev Mishra et al. “Fixed-rank matrix factorizations and Riemannian low-rank optimization”. In: *Computational Statistics* 29.3 (June 1, 2014), pp. 591–621.
- [NS12] Thanh Ngo and Yousef Saad. “Scaled Gradients on Grassmann Manifolds for Matrix Completion”. In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.
- [Nou08] Anthony Nouy. “Generalized spectral decomposition method for solving stochastic finite element equations: Invariant subspace problem and dedicated algorithms”. In: *Computer Methods in Applied Mechanics and Engineering* 197.51 (Oct. 15, 2008), pp. 4718–4736.
- [Nou10] Anthony Nouy. “Proper Generalized Decompositions and Separated Representations for the Numerical Solution of High Dimensional Stochastic Problems”. In: *Archives of Computational Methods in Engineering* 17.4 (Dec. 1, 2010), pp. 403–434.
- [Pal21] Davide Palitta. “Matrix Equation Techniques for Certain Evolutionary Partial Differential Equations”. In: *Journal of Scientific Computing* 87.3 (May 12, 2021), p. 99.
- [PK21] Davide Palitta and Patrick Kürschner. “On the convergence of Krylov methods with low-rank truncations”. In: *Numerical Algorithms* 88.3 (Nov. 1, 2021), pp. 1383–1417.

-
- [PS16] Davide Palitta and Valeria Simoncini. “Matrix-equation-based strategies for convection–diffusion equations”. In: *BIT Numerical Mathematics* 56.2 (June 1, 2016), pp. 751–776.
 - [PR55] D. W. Peaceman and H. H. Rachford Jr. “The Numerical Solution of Parabolic and Elliptic Differential Equations”. In: *Journal of the Society for Industrial and Applied Mathematics* 3.1 (Mar. 1955), pp. 28–41.
 - [Pen98] Thilo Penzl. “Numerical solution of generalized Lyapunov equations”. In: *Advances in Computational Mathematics* 8.1 (Jan. 1, 1998), pp. 33–48.
 - [Pen99] Thilo Penzl. “A Cyclic Low-Rank Smith Method for Large Sparse Lyapunov Equations”. In: *SIAM Journal on Scientific Computing* 21.4 (Jan. 1999), pp. 1401–1418.
 - [Pen00a] Thilo Penzl. “Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case”. In: *Systems & Control Letters* 40.2 (June 15, 2000), pp. 139–144.
 - [Pen00b] Thilo Penzl. “LYAPACK A MATLAB Toolbox for Large Lyapunov and Riccati Equations , Model Reduction Problems , and Linear – Quadratic Optimal Control Problems Users”. In: 2000.
 - [PCK22] David Persson, Alice Cortinovis, and Daniel Kressner. *Improved variants of the Hutch++ algorithm for trace estimation*. May 6, 2022. arXiv: 2109.10659[cs,math].
 - [PSS17] C. E. Powell, D. Silvester, and V. Simoncini. “An Efficient Reduced Basis Solver for Stochastic Galerkin Matrix Equations”. In: *SIAM Journal on Scientific Computing* 39.1 (Jan. 2017), A141–A163.
 - [PE09] Catherine E. Powell and Howard C. Elman. “Block-diagonal preconditioning for spectral stochastic finite-element systems”. In: *IMA Journal of Numerical Analysis* 29.2 (Apr. 1, 2009), pp. 350–375.
 - [Ric11] L. F. Richardson. “The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam”. In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 210 (1911), pp. 307–357.
 - [Ros80] Donald J. Rose. “Matrix identities of the fast fourier transform”. In: *Linear Algebra and its Applications*. Special Volume Dedicated to Alson S. Householder 29 (Feb. 1, 1980), pp. 423–443.
 - [Saa03] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 2003. 537 pp.

-
- [SKB23] Jens Saak, Martin Köhler, and Peter Benner. *M-M.E.S.S. - The Matrix Equation Sparse Solver Library*. In collab. with Jens Saak et al. Version 3.0. May 4, 2023.
 - [Sat16] Hiroyuki Sato. “A Dai–Yuan-type Riemannian conjugate gradient method with the weak Wolfe conditions”. In: *Computational Optimization and Applications* 64.1 (May 1, 2016), pp. 101–118.
 - [Sat21] Hiroyuki Sato. *Riemannian Optimization and Its Applications*. Springer-Briefs in Electrical and Computer Engineering. Springer International Publishing, 2021.
 - [Sat22] Hiroyuki Sato. “Riemannian Conjugate Gradient Methods: General Framework and Specific Algorithms with Convergence Analyses”. In: *SIAM Journal on Optimization* 32.4 (Dec. 31, 2022), pp. 2690–2717.
 - [SI15] Hiroyuki Sato and Toshihiro Iwai. “A new, globally convergent Riemannian conjugate gradient method”. In: *Optimization* 64.4 (Apr. 3, 2015), pp. 1011–1031. arXiv: 1302.0125[math].
 - [SU15] Reinhold Schneider and André Uschmajew. “Convergence Results for Projected Line-Search Methods on Varieties of Low-Rank Matrices Via Łojasiewicz Inequality”. In: *SIAM Journal on Optimization* 25.1 (Jan. 2015), pp. 622–646.
 - [SSS16] Stephen D. Shank, Valeria Simoncini, and Daniel B. Szyld. “Efficient low-rank solution of generalized Lyapunov equations”. In: *Numerische Mathematik* 134.2 (Oct. 1, 2016), pp. 327–342.
 - [Sha23] Meiyue Shao. “Householder Orthogonalization with a Nonstandard Inner Product”. In: *SIAM Journal on Matrix Analysis and Applications* 44.2 (June 30, 2023), pp. 481–502.
 - [Sim07] V. Simoncini. “A New Iterative Method for Solving Large-Scale Lyapunov Matrix Equations”. In: *SIAM Journal on Scientific Computing* 29.3 (Jan. 2007), pp. 1268–1288.
 - [Sim16] V. Simoncini. “Computational Methods for Linear Matrix Equations”. In: *SIAM Review* 58.3 (Jan. 2016), pp. 377–441.
 - [Smi68] R. A. Smith. “Matrix Equation $XA + BX = C$ ”. In: *SIAM Journal on Applied Mathematics* 16.1 (Jan. 1968), pp. 198–201.
 - [SZ70] J. Snyders and M. Zakai. “On Nonnegative Solutions of the Equation $AD + DA' = -C$ ”. In: *SIAM Journal on Applied Mathematics* 18.3 (May 1970), pp. 704–714.
 - [Sta92] Gerhard Starke. “Near-circularity for the rational Zolotarev problem in the complex plane”. In: *Journal of Approximation Theory* 70.1 (July 1, 1992), pp. 115–130.

- [SB15] Martin Stoll and Tobias Breiten. “A Low-Rank in Time Approach to PDE-Constrained Optimization”. In: *SIAM Journal on Scientific Computing* 37.1 (Jan. 2015), B1–B29.
- [TLN14] L. Tamellini, O. Le Maître, and A. Nouy. “Model Reduction Based on Proper Generalized Decomposition for the Stochastic Steady Incompressible Navier–Stokes Equations”. In: *SIAM Journal on Scientific Computing* 36.3 (Jan. 2014), A1089–A1117.
- [Tan+14] Mingkui Tan et al. “Riemannian pursuit for big matrix recovery”. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML’14. JMLR.org, June 21, 2014, pp. II–1539–II–1547.
- [TS07] Radu Alexandru Todor and Christoph Schwab. “Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients”. In: *IMA Journal of Numerical Analysis* 27.2 (Apr. 1, 2007), pp. 232–261.
- [TW18] Alex Townsend and Heather Wilber. “On the singular values of matrices with high displacement rank”. In: *Linear Algebra and its Applications* 548 (July 1, 2018), pp. 19–41.
- [Ull10] Elisabeth Ullmann. “A Kronecker Product Preconditioner for Stochastic Galerkin Finite Element Discretizations”. In: *SIAM Journal on Scientific Computing* 32.2 (Jan. 2010), pp. 923–946.
- [UV14] Andre Uschmajew and Bart Vandereycken. “Line-search methods and rank increase on low-rank matrix varieties”. In: *IEICE Proceedings Series* 46 (A1L-C4 Sept. 14, 2014).
- [UV15] André Uschmajew and Bart Vandereycken. “Greedy rank updates combined with Riemannian descent methods for low-rank optimization”. In: *2015 International Conference on Sampling Theory and Applications (SampTA)*. 2015 International Conference on Sampling Theory and Applications (SampTA). May 2015, pp. 420–424.
- [Van76] Charles F. Van Loan. “Generalizing the Singular Value Decomposition”. In: *SIAM Journal on Numerical Analysis* 13.1 (Mar. 1976), pp. 76–83.
- [Van10] B. Vandereycken. “Riemannian and multilevel optimization for rank-constrained matrix problems”. PhD thesis. Department of Computer Science, KU Leuven, Dec. 2010.
- [Van13] Bart Vandereycken. “Low-Rank Matrix Completion by Riemannian Optimization”. In: *SIAM Journal on Optimization* 23.2 (Jan. 2013), pp. 1214–1236.

- [VV10] Bart Vandereycken and Stefan Vandewalle. “A Riemannian Optimization Approach for Computing Low-Rank Solutions of Lyapunov Equations”. In: *SIAM Journal on Matrix Analysis and Applications* 31.5 (Jan. 2010), pp. 2553–2579.
- [Var00] Richard S. Varga. “Alternating-Direction Implicit Iterative Methods”. In: *Matrix Iterative Analysis*. Ed. by Richard S. Varga. Springer Series in Computational Mathematics. Springer, 2000, pp. 235–274.
- [Wac13] Eugene Wachspress. *The ADI Model Problem*. Springer, 2013.
- [Wac88] Eugene L. Wachspress. “Iterative solution of the Lyapunov matrix equation”. In: *Applied Mathematics Letters* 1.1 (Jan. 1, 1988), pp. 87–90.
- [Wac08] Eugene L. Wachspress. “Trail to a Lyapunov equation solver”. In: *Computers & Mathematics with Applications* 55.8 (Apr. 1, 2008), pp. 1653–1659.
- [WBR20] Roman Weinhandl, Peter Benner, and Thomas Richter. *Low-rank Linear Fluid-structure Interaction Discretizations*. Apr. 23, 2020. arXiv: 1905 . 11000[cs,math].
- [Won68] W. M. Wonham. “On a Matrix Riccati Equation of Stochastic Control”. In: *SIAM Journal on Control* 6.4 (Nov. 1968), pp. 681–697.
- [WN06] Stephen J. Wright and Jorge Nocedal. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.
- [YGL18] Wenjian Yu, Yu Gu, and Yaohang Li. “Efficient Randomized Algorithms for the Fixed-Precision Low-Rank Matrix Approximation”. In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (Jan. 2018), pp. 1339–1359.
- [Zho+16] Guifang Zhou et al. “A Riemannian rank-adaptive method for low-rank optimization”. In: *Neurocomputing*. Advances in artificial neural networks, machine learning and computational intelligence 192 (June 5, 2016), pp. 72–80.

Appendix A

Convergence of RTR with tCG

A.1 Proof of Proposition 3.11

Lemma A.1. *Let $\mathcal{M}$ be a Riemannian manifold equipped with a retraction R . Let $X_\star \in \mathcal{M}$, then there exists constants $\delta > 0$ and $c > 1$ such that R is well-defined on*

$$\mathcal{V}(X_\star, \delta) = \{(X, \xi) \in T\mathcal{M} : \text{dist}(X_\star, X) \leq \delta, \|\xi\|_X \leq \delta\} \subseteq T\mathcal{M}$$

and

$$\text{dist}(X, R_X(\xi)) \leq c \|\xi\|_X, \quad \forall (X, \xi) \in \mathcal{V}(X_\star, \delta)$$

Proof. Follows from [Bou23b]. □

Proposition 3.11 (Local convergence of R-TR with tCG). *Let $X_\star \in \mathcal{M}$ be a non-degenerate local minimizer of f , i.e., $\text{grad}f(X_\star) = 0$ and $\text{Hess}f(X_\star) \succ 0$. Assume that in a neighborhood of $X_\star$ the model Hessian $H_X : T_X\mathcal{M} \rightarrow T_X\mathcal{M}$ is positive definite and $X \mapsto \|H_X\|_X$, $X \mapsto \|H_X^{-1}\|_X$ are bounded. Then Riemannian Trust-Region with tCG is locally convergent to $X_\star$.*

Proof. We reproduce almost verbatim the proof of [AMS08, Theorem 7.4.10], to show that it can be easily adapted with only local arguments.

Take c and δ as in Lemma A.1. Due to all the assumptions, there exists $\delta_1 < \delta$ such that H_X is positive definite in $B_{\delta_1}(X_\star)$, that $\|H_X\|$ and $\|H_X^{-1}\|$ are bounded on $B_{\delta_1}(X_\star)$, that $X_\star$ is the only critical point in $B_{\delta_1}(X_\star)$, and that $f(X) > f(X_\star)$ for all $X \in \overline{B_{\delta_1}(X_\star)} \setminus \{X_\star\}$.

For all $X \in B_{\delta_1}(X_\star)$ let us denote $\eta^*(X) = -H_X^{-1}(\text{grad}f(X))$ and let $\eta^{\text{tCG}}(X, \Delta)$ be the tangent vector returned by tCG when $X_k = X$ and $\Delta_k = \Delta$. Take $\delta_2 < \delta_1$ small enough such that $\|\eta^*(X)\|_X \leq \frac{1}{c}(\delta_1 - \delta_2)$ for all $X \in B_{\delta_2}(X_\star)$. Such a δ_2 exists for the bound on $\|H_X^{-1}\|$ and [AMS08, Lemma 7.4.8]. Consider a level set $\mathcal{L}$ of f such that $\mathcal{U} := \mathcal{L} \cap B_{\delta_1}(X_\star) \subseteq B_{\delta_2}(X_\star)$ and $\mathcal{U} \neq \emptyset$. Such a level set exists because f is smooth, see for instance [Bou23a, § 10.4]. Then $\mathcal{U}$ is a neighborhood of $X_\star$.

A.1. Proof of local convergence of RTR with tCG

Let $X \in \mathcal{U}$ and $\Delta > 0$, since the iterates of tCG grow in norm [CGT00, Theorem 7.5.1], it holds $\|\eta^{\text{tCG}}(X, \Delta)\|_X \leq \|\eta^*(X)\|_X$. We have

$$\|\eta^{\text{tCG}}(X, \Delta)\|_X \leq \|\eta^*(X)\|_X \leq \frac{1}{c}(\delta_1 - \delta_2) < \frac{1}{c}\delta < \delta,$$

thus $(X, \eta^{\text{tCG}}(X, \Delta)) \in \mathcal{V}(X_*, \delta)$ and

$$\text{dist}(X, X_+) \leq \text{dist}(X, R_X(\eta^{\text{tCG}}(X, \Delta))) \leq c \|\eta^{\text{tCG}}(X, \Delta)\|_X \leq c \|\eta^*(X)\|_X \leq (\delta_1 - \delta_2).$$

Therefore, $X_+ \in B_{\delta_1}(X_*)$, and since $f(X_+) \leq f(X)$ it follows $X_+ \in \mathcal{U}$.

Thus, if $X_k \in \mathcal{U}$ for some k then all subsequent iterates stay in $\mathcal{U} \subseteq \overline{B_{\delta_2}(X_*)}$. Since $\overline{B_{\delta_2}(X_*)}$ is compact and (3.12) holds in $\mathcal{U}$, by Proposition 3.9 it holds $\lim_{k \rightarrow \infty} \|\text{grad} f(X_k)\|_{X_k} = 0$. But the only critical point of f in $\mathcal{U}$ is X_* and therefore $X_k \rightarrow X_*$. $\square$