# A Riemannian perspective on matrix recovery and constrained optimization

Florentin Goyens

Oriel College

University of Oxford

A thesis submitted for the degree of

*Doctor of Philosophy*

Trinity 2021

# Acknowledgements

My DPhil has been an incredibly rich experience, where I have learned a lot and grown as a researcher and as a person. I can truly say that I have been privileged to work with nothing but talented and kind-hearted people, to whom I would like to express my gratitude. First of all, I would like to thank my supervisors, Coralia Cartis and Armin Eftekhari.

Coralia, I am extremely grateful for all your energy, your kindness and your dedication to me during these years. You have given me the freedom to experiment and always trusted my ideas, while keeping me on the right track. Thank you for these wonderful years, it was a pleasure to work with you.

Armin, you welcomed me with open arms at the Turing at the start of my DPhil. Thank you for sharing with me your passion for mathematics, your motivation and your sense of hard work.

I would also like to thank some collaborators. To Stéphane Chrétien, thank you for your interest in my work, your endless creativity and enthusiasm. Thank you to Nicolas Boumal, your dynamism made every meeting enjoyable and I learned a lot from your expertise.

I would also like to thank Estelle Massart, for some very helpful discussions. To all the researchers and students in the numerical analysis group in Oxford, it was a pleasure to work alongside you everyday. I will surely miss the Thursday lunches at Balliol college. Additionally, I should thank my family, friends and most of all Louise, who has provided an incredible support during these years. I also gratefully acknowledge the Alan Turing Institute for funding my research and allowing me to travel to conferences.

# Abstract

Nonlinear matrix recovery is an emerging paradigm in which specific classes of high-rank matrices can be recovered from an underdetermined linear system of measurements. In particular, we consider matrices whose columns, seen as data points, belong to an algebraic variety, namely, a set defined by finitely many polynomial equations. The nonlinear structure is exploited using an appropriately chosen feature map, which induces a low-rank structure in the feature space, thus allowing recovery of the original (high-rank) matrix. The same framework is applicable to matrices whose columns are divided in clusters, so that we can address clustering tasks with missing entries.

We formulate the rank minimization of the feature matrix as the minimization of a smooth cost function on a Riemannian manifold, which then allows us to employ the theoretically rigorous Riemannian optimization framework, that calls on differential geometry tools to construct feasible iterative algorithms on specific Riemannian manifolds. In particular, we show the potential benefits of using second-order algorithmic variants — instead of first-order ones — as they are able to achieve recovery of the original data with high accuracy. We additionally develop an alternating minimization method to solve the recovery problem, for which we give convergence and complexity guarantees.

Considering algebraic varieties also leads us to propose a new approach for the denoising of point clouds that approximately fit such structures, as well as for their registration, namely, the task of estimating a transformation which overlaps two point clouds representing the same shape in different coordinate systems.

Finally, we consider general optimization problems with smooth equality constraints. Due to the challenge of achieving feasibility for generic nonlinear equality constraints, we must depart from the Riemannian framework and we propose an infeasible algorithm which minimizes a smooth penalty function, known as Fletcher's augmented Lagrangian. We provide optimal worst-case bounds on the number of iterations that are required to find an approximate first- and second-order critical point.

# Contents

**1 Introduction** **9**

**2 Riemannian optimization** **16**

2.1 Charts and manifolds . . . 16

2.2 Tangent vectors and differential map . . . 18

2.2.1 Embedded submanifolds . . . 20

2.2.2 Quotient manifolds . . . 20

2.3 Riemannian structure and gradients . . . 22

2.3.1 Riemannian submanifolds . . . 23

2.3.2 Riemannian quotient manifolds . . . 24

2.4 Connections and Hessians . . . 25

2.4.1 Riemannian submanifolds . . . 27

2.4.2 Riemannian quotient manifolds . . . 27

2.5 Optimality conditions . . . 28

2.6 Retractions . . . 29

2.7 Parallel transport . . . 30

2.8 Algorithms and implementations . . . 31

2.8.1 Riemannian gradient descent . . . 31

2.8.2 Riemannian trust-region . . . 32

2.8.3 Riemannian optimization toolboxes: Manopt and PyManopt . . . 35

2.9 Manifolds of interest . . . 36

2.9.1 Euclidean spaces . . . 37

2.9.2 Affine subspaces . . . 37

2.9.3 The Stiefel manifold: orthonormal matrices . . . 38

2.9.4 The orthogonal group and rotation matrices . . . 40

2.9.5 The Grassmann manifold . . . 41

2.9.6 Manifolds defined by $h(x) = 0$ . . . 43

**3 Matrix recovery problems** **46**
3.1 Low-rank matrix recovery . . . . . . . . . . 46
3.2 Nonlinear matrix recovery . . . . . . . . . . 52
3.2.1 Problem description and feature map . . . . . . . . . . 52
3.2.2 The algebraic variety model . . . . . . . . . . 53
3.2.3 Union of subspaces . . . . . . . . . . 55
3.2.4 Clustering and the Gaussian kernel . . . . . . . . . . 57
3.2.5 Related work . . . . . . . . . . 58

**4 Nonlinear matrix recovery on the Grassmann manifold** **62**
4.1 Nonlinear matrix recovery as an optimization problem . . . . . . . . . . 64
4.1.1 Kernel representation of the features . . . . . . . . . . 65
4.1.2 Derivatives of the cost function . . . . . . . . . . 66
4.2 Riemannian optimization algorithms . . . . . . . . . . 67
4.3 Alternating minimization algorithms . . . . . . . . . . 69
4.4 Convergence of the alternating minimization algorithm . . . . . . . . . . 72
4.4.1 Global convergence results . . . . . . . . . . 73
4.4.2 Convergence of the iterates using the Kurdyka-Lojasiewicz property 77
4.5 Discussion of assumptions in convergence results . . . . . . . . . . 85
4.6 Numerical experiments . . . . . . . . . . 88
4.6.1 Implementation of the algorithms . . . . . . . . . . 88
4.6.2 Test problems . . . . . . . . . . 89
4.6.3 Testing methodology . . . . . . . . . . 89
4.6.4 Numerical results . . . . . . . . . . 90
4.7 Conclusions . . . . . . . . . . 98

**5 Applications of the algebraic variety model** **99**
5.1 Denoising an algebraic variety . . . . . . . . . . 100
5.1.1 Problem description . . . . . . . . . . 101
5.1.2 The feature space and the monomial features . . . . . . . . . . 102
5.1.3 Denoising as an optimization problem . . . . . . . . . . 102
5.1.4 Statistical error estimation . . . . . . . . . . 105
5.1.5 Numerical results . . . . . . . . . . 108
5.2 Registration . . . . . . . . . . 112
5.2.1 Problem set up . . . . . . . . . . 116
5.2.2 Registration as an optimization problem . . . . . . . . . . 117
5.2.3 Numerical results for noiseless registration . . . . . . . . . . 119
5.2.4 Noisy registration . . . . . . . . . . 121

5.2.5 Numerical results for noisy registration . . . . . . . . . . . . . . . 122
5.3 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126

**6 Equality constrained optimization 128**
6.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
6.1.1 Assumptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
6.1.2 Layered manifolds . . . . . . . . . . . . . . . . . . . . . . . . . . 130
6.1.3 Optimality conditions . . . . . . . . . . . . . . . . . . . . . . . . 132
6.2 Related work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133
6.3 Properties of Fletcher's augmented Lagrangian . . . . . . . . . . . . . . . 138
6.3.1 Approximate first-order criticality . . . . . . . . . . . . . . . . . . 141
6.3.2 Approximate second-order criticality . . . . . . . . . . . . . . . . . 143
6.3.3 Property of the region $\mathcal{C}$ . . . . . . . . . . . . . . . . . . . . . . 146
6.4 Gradient-Eigenstep algorithm . . . . . . . . . . . . . . . . . . . . . . . 148
6.4.1 Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148
6.4.2 First-order steps . . . . . . . . . . . . . . . . . . . . . . . . . . . 150
6.4.3 Second-order steps . . . . . . . . . . . . . . . . . . . . . . . . . . 152
6.4.4 Worst-case global complexity . . . . . . . . . . . . . . . . . . . . . 154
6.5 Estimating the penalty parameter . . . . . . . . . . . . . . . . . . . . . 156
6.6 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160

**Bibliography 162**

**A Proofs 176**
A.1 Proofs for Chapter 4 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
A.2 Proofs for Chapter 6 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 179

# Notation and symbols

## Matrices

| | |
|---|---|
| $\sigma_{\min}(A)$ | The smallest singular value of the matrix $A$ |
| $\sigma_{\max}(A)$ | The largest singular value of the matrix $A$ |
| $\sigma_i(A)$ | The $i$th singular value of the matrix $A$, in decreasing order |
| $\lambda_{\min}(A)$ | The smallest eigenvalue of the matrix $A$, in magnitude |
| $\lambda_{\max}(A)$ | The largest eigenvalue of the matrix $A$, in magnitude |
| $\|A\|_{\mathrm{F}}$ | The Frobenius norm of the matrix $A$, $\|A\|_{\mathrm{F}} = \sqrt{A^\top A}$ |
| $A \succeq 0$ | Positive semidefinite matrix $A$ |
| $\mathrm{I}_n$ | Identity matrix of size $n$ |
| $\mathbb{1}_{n\times s}$ | Matrix of all ones of size $n \times s$ |
| $\mathrm{skew}(A)$ | $(A - A^\top)/2$, the skew-symmetric part of a matrix $A$ |
| $\mathrm{sym}(A)$ | $(A + A^\top)/2$, the symmetric part of a matrix $A$ |
| $\mathrm{null}(A)$ | The null space of the matrix $A$ |

## Sets

| | |
|---|---|
| $\mathcal{M}$ | A Riemannian manifold |
| $\mathcal{E}$ | A Euclidean space |
| $\mathrm{St}(n, r)$ | The Stiefel manifold $\{X \in \mathbb{R}^{n\times r} : X^\top X = \mathrm{I}_r\}$ |
| $\mathrm{O}(n)$ | The orthogonal group $\{X \in \mathbb{R}^{n\times n} : X^\top X = \mathrm{I}_n\}$ |
| $\mathrm{SO}(n)$ | The special orthogonal group $\{X \in \mathbb{R}^{n\times n} : X^\top X = \mathrm{I}_n, \det(X) = 1\}$ |
| $\mathrm{Grass}(n, r)$ | The Grassmann manifold of linear subspaces of dimension $r$ in $\mathbb{R}^n$ |
| Sym(p) | The set of symmetric matrices of size $p$ |

## Tools on manifolds

| | |
|---|---|
| $\mathfrak{F}(\mathcal{M})$ | The set of real-valued functions on the manifold $\mathcal{M}$ |
| $\mathfrak{X}(\mathcal{M})$ | The set of vector fields on the manifold $\mathcal{M}$ |
| $\mathrm{T}_x\mathcal{M}$ | Tangent space at $x$ to the manifold $\mathcal{M}$ |
| $\mathrm{Proj}_x$ | Orthogonal projection onto $\mathrm{T}_x\mathcal{M}$ |
| $\mathrm{R}_x$ | Retraction at $x$ |

## Functions

| | |
|---|---|
| Id | Identity map |
| $\nabla f(x)$ | Classical gradient of $f$, seen as a function in a Euclidean space |
| $\mathrm{grad} f(x)$ | Riemannian gradient of $f$, with respect to the manifold $f$ is defined on |
| $\nabla^2 f(x)$ | Classical Hessian of $f$, seen as a function in a Euclidean space |
| $\mathrm{Hess} f(x)$ | Riemannian Hessian of $f$ |
| $C^\infty$ | The class of infinitely differentiable functions |

## Abreviations

| | |
|---|---|
| PCA | Principal component analysis |
| SURE | Stein's unbiased risk estimator |
| RTR | Riemannian trust-region |
| SVD | Singular value decomposition |

# Chapter 1

# Introduction

This thesis revolves around two core concepts: recovery problems involving data sets of low intrinsic dimension and optimization over smooth manifolds. These two problems are tightly connected as we are able to phrase recovery problems as the minimization of cost functions over smooth manifolds, thus allowing us to use the powerful set of Riemannian optimization methods to solve them.

With the emergence of data science and the desire to extract meaningful information from the ever-increasing amount of collected data, new mathematical challenges arise constantly. One of which is the prevalent *matrix completion* problem, with applications in collaborative filtering, system identification and perhaps most notably in recommender systems. Given a subset of the entries of a matrix $M$, the goal is to "complete" the matrix in a meaningful way. The rows of the matrix typically represent products and the columns represent customers. Each available entry represents the rating of a customer to a specific product and recommendations are based on the completed matrix, as customers are recommended products for which the completion estimates a good rating. Traditionally, the matrix completion problem has been tackled under the assumption that a low-rank matrix completes the observed data, which means that the profile of any customer can be expressed as a linear combination of a few singular profiles. Low-rank matrix completion is a well-developed, yet still active, subject which is summarized in surveys such as (Davenport and Romberg, 2016; Nguyen et al., 2019). Successful approaches include optimization methods where rank constraints are formulated as specific smooth manifolds (Vandereycken, 2013; Boumal and Absil, 2015).

However, some data sets are poorly approximated by low-rank models. This is the case of nonlinear surfaces such as certain three-dimensional scans encountered in medical imaging. As a mathematical model representing nonlinear surfaces, we consider problems where the data is represented by an algebraic variety, namely, a set defined by finitely many polynomial equations. In particular, when the data belongs to a finite union of subspaces, it is also represented by an algebraic variety. Data sets divided into clusters constitute another example where low-rank models are of limited use, which we also investigate. Such data sets, though they may be intrinsically low-dimensional in some way, feature a complex relationship between the data points which includes nonlinearities. These examples abound in computer vision and machine learning applications (Bishop, 2006; Xu and Wunsch, 2008; Forsyth and Ponce, 2011; Shalev-Shwartz and Ben-David, 2014). It is therefore natural to investigate whether matrix completion techniques can be adapted to particular classes of high-rank matrices.

An approach was recently proposed, which recovers a high-rank matrix $M$ from linear measurements on the condition that suitable *features* of the columns of $M$ are linearly dependent (Ongie et al., 2017; Fan and Chow, 2018). That is, the approach relies on the existence of a nonlinear function $\Phi$, a feature map, such that $\Phi(M)$ is low-rank. This approach, known as *nonlinear matrix recovery*, lifts the problem to a higher-dimensional feature space, where a low-rank structure of the lifted data matrix can be found. The recovery problem comes down to the rank minimization of the feature map $\Phi$ while satisfying the linear measurements on the matrix $M$. The use of a feature map, which may be represented by a kernel, is a well-known technique in machine learning, which appears in methods such as kernel Principal Component Analysis (PCA) (Schölkopf et al., 1997). Kernel PCA classifies data points into several categories which are not linearly separable in the original space but become linearly separable in the feature space. Kernel PCA has also proved efficient in image denoising and in detecting patterns in data sets which do not have a linear structure (Hoffmann, 2007). The introduction of a feature map for matrix completion poses several computational challenges, as the dimension of the problem increases dramatically, and low-rank matrix completion techniques must be adapted to the nonlinear feature map $\Phi$, in order to estimate $M$. In this thesis, we show how nonlinear matrix recovery can be phrased as a nonconvex optimization problem over a smooth manifold, thus adapting low-rank matrix recovery techniques to the nonlinear case.

Nonlinear recovery questions and algebraic varieties also appear jointly in two other applications: the denoising of point clouds and their registration, which consists in estimating a transformation that overlaps two point clouds representing the same object. This problem is commonly encountered in medical imaging applications, amongst others. CT scans (computed tomography) for dental care is a technology that measures a tooth and produces a three dimensional point cloud of its surface. Scans taken months or years apart can be of great help to diagnose tooth decay. It is therefore desirable to have an automated way to overlap two scans of the same dental records which may have been taken from different angles. CT scans provide point clouds of up to several million points which can help estimate the surface of a tooth. These scans may also contain outliers, which should be handled in a preprocessing phase before the registration. Point cloud registration is a vast and active topic of research where approaches vary depending on the type of transformation that is required for the point clouds to overlap (Goshtasby, 2005; Van Kaick et al., 2011; Tam et al., 2013; Chaudhury et al., 2015; Huang et al., 2021). The approach adopted here is to assume that the sampled point clouds belong, exactly or approximately, to an algebraic variety. The estimation of a transformation such that two algebraic varieties align is formulated as an optimization problem involving the group of rotations in $\mathbb{R}^n$, which also has a manifold structure that we can exploit.

The problems considered in this thesis naturally involve the minimization of a cost function subject to smooth constraints, such as fixed-rank manifolds and the special orthogonal group of rotations. We solve these problems using the Riemannian optimization framework, whose central idea is to treat the problem as the unconstrained minimization of a function whose domain of definition is the feasible set. In this way, the constraint is implicitly part of the problem definition. Riemannian optimization uses differential geometry tools to generalize to manifolds classical methods from unconstrained optimization, such as gradient descent and trust-regions. The Riemannian counterparts of these methods iteratively travel along the manifold and produce a feasible sequence of iterates. Each step finds a new direction based on a local linearization of the manifold, namely, the tangent space. The appeal of formulating optimization problems on manifolds resides in the fact that these methods are able to keep the computational costs proportional to the intrinsic dimension of the manifold, and avoid computations in the ambient space. Riemannian optimization forms the backbone of the algorithms used in this thesis.

Riemannian optimization methods, though not widespread, have steadily gained in popularity since the 1990s, when the first practical algorithms were introduced (Edelman et al., 1998). A great contributor to the development of the field is the textbook (Absil et al., 2008), which lays out the foundations to define optimization methods on Riemannian manifolds. The book also conveys the idea that Riemannian optimization is naturally related to a number of classical problems in numerical analysis, such as the computation of eigenvectors and low-rank approximations. A more recent introduction to Riemannian optimization can be found in (Boumal, 2020), without prerequisites in differential geometry.

Smooth manifolds are typically nonconvex sets; thus the ensuing optimization problems may be difficult to solve in general. One is commonly only guaranteed to find a local solution, that may not be the global minimizer of the problem. Therefore, it is important to distinguish the output of an algorithm from the desired solution. Nevertheless, some favourable circumstances may allow the calculation of global minimizers of a nonconvex problem. For example, a particularly good initialization may be available, with the function having a convex behaviour near the desired solution. It has been shown that some nonconvex problems do not have spurious local minima and that all saddle points can be escaped using a negative curvature direction prescribed by the second derivatives. Finally, it is sometimes possible to show that a particular convex relaxation of the problem is tight, which means that the global minimizer of the relaxation is also the global minimizer of the nonconvex problem. A classical example of a nonconvex problem for which it is possible to find the global minimizer is the computation of eigenvalues, a fact that we exploit in this thesis. The study of nonconvex problems for which it is possible to find a global minimum using local optimization methods is a very active topic of research. These problems appear in a variety of settings, such as low-rank matrix completion (Candes and Plan, 2011; Ge et al., 2016; Bhojanapalli et al., 2016; Uschmajew and Vandereycken, 2018; Li and Tang, 2017); dictionary learning (Sun et al., 2017); phase retrieval (Sun et al., 2015, 2018); phase synchronization (Bandeira et al., 2017; Boumal, 2016), principal component analysis (Hauser et al., 2018) and the Burer-Monteiro factorization of semidefinite programs (Cifuentes, 2019; Boumal et al., 2020).

The traditional performance analysis of optimization methods has addressed the local convergence rate, that is, the speed at which the iterative process asymptotically converges towards a minimizer. Methods which rely on first derivatives of the problem's function typically converge at best at a linear rate. Methods which use second-order derivatives or their approximation may hope to converge superlinearly or quadratically, which in practice, ensures convergence in a small number of iterations. However, these results do not give any indication of the behaviour of the method in the early iterations, notably in the case of a poor initial guess. Moreover, optimization algorithms aim to find points which satisfy approximate necessary optimality conditions involving the derivatives of the cost function, so that they may return in finite time. This is not taken into account in the analysis of the asymptotic behaviour. Therefore, an important task in safeguarding the performance of these methods is to give worst-case estimates on the number of iterations that a method requires to satisfy an approximate criticality condition from an arbitrary starting point. The study of global (worst-case) complexity analysis of optimization methods is a recent perspective which we contribute to in this thesis. For unconstrained nonconvex problems, the worst-case complexity of classical methods such as gradient descent, Newton's method and regularization variants was derived in (Nesterov, 2004, 2018; Cartis et al., 2010, 2011, 2012).

As most of the optimization problems that appear in this manuscript are constrained problems, we investigate the worst-case complexity of algorithms for constrained optimization. Remarkably, Riemannian optimization methods achieve the same complexity bounds as their unconstrained counterparts (Boumal et al., 2019; Agarwal et al., 2018). However, Riemannian methods can only be used for constraints for which it is straightforward to generate a feasible point and maintain a feasible sequence of iterates using projection-like mappings to the feasible set. For many constrained problems, as smooth as they may be, Riemannian methods are therefore not applicable. In those cases, traditional infeasible methods must be used and the study of their worst-case complexity is also of interest (Cartis et al., 2015, 2019). An upcoming research monograph summarizes these efforts for the unconstrained and constrained cases (Cartis et al., 2022). Infeasible methods are initialized in the ambient space and attempt to converge towards the feasible set by penalizing infeasibility in addition to minimizing the cost function.

The study of nonlinear constrained optimization problems goes back a long way with the introduction of penalty methods in the 1960s and Augmented Lagrangian methods in the 1970s (Bertsekas, 1982). Augmented Lagrangian methods are still widely used today in optimization software, which has prompted recent research in the study of their worst-case complexity (Grapiglia and xiang Yuan, 2019; Xie and Wright, 2019; Birgin and Martnez, 2019).

In this thesis, we consider problems with smooth equality constraints which require the use of infeasible methods. We devote some attention to the definition of an approximate second-order critical point in the presence of equality constraints. We show worst-case complexity bounds for finding such points when using an algorithm based on the minimization of a smooth penalty function known as Fletcher's augmented Lagrangian (Fletcher, 1970). The rates are sharp as they match the best bounds available in the unconstrained setting and our analysis draws on concepts from Riemannian optimization.

## Outline of the thesis

Chapters 2 and 3 provide the background material that the research presented in this thesis builds upon. Chapter 2 is concerned with Riemannian optimization. It defines the tools from differential geometry which are used to develop optimization methods on manifolds. Optimality conditions for optimization problems defined on smooth manifolds are also presented. We further outline the particular manifolds as well as algorithms that appear throughout the thesis. Chapter 3 surveys the existing literature for both low-rank matrix recovery and nonlinear matrix recovery. The low-rank section offers some background, as nonlinear matrix recovery is, in part, an extension of low-rank matrix recovery to particular classes of high-rank matrices. The section on nonlinear matrix recovery is an up-to-date overview of this emerging field. We formulate the problem, which relies on the definition of a feature map, and introduce our case studies: the algebraic variety model, unions of subspaces and clustering, which are used in Chapters 4 and 5.

Chapter 4 presents our main contributions to the nonlinear matrix recovery area. In addition to algebraic variety data, we introduce a new application of nonlinear matrix recovery, namely, clustering with missing entries. We propose a formulation which is new in the context of nonlinear matrix recovery and requires to solve an optimization problem on the Grassmann manifold, the set of all linear subspaces of a given dimension in $\mathbb{R}^n$. We present two approaches to solve it: a Riemannian optimization algorithm and an alternating minimization method. Both classes of algorithms have theoretical guarantees. In particular, for the alternating minimization, we establish global convergence and worst-case complexity bounds. Additionally, using the Kurdyka-Lojasiewicz property, we show that the alternating minimization converges to a unique limit point. We provide extensive numerical results for the recovery of unions of subspaces and clustering with missing entries. Our methods are competitive with existing approaches and, in particular, high accuracy is achieved in the recovery using Riemannian second-order methods. The content of this chapter is presented in (Goyens et al., 2021) and a preliminary part appears in the conference proceedings (Goyens et al., 2019).

In Chapter 5 we investigate two other estimation problems which use the algebraic variety model. Both of these problems are based on a concept central to Chapter 4, the rank minimization of the monomial features. This chapter is a collaboration with Stéphane Chrétien (National Physics Laboratory; now at Lyon 2). Section 5.1 is concerned with the problem of denoising a point cloud using an approximation by an algebraic variety. We include an analysis of the estimation error, using Stein's unbiased risk estimator (SURE). The majority of Section 5.1 has been presented in a conference proceedings (Goyens et al., 2020). It serves as an important building block for the next section which deals with the registration problem. In Section 5.2, we estimate the transformation between two noisy data sets representing the same algebraic variety up to a change of coordinates. The problem is phrased as optimization over the special orthogonal group $\mathrm{SO}(n)$. We validate our approach on synthetic problems and real data such as dental scans.

Chapter 6 is dedicated to optimization with smooth equality constraints. In collaboration with Nicolas Boumal, we consider well-behaved equality constraints which require

the use of infeasible methods and we propose first- and second-order approximate criticality conditions which have a natural geometric interpretation. We consider a penalty function known as Fletcher's Augmented Lagrangian and prove that approximately critical points of the penalty function coincide with approximately critical points of the constrained problem. We present an algorithm to minimize Fletcher's augmented Lagrangian, which has first- and second-order variants. We then establish sharp worst-case bounds on the number of iterations that the algorithm takes to satisfy our first- and second-order approximate criticality conditions.

# Chapter 2

# Riemannian optimization

This chapter gives an overview of Riemannian optimization, the task of minimizing a function defined over a smooth manifold, a recurring theme in this thesis. We define smooth manifolds and the choice of a Riemannian structure. We describe the main concepts from differential geometry that play a role in Riemannian optimization algorithms. These include tangent vectors, the Riemannian gradient and Riemannian Hessian of a real-valued function defined on a Riemannian manifold. The retraction map, which allows to travel along the manifold in a direction prescribed by a tangent vector, is an instrumental tool to maintain feasibility across the iterations. Optimality conditions for optimization on manifolds are also addressed. We define two classical minimization algorithms on Riemannian manifolds: Riemannian gradient descent and Riemannian trust-region. We end the chapter with the description of a number of specific manifolds which appear throughout the thesis. All the notions presented here are standard and we follow closely the presentation in (Absil et al., 2008; Boumal, 2014, 2020).

## 2.1 Charts and manifolds

A smooth manifold is a set that can locally be identified with an open set in $\mathbb{R}^n$. This identification is called a chart. A collection of compatible charts which cover the set is called an atlas for that set. The set together with the atlas form a manifold.

**Definition 2.1** (charts). *Let $M$ be a set. A chart of $M$ is a pair $(U, \varphi)$ where $U \subset M$ and $\varphi$ is a bijection between $U$ and an open set of $\mathbb{R}^n$. $U$ is the chart's domain and $n$ is the chart's dimension. Given $x \in U$, the elements of $\varphi(x) = (\varphi(x)_1, \varphi(x)_2, \ldots, \varphi(x)_n)$ are the coordinates of the point $x \in U$ in the chart $(U, \varphi)$.*

**Definition 2.2** (compatible charts). *Two charts $(U, \varphi)$ and $(V, \psi)$ of $M$, are* smoothly compatible *if they have the same dimension $n$ and either $U \cap V = \emptyset$ or $U \cap V \neq \emptyset$ and*

- *$\varphi(U \cap V)$ is an open set of $\mathbb{R}^n$*

- *$\psi(U \cap V)$ is an open set of $\mathbb{R}^n$*
- *$\psi \circ \varphi^{-1} : \varphi(U \cap V) \to \psi(U \cap V)$ is a smooth diffeomorphism (a smooth invertible function with smooth inverse).*

**Definition 2.3** (atlas)**.** *A set $\mathcal{A} = \{(U_i, \varphi_i), i \in I\}$ of pairwise smoothly compatible charts such that $\cup_{i \in I} U_i = M$ is a smooth atlas of $M$.*

Two atlases $\mathcal{A}_1$ and $\mathcal{A}_2$ are *compatible* if $\mathcal{A}_1 \cup \mathcal{A}_2$ is an atlas. In other words, for every chart $(\mathcal{U}, \varphi)$ in $\mathcal{A}_2$, the set of charts $\mathcal{A}_1 \cup \{(\mathcal{U}, \varphi)\}$ is still an atlas. Given an atlas $\mathcal{A}$, one can generate $\mathcal{A}^+$, the set of all charts $(\mathcal{U}, \varphi)$ such that $\mathcal{A} \cup \{(\mathcal{U}, \varphi)\}$ is also an atlas. It is clear that $\mathcal{A}^+$ is also an atlas, called the *maximal atlas* generated by the atlas $\mathcal{A}$. A maximal atlas of a set $M$ is called a *differentiable structure* on $M$. A maximal atlas defines a topology on $\mathcal{M}$ called the *manifold topology* (or atlas topology).

**Definition 2.4** (manifold)**.** *A smooth (n-dimensional) manifold is a pair $\mathcal{M} = (M, \mathcal{A}^+)$, where $M$ is a set and $\mathcal{A}^+$ is a maximal atlas of $M$ into $\mathbb{R}^n$, such that the atlas topology is Hausdorff and second-countable.*

The requirement that the atlas topology is Hausdorff and second-countable avoids counter-intuitive situations. For details on these (unusual) topological issues, see (Absil et al., 2008, Section 3.1.2). For a manifold $\mathcal{M} = (M, \mathcal{A}^+)$, we often use $\mathcal{M}$ and $M$ interchangeably when the differentiable structure is clear from the context. The differentiable structure allows to define a smooth map between two manifolds.

**Definition 2.5** (smooth mapping)**.** *Let $\mathcal{M}$ and $\mathcal{N}$ be two smooth manifolds. A mapping $f\colon \mathcal{M} \to \mathcal{N}$ is of class $\mathcal{C}^k$ if, for all $x$ in $\mathcal{M}$, there is a chart $(U, \varphi)$ of $\mathcal{M}$ and a chart $(V, \psi)$ of $\mathcal{N}$ such that $x \in U, f(U) \subset V$ and*

$$\psi \circ f \circ \varphi^{-1} : \varphi(U) \to \psi(V)$$

*is of class $\mathcal{C}^k$, that is, if $\psi \circ f \circ \varphi^{-1}$ is $k$ times continuously differentiable. The latter is called the local expression of $f$ in the charts $(U, \varphi)$ and $(V, \psi)$. A* smooth map *is of class $\mathcal{C}^\infty$.*

This definition does not depend on the choice of charts.

## 2.2 Tangent vectors and differential map

A function $f\colon \mathcal{M} \to \mathbb{R}$ is called a scalar field on $\mathcal{M}$. We wish to generalize the usual notion of directional derivative

$$\mathrm{D}f(x)[\xi] = \lim_{t\to 0} \frac{f(x+t\xi) - f(x)}{t}$$

to a real-valued function defined on a manifold. One approach is to replace $t \mapsto (x+t\xi)$ by a smooth curve $c$ on $\mathcal{M}$ that passes through $x$. The function $f \circ c$ goes from $\mathbb{R} \to \mathbb{R}$ so its derivative is well defined in the usual sense, $\frac{\mathrm{d}}{\mathrm{d}t} f(c(t))\big|_{t=0}$. A priori, this definition depends on the curve $c$. To alleviate this issue, we introduce tangent vectors as equivalence classes of curves, as is customary in differential geometry. Consider

$$C_x = \{c : I \to \mathcal{M} : c \in \mathcal{C}^1,\ 0 \in I \text{ an open interval in } \mathbb{R},\ c(0) = x\},$$

the set of differentiable curves on $\mathcal{M}$ that pass through $x \in \mathcal{M}$ at $t = 0$. We introduce an equivalence relation on $C_x$. Let $(U, \varphi)$ be a chart such that $x \in U$ and let $c_1, c_2 \in C_x$. The curves $c_1$ and $c_2$ are said to be equivalent if $\varphi(c_1(t))$ and $\varphi(c_2(t))$ have the same derivative at $t = 0$, that is:

$$c_1 \sim c_2 \Leftrightarrow \frac{\mathrm{d}}{\mathrm{d}t}\varphi(c_1(t))\bigg|_{t=0} = \frac{\mathrm{d}}{\mathrm{d}t}\varphi(c_2(t))\bigg|_{t=0}.$$

The tangent space is then defined as the set of equivalence classes in $C_x$.

**Definition 2.6.** *The* tangent space *to $\mathcal{M}$ at $x$, noted $\mathrm{T}_x\mathcal{M}$, is the quotient space*

$$\mathrm{T}_x\mathcal{M} = C_x/\!\sim\, = \{[c] : c \in C_x\}.$$

*Given $c \in C_x$, the equivalence class $[c]$ is an element of $\mathrm{T}_x\mathcal{M}$ called a* tangent vector *to $\mathcal{M}$ at $x$.*

The tangent space admits a structure of vector space, as described in (Absil et al., 2008, Section 3.5.1). This is important, as the tangent space may be viewed as a local vector space approximation of the manifold, in the same way that the derivative of a real-valued function provides a linear approximation of that function.

**Definition 2.7.** *The directional derivative of a scalar field $f$ on $\mathcal{M}$ at $x \in \mathcal{M}$ in the direction $\xi = [c] \in \mathrm{T}_x\mathcal{M}$ is the scalar:*

$$\mathrm{D}f(x)[\xi] := \frac{\mathrm{d}}{\mathrm{d}t} f(c(t))\bigg|_{t=0} = (f \circ c)'(0). \tag{2.1}$$

The equivalence relation over $C_x$ ensures that this definition does not depend on the choice of $c$ to represent the tangent vector $\xi \in \mathrm{T}_x\mathcal{M}$. An alternative approach is to define tangent vectors as abstract derivation objects. Let $\mathfrak{F}(\mathcal{M})$ denote the set of smooth real-valued functions on $\mathcal{M}$. A derivation at $x \in \mathcal{M}$ is a mapping $\xi \colon \mathfrak{F}(\mathcal{M}) \to \mathbb{R}$ such that

1. $\xi(af + bg) = a\xi(f) + b\xi(g)$, and

2. $\xi(fg) = \xi(f)g(x) + f(x)\xi(g)$ for all $f, g \in \mathfrak{F}(\mathcal{M})$ and $a, b \in \mathbb{R}$.

The notions of tangent vectors as derivations and equivalence classes of curves are equivalent in the following sense: given a smooth curve $c$ on $\mathcal{M}$ with $c(0) = x$, the mapping (2.1) is a derivation at $x \in \mathcal{M}$; and given a derivation $\xi$ at $x$, there exists a curve $c$ on $\mathcal{M}$ with $c(0) = x$ such that $\xi$ is given by (2.1). The curve $c$ is said to *realize* the tangent vector $\xi$.

**Definition 2.8** (tangent bundle)**.** *Let $\mathcal{M}$ be a smooth manifold. The* tangent bundle*, noted* $\mathrm{T}\mathcal{M}$*, is the set:*

$$\mathrm{T}\mathcal{M} = \coprod_{x \in \mathcal{M}} \mathrm{T}_x\mathcal{M},$$

*where $\coprod$ stands for a disjoint union.*

**Definition 2.9** (vector field)**.** *A* vector field *$X$ is a smooth mapping from $\mathcal{M}$ to $\mathrm{T}\mathcal{M}$ that assigns to each point $x \in \mathcal{M}$ a tangent vector $X_x \in \mathrm{T}_x\mathcal{M}$. The set of vector fields on $\mathcal{M}$ is denoted as $\mathfrak{X}(\mathcal{M})$.*

Let $F \colon \mathcal{M} \to \mathcal{N}$ be a smooth mapping between two manifolds (Definition 2.5) and let $\xi$ be a tangent vector at $x \in \mathcal{M}$. It can be shown that the mapping $\mathrm{D}F(x)[\xi]$ from $\mathfrak{F}(\mathcal{N})$ to $\mathbb{R}$ defined by

$$(\mathrm{D}F(x)[\xi])f := \xi(f \circ F)$$

is a tangent vector to $\mathcal{N}$ at $F(x)$. The tangent vector is realized by $F \circ c$, where $c$ is any curve that realizes $\xi$. The mapping

$$\mathrm{D}F(x) \colon \mathrm{T}_x\mathcal{M} \to \mathrm{T}_{F(x)}\mathcal{N} \colon \mathrm{D}F(x)[\xi] \tag{2.2}$$

is a linear mapping called the *differential* (or *tangent map*) of $F$ at $x$.

### 2.2.1 Embedded submanifolds

We introduce embedded submanifolds as a way to connect two manifolds in situations where one is a subset of the other. This is useful as many manifolds encountered in practice are embedded in a matrix space such as $\mathbb{R}^{n\times s}$. Let $(\overline{\mathcal{M}}, \mathcal{A}^+)$ and $(\mathcal{M}, \mathcal{B}^+)$ be manifolds such that $\mathcal{M} \subset \overline{\mathcal{M}}$. If the manifold topology on $\mathcal{M}$ coincides with its subspace topology induced from $\overline{\mathcal{M}}$, we say that $\mathcal{M}$ is an *embedded submanifold* of $\overline{\mathcal{M}}$, and $\overline{\mathcal{M}}$ is called the embedding space. Importantly, $\mathcal{M}$ admits at most one differentiable structure which makes it an embedded submanifold of $\overline{\mathcal{M}}$. In that case, a smooth map on $\overline{\mathcal{M}}$, when restricted to $\mathcal{M}$, is also a smooth map. When the set $\overline{\mathcal{M}}$ is a Euclidean space, there is a convenient definition of submanifolds in terms of characteristic functions.

**Theorem 2.1.** *Let $\mathcal{E}$ be a Euclidean space of dimension $n$ and $\mathcal{M}$ be a subset of $\mathcal{E}$. Statements (1) and (2) below are equivalent:*

1. *$\mathcal{M}$ is a smooth embedded manifold of $\mathcal{E}$ of dimension $n - m$;*

2. *For all $x \in \mathcal{M}$, there is an open set $V$ of $\mathcal{E}$ containing $x$ and a smooth function $h\colon V \to \mathbb{R}^m$ such that the differential $\mathrm{D}h(x)\colon \mathcal{E} \to \mathbb{R}^m$ has rank $m$ and $V \cap \mathcal{M} = h^{-1}(0)$.*

*Furthermore, the tangent space at $x$ is given by $\mathrm{T}_x\mathcal{M} = \ker \mathrm{D}h(x)$, the null space of $\mathrm{D}h(x)$.*

### 2.2.2 Quotient manifolds

Let $\overline{\mathcal{M}}$ be a manifold with an equivalence relation $\sim$ on that set. Let $\overline{x} \in \overline{\mathcal{M}}$ and consider the equivalence class,

$$[\overline{x}] = \left\{\overline{y} \in \overline{\mathcal{M}} : \overline{y} \sim \overline{x}\right\}.$$

The equivalence relation induces a quotient space, that is, the set of equivalence classes

$$\mathcal{M} = \overline{\mathcal{M}}/\sim\, = \left\{[\overline{x}] : \overline{x} \in \overline{\mathcal{M}}\right\}.$$

The map $\pi\colon \overline{\mathcal{M}} \to \mathcal{M} : \pi(\overline{x}) = [\overline{x}]$ is called the *canonical projection.* Clearly, $\pi(\overline{x}) = \pi(\overline{y})$ if and only if $\overline{x} \sim \overline{y}$. Hence, we have $[\overline{x}] = \pi^{-1}(x)$ where $x = \pi(\overline{x}) \in \mathcal{M}$. We use $\pi(\overline{x})$ to denote $[\overline{x}]$ viewed as a point on $\mathcal{M}$ and $\pi^{-1}(x)$ with $x = \pi(\overline{x})$ to denote $[\overline{x}]$ viewed as a subset of $\overline{\mathcal{M}}$. The space $\mathcal{M}$ may in general admit several smooth manifold structures. Under certain conditions, there exists a unique manifold structure on $\mathcal{M}$ that turns it into a *quotient manifold* of $\overline{\mathcal{M}}$, the precise definition of which can be found in (Absil et al., 2008, Section 3.4). The manifold $\overline{\mathcal{M}}$ is called the total space of the quotient $\mathcal{M}$. Importantly, equivalence classes $[\overline{x}] \subset \overline{\mathcal{M}}$ are embedded submanifolds of $\overline{\mathcal{M}}$. In numerical algorithms, a class $x \in \mathcal{M}$ is represented on a computer by some $\overline{x} \in \overline{\mathcal{M}}$, an arbitrary member of the class.

Consider a scalar field $f\colon \mathcal{M} \to \mathbb{R}$ and some $x = [\overline{x}] \in \mathcal{M}$. A tangent vector, $\xi \in \mathrm{T}_x\mathcal{M}$, though well defined as the tangent vector to an equivalence class, is impractical to represent. Fortunately, it is conveniently represented by $\overline{\xi} \in \mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$, a tangent vector to $\overline{\mathcal{M}}$ at $\overline{x}$. It turns out that $\overline{\xi}$ will have the same properties as $\xi$ in terms of directional derivatives. Given $\xi \in \mathrm{T}_x\mathcal{M}$, we wish to find $\overline{\xi} \in \mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$ such that, for all $f\colon \mathcal{M} \to \mathbb{R}$,

$$\mathrm{D}f(x)[\xi] = \mathrm{D}\overline{f}(\overline{x})[\overline{\xi}]$$

where $\overline{f} = f \circ \pi\colon \overline{\mathcal{M}} \to \mathbb{R}$. Unfortunately, this representation is not unique. This is in part due to the fact that the dimension of $\mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$ is greater than that of $\mathrm{T}_x\mathcal{M}$. We need to use the quotient manifold structure to identify a unique vector $\overline{\xi}$ to represent $\xi$. Recall that equivalence classes $\pi^{-1}(x) = [\overline{x}]$ are embedded submanifolds of $\overline{\mathcal{M}}$. The tangent space to the submanifold $\pi^{-1}(x) = [\overline{x}]$ at $\overline{x} \in \overline{\mathcal{M}}$, namely, $\mathrm{T}_{\overline{x}}(\pi^{-1}(x))$, is called the *vertical space* at $\overline{x}$:

$$\mathrm{V}_{\overline{x}} := \mathrm{T}_{\overline{x}}\left(\pi^{-1}(x)\right) \subset \mathrm{T}_{\overline{x}}\overline{\mathcal{M}},$$

where $\pi^{-1}(x)$ is viewed as an embedded submanifold of $\overline{\mathcal{M}}$ and the inclusion $\mathrm{T}_{\overline{x}}(\pi^{-1}(x)) \subset \mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$ comes as a consequence.

A mapping H that assigns to each element $\overline{x}$ of $\overline{\mathcal{M}}$ a subspace $\mathrm{H}_{\overline{x}}$ of $\mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$ such that $\mathrm{V}_{\overline{x}} \oplus \mathrm{H}_{\overline{x}} = \mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$ is called a horizontal distribution on $\overline{\mathcal{M}}$, where $\oplus$ denotes the direct sum of two subspaces. The complementary space $\mathrm{H}_{\overline{x}} \subset \mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$ is called the *horizontal space* at $\overline{x}$. Intuitively, the tangent space $\mathrm{T}_{\overline{x}}\overline{\mathcal{M}}$ is composed of the vertical space $V_{\overline{x}}$, whose vectors point in directions that "remain" on the submanifold $[\overline{x}] \subset \overline{\mathcal{M}}$ where $f$ is constant, and the horizontal space $H_{\overline{x}}$ whose vectors point "outside" of the class $[\overline{x}] \subset \overline{\mathcal{M}}$, in directions where $f$ varies ; as illustrated in Figure 2.1. Let $\overline{\mathcal{M}}$ be endowed with a horizontal distribution, then there exists one and only one element $\overline{\xi} \in \mathrm{H}_{\overline{x}}$, called the *horizontal lift* of $\xi$ at $\overline{x}$, that satisfies $\mathrm{D}\pi(\overline{x})[\overline{\xi}] = \xi$. As a consequence, for all scalar fields $f\colon \mathcal{M} \to \mathbb{R}$,

$$\mathrm{D}\overline{f}(\overline{x})[\overline{\xi}] = \mathrm{D}f(\pi(\overline{x}))\left[\mathrm{D}\pi(\overline{x})[\overline{\xi}]\right] = \mathrm{D}f(x)[\xi].$$

The possibly ambiguity in the choice of an horizontal distribution is not usually of any practical concern, as the Riemannian metric introduced in the next section naturally selects a suitable horizontal distribution. The horizontal distribution that stems from the choice of a Riemannian metric satisfies the property that a vector field $V$ on a quotient manifold $\mathcal{M}$ is smooth if and only if its horizontal lift $\overline{V}$ is smooth on $\overline{\mathcal{M}}$ (Boumal, 2020, Thm. 9.26).

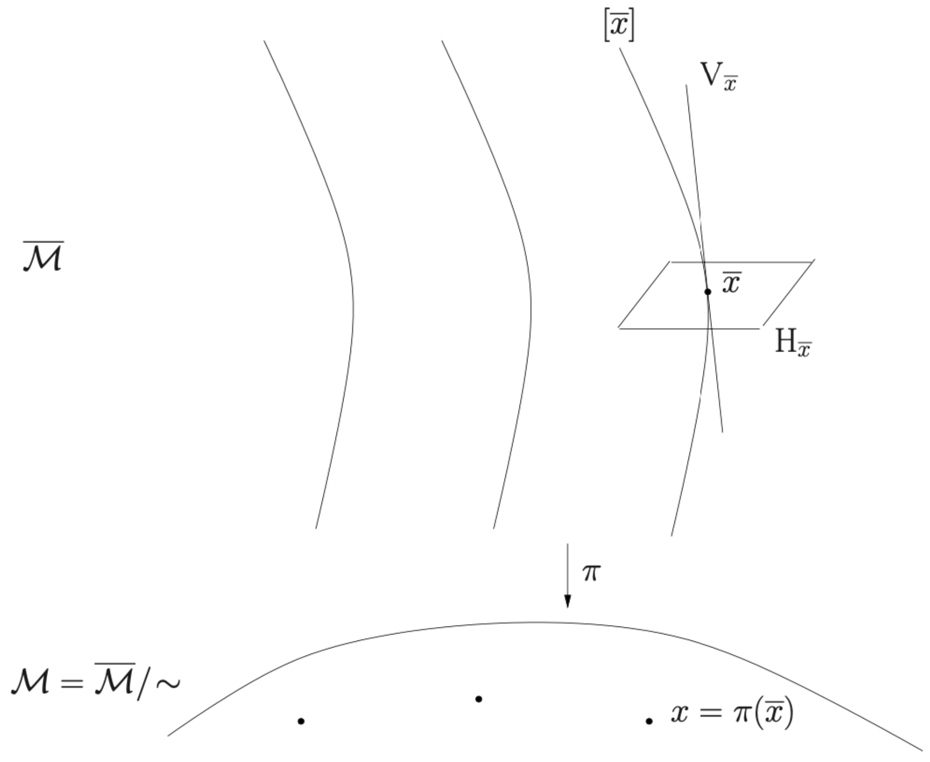

$[\overline{x}]$
$\mathrm{V}_{\overline{x}}$
$\overline{\mathcal{M}}$
$\overline{x}$
$\mathrm{H}_{\overline{x}}$
$\pi$
$\mathcal{M} = \overline{\mathcal{M}}/\sim$
$x = \pi(\overline{x})$

Figure 2.1: A quotient manifold (Absil et al., 2008)

## 2.3 Riemannian structure and gradients

In order to define optimization algorithms on manifolds, it is necessary to have a notion of length of a tangent vector. This is addressed by considering Riemannian manifolds, whose tangent spaces are associated with a smoothly varying inner product.

**Definition 2.10.** *Let $\mathcal{M}$ be a smooth manifold and take $x \in \mathcal{M}$. An inner product $\langle \cdot, \cdot \rangle_x$ on $\mathrm{T}_x\mathcal{M}$ is a bilinear, symmetric and positive-definite form on $\mathrm{T}_x\mathcal{M}$, i.e., for all $\xi, \zeta, \eta \in \mathrm{T}_x\mathcal{M}, a, b \in \mathbb{R}$:*

- $\langle a\xi + b\zeta, \eta \rangle_x = a \langle \xi, \eta \rangle_x + b \langle \zeta, \eta \rangle_x$,
- $\langle \xi, \zeta \rangle_x = \langle \zeta, \xi \rangle_x$, *and*
- $\langle \xi, \xi \rangle_x \geq 0$, *with* $\langle \xi, \xi \rangle_x \Leftrightarrow \xi = 0$.

*An inner product induces a norm on the tangent space. The norm of a tangent vector $\xi \in \mathrm{T}_x\mathcal{M}$ is $\|\xi\|_x = \sqrt{\langle \xi, \xi \rangle_x}$.*

A smoothly varying inner product is called a Riemannian metric and the subscript is often omitted when no confusion is possible, $\langle \xi, \eta \rangle_x = \langle \xi, \eta \rangle$; this also applies to the associated norm $\|\xi\|_x = \|\xi\|$.

**Definition 2.11** (Riemannian manifold)**.** *A* Riemannian manifold *is a pair* $(\mathcal{M}, g)$*, where* $\mathcal{M}$ *is a smooth manifold and* $g$ *is a Riemannian metric. A* Riemannian metric *is a smoothly varying inner product defined on the tangent spaces of* $\mathcal{M}$*, that is, for each* $x \in \mathcal{M}$*,* $g_x(\cdot,\cdot) = \langle \cdot,\cdot \rangle_x$ *is an inner product on* $\mathrm{T}_x\mathcal{M}$*.*

In this definition, *smoothly varying* is understood in the sense that, for any vector fields $X, Y \in \mathfrak{X}(\mathcal{M})$ the map $x \mapsto \langle X_x, Y_x \rangle$ is a smooth function from $\mathcal{M}$ to $\mathbb{R}$. A vector space with an inner product is a particular Riemannian manifold called a *Euclidean space.* When the metric is clear from context, we will often write $\mathcal{M}$ to refer to the Riemannian manifold $(\mathcal{M}, g)$. We introduce the Riemannian gradient of a real-valued function on a Riemannian manifold, a notion that is central to optimization algorithms.

**Definition 2.12** (gradient)**.** *Let* $f\colon \mathcal{M} \to \mathbb{R}$ *be a scalar field on a Riemannian manifold* $\mathcal{M}$*. The (Riemannian) gradient of* $f$ *at* $x \in \mathcal{M}$*, written* $\mathrm{grad}f(x)$*, is defined as the unique element of* $\mathrm{T}_x\mathcal{M}$ *that satisfies*

$$\mathrm{D}f(x)[\xi] = \langle \mathrm{grad}f(x), \xi \rangle_x \textit{ for all } \xi \in \mathrm{T}_x\mathcal{M}.$$

*Thus* $\mathrm{grad}f\colon \mathcal{M} \to \mathrm{T}\mathcal{M}$ *is a vector field on* $\mathcal{M}$*.*

For scalar fields defined on Euclidean spaces, $\mathrm{grad}f$ is the usual gradient, denoted by $\nabla f$. The intuition that the gradient represents the direction of steepest ascent carries over to Riemannian manifolds. Indeed,

$$\max_{\substack{\xi \in \mathrm{T}_x\mathcal{M} \\ \|\xi\|_x = 1}} \mathrm{D}f(x)[\xi] = \|\mathrm{grad}f(x)\|_x\,,$$

and the maximum is reached for $\xi = \mathrm{grad}f(x) / \|\mathrm{grad}f(x)\|_x$.

### 2.3.1 Riemannian submanifolds

Let $\overline{\mathcal{M}}$ be a Riemannian manifold. If $\mathcal{M} \subset \overline{\mathcal{M}}$ is an embedded submanifold of $\overline{\mathcal{M}}$, it can be endowed with a Riemannian metric simply by restricting the metric of $\overline{\mathcal{M}}$ to the tangent spaces of $\mathcal{M}$.

**Definition 2.13** (Riemannian submanifold)**.** *Let* $(\overline{\mathcal{M}}, \overline{g})$ *be a Riemannian manifold and let* $(\mathcal{M}, g)$ *be such that* $\mathcal{M}$ *is a submanifold of* $\overline{\mathcal{M}}$ *and such that* $g$ *is the restriction of* $\overline{g}$ *to the tangent spaces of* $\mathcal{M}$*. More precisely, for all* $x \in \mathcal{M}$ *and for all tangent vectors* $\xi, \eta \in \mathrm{T}_x\mathcal{M} \subset \mathrm{T}_x\overline{\mathcal{M}}$*, the metrics* $g$ *and* $\overline{g}$ *are compatible in the sense that* $g_x(\xi,\eta) = \overline{g}_x(\xi,\eta)$*. Then,* $\mathcal{M}$ *is a* Riemannian submanifold *of* $\overline{\mathcal{M}}$*.*

Since $\mathrm{T}_x\mathcal{M} \subset \mathrm{T}_x\overline{\mathcal{M}}$, it is natural to define the orthogonal complement of $\mathrm{T}_x\mathcal{M}$ with respect to the metric $\overline{g}$. This gives the normal space

$$\mathrm{T}_x^{\perp}\mathcal{M} = \left\{\xi \in \mathrm{T}_x\overline{\mathcal{M}} : \langle \xi, \eta \rangle = 0 \text{ for all } \eta \in \mathrm{T}_x\mathcal{M}\right\}.$$

All vectors of $\mathrm{T}_x\overline{\mathcal{M}}$ are uniquely decomposed as $\xi = \mathrm{Proj}_x(\xi) + \mathrm{Proj}_x^{\perp}(\xi)$ where $\mathrm{Proj}\colon \mathrm{T}_x\overline{\mathcal{M}} \to \mathrm{T}_x\mathcal{M}$ and $\mathrm{Proj}^{\perp}\colon \mathrm{T}_x\overline{\mathcal{M}} \to \mathrm{T}_x^{\perp}\mathcal{M}$ are orthogonal projections. Let $\mathcal{M}$ be a Riemannian submanifold of $\overline{\mathcal{M}}$. Consider $\overline{f}\colon \overline{\mathcal{M}} \to \mathbb{R}$ and $f\colon \mathcal{M} \to \mathbb{R}$, its restriction to $\mathcal{M}$. Then,

$$\mathrm{grad} f(x) = \mathrm{Proj}_x\left(\mathrm{grad}\overline{f}(x)\right). \tag{2.3}$$

Indeed, for any $\xi \in \mathrm{T}_x\mathcal{M}$,

$$\begin{aligned}
\mathrm{D}f(x)[\xi] &= \mathrm{D}\overline{f}(x)[\xi] \\
&= \left\langle \mathrm{grad}\overline{f}(x), \xi \right\rangle \\
&= \left\langle \mathrm{Proj}_x(\mathrm{grad}\overline{f}(x)) + \mathrm{Proj}_x^{\perp}(\mathrm{grad}\overline{f}(x)), \xi \right\rangle \\
&= \left\langle \mathrm{Proj}_x(\mathrm{grad}\overline{f}(x)), \xi \right\rangle.
\end{aligned}$$

In particular, if the embedding manifold $\overline{\mathcal{M}}$ is a Euclidean space (such as $\mathbb{R}^n$), the Riemannian gradient is the projection of the Euclidean gradient on the tangent space:

$$\mathrm{grad} f(x) = \mathrm{Proj}_x\left(\nabla f(x)\right),$$

where $\nabla$ denotes the classical (Euclidean) gradient in $\mathbb{R}^n$.

### 2.3.2 Riemannian quotient manifolds

Let $(\overline{\mathcal{M}}, \overline{g})$ be a Riemannian manifold and let $\mathcal{M} = \overline{\mathcal{M}}/\sim$ be a quotient manifold of $\overline{\mathcal{M}}$. We use the Riemannian structure on $\overline{\mathcal{M}}$ to equip $\mathcal{M}$ with a Riemannian structure as well. The Riemannian metric on $\overline{\mathcal{M}}$ allows to single out a horizontal distribution. For all $\overline{x} \in \overline{\mathcal{M}}$,

$$\mathrm{H}_{\overline{x}} := \mathrm{V}_{\overline{x}}^{\perp} = \left\{\overline{\xi} \in \mathrm{T}_{\overline{x}}\overline{\mathcal{M}} : \overline{g}_{\overline{x}}(\overline{\xi}, \overline{\eta}) = 0 \text{ for all } \overline{\eta} \in \mathrm{V}_{\overline{x}}\right\}.$$

The map defined by

$$g_x(\eta, \xi) := \overline{g}_{\overline{x}}(\overline{\xi}, \overline{\eta})$$

is a Riemannian metric on $\mathrm{T}_x\mathcal{M}$ if $\overline{g}_{\overline{x}}(\overline{\xi}, \overline{\eta})$ does not depend on the choice of $\overline{x}$ to represent $x \in \mathcal{M}$. We now explain how to compute a horizontal lift of $\mathrm{grad} f(x)$. Choose any scalar field $\overline{f}$ on $\overline{\mathcal{M}}$ such that $\overline{f} = f \circ \pi$, where $\pi$ is the canonical projection. Notice that the derivative of $\overline{f}$ along vertical vectors is zero. Indeed, for any $\overline{\xi} \in \mathrm{V}_{\overline{x}}$,

$$\mathrm{D}\overline{f}(\overline{x})[\overline{\xi}] = \mathrm{D}(f \circ \pi)(\overline{x})[\overline{\xi}] = \mathrm{D}f(\pi(\overline{x}))\left[\mathrm{D}\pi(\overline{x})[\overline{\xi}]\right] = \mathrm{D}f(x)[0] = 0.$$

As a consequence, for all $\overline{x} \in \overline{\mathcal{M}}$, $\mathrm{grad}\overline{f}(\overline{x}) \in \mathrm{H}_{\overline{x}}$. This horizontal vector is the horizontal lift of the gradient of $f$ at $x$:

$$\overline{\mathrm{grad} f(x)} = \mathrm{grad}\overline{f}(\overline{x}), \tag{2.4}$$

where the left hand-side denotes the horizontal lift of $\mathrm{grad} f(x)$ at $\overline{x}$. Indeed, for all $x \in \mathcal{M}$ and $\xi \in \mathrm{T}_x\mathcal{M}$ and for any $\overline{x} \in \pi^{-1}(x)$,

$$\begin{aligned} g_x(\mathrm{grad} f(x), \xi) &= g_x\left(\mathrm{D}\pi(\overline{x})[\overline{\mathrm{grad} f(x)}], \mathrm{D}\pi(\overline{x})[\overline{\xi}]\right) \\ &= g_x\left(\mathrm{D}\pi(\overline{x})[\mathrm{grad}\overline{f}(\overline{x})], \mathrm{D}\pi(\overline{x})[\overline{\xi}]\right) \\ &= \overline{g}_{\overline{x}}\left(\mathrm{grad}\overline{f}(\overline{x}), \overline{\xi}\right) \\ &= \mathrm{D}\overline{f}(\overline{x})[\overline{\xi}] = \mathrm{D}f(x)[\xi]. \end{aligned}$$

The orthogonal projection onto the horizontal space at $\overline{x}$ is denoted by $\mathrm{Proj}^h_{\overline{x}} \colon \mathrm{T}_{\overline{x}}\overline{\mathcal{M}} \to H_{\overline{x}}$.

## 2.4 Connections and Hessians

We would like to define the second derivative of a smooth function $f \colon \mathcal{M} \to \mathbb{R}$ on a Riemannian manifold $\mathcal{M}$. If $f$ were defined on a Euclidean space, the directional derivative of the gradient in some direction $\xi$ would we given by

$$\mathrm{D}(\mathrm{grad} f(x))[\xi] = \lim_{t \to 0} \frac{\mathrm{grad} f(x + t\xi) - \mathrm{grad} f(x)}{t}.$$

This definition is not applicable on manifolds for two reasons. The expression $x + t\xi$ is undefined, since $\mathcal{M}$ is not a vector space. Additionally, gradients at different points on the manifold belong to different tangent spaces, and their difference is also undefined. To resolve these issues, the concept of affine connection is introduced.

**Definition 2.14** (affine connection). *Let $\mathfrak{X}(\mathcal{M})$ denote the set of smooth vector fields on $\mathcal{M}$ and $\mathfrak{F}(\mathcal{M})$ denote the set of smooth scalar fields on $\mathcal{M}$. An* affine connection $\nabla$ *on a manifold $\mathcal{M}$ is a mapping*

$$\nabla \colon \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M}) \colon (X, Y) \mapsto \nabla_X Y$$

*which satisfies the following properties:*

1. *$\mathfrak{F}(\mathcal{M})$-linearity in $X$: $\nabla_{fX+gY} Z = f\nabla_X Z + g\nabla_Y Z$,*
2. *$\mathbb{R}$-linearity in $Y$: $\nabla_X(aY + bZ) = a\nabla_X Y + b\nabla_X Z$,*
3. *Product rule (Leibniz's law): $\nabla_X(fY) = (Xf)Y + f\nabla_X Y$,*

*in which $X, Y, Z \in \mathfrak{X}(\mathcal{M})$, $f, g \in \mathfrak{F}(\mathcal{M})$ and $a, b \in \mathbb{R}$.*

In the definition above, vector fields are used as derivation objects, as described in Section 2.2, so that $Xf(x) = \mathrm{D}f(x)[X_x]$. The symbol $\nabla$ in this context denotes an affine connection and not the Euclidean gradient of a scalar-valued function, which uses the same symbol.

**Definition 2.15** (covariant derivative)**.** *The vector field $\nabla_X Y$ is called the covariant derivative of $Y$ with respect to $X$ for the affine connection $\nabla$. Since $(\nabla_X Y)_x$ only depends on $X$ through $X_x$, we can make sense of the notation $\nabla_\xi Y$ where $\xi \in \mathrm{T}_x\mathcal{M}$. It is interpreted as $(\nabla_\xi Y)_x = (\nabla_X Y)_x$ for any vector field $X$ such that $X_x = \xi$.*

At each point $x \in \mathcal{M}$, the vector field $(\nabla_X Y)_x$ captures how $Y$ varies at $x$ along the direction $X_x$. The Levi-Civita theorem singles out one particular affine connection for each Riemannian manifold.

**Theorem 2.2** (Levi-Civita)**.** *On a Riemannian manifold $\mathcal{M}$, there exists a unique affine connection $\nabla$ that satisfies*

1. $\nabla_X Y - \nabla_Y X = [X, Y]$ *(symmetry), and*

2. $Z \langle X, Y \rangle = \langle \nabla_Z X, Y \rangle + \langle X, \nabla_Z Y \rangle$ *(compatibility with the Riemannian metric),*

*for all $X, Y, Z \in \mathfrak{X}(\mathcal{M})$. This affine connection is called the* Levi-Civita connection *or the* Riemannian connection*.*

In the above definition, we used the notation $[X, Y]$ for the Lie bracket of $X$ and $Y$, which is a vector defined by $[X, Y]f = X(Yf) - Y(Xf)$, for all $f \in \mathfrak{F}(\mathcal{M})$. Since affine connections provide a notion of derivative of a vector field, we can use them to define the Hessian of a scalar field as the derivative of the gradient vector field. On Riemannian manifolds, the Levi-Civita connection defines the Riemannian Hessian.

**Definition 2.16** (Riemannian Hessian)**.** *Let $f$ be a scalar field on a Riemannian manifold $\mathcal{M}$ equipped with the Riemannian connection $\nabla$, the* Riemannian Hessian *of $f$ at $x \in \mathcal{M}$ is the linear map $\mathrm{Hess}f(x) \colon \mathrm{T}_x\mathcal{M} \to \mathrm{T}_x\mathcal{M}$ defined by*

$$\mathrm{Hess}f(x)[\xi] = \left(\nabla_\xi \mathrm{grad} f\right)_x = (\nabla_X \mathrm{grad} f)_x,$$

*where $X$ is any vector field on $\mathcal{M}$ such that $X_x = \xi$.*

We now particularize the Riemannian Hessian to Riemannian submanifolds and Riemannian quotient manifolds.

### 2.4.1 Riemannian submanifolds

The Riemannian connection on a submanifold of a Riemannian manifold is obtained by a projection onto the tangent space of the connection on the embedding manifold.

**Theorem 2.3.** *Let $\mathcal{M}$ be a Riemannian submanifold of a Riemannian manifold $\overline{\mathcal{M}}$ and let $\nabla$ and $\overline{\nabla}$ denote the Riemannian connections on $\mathcal{M}$ and $\overline{\mathcal{M}}$. Then,*

$$(\nabla_X Y)_x = \mathrm{Proj}_x(\overline{\nabla}_X Y)_x$$

*for all $X, Y \in \mathfrak{X}(\mathcal{M})$.*

In particular, if $\overline{\mathcal{M}}$ is a Euclidean space, then

$$(\nabla_X Y)_x = \mathrm{Proj}_x\left(\mathrm{D}Y(x)[X_x]\right). \tag{2.5}$$

This means that the Riemannian connection on $\mathcal{M}$ can be computed by a classical derivative in the Euclidean space followed by a projection onto the tangent space. Since $\mathrm{grad} f(x) = \mathrm{Proj}_x \nabla f(x)$, where $\nabla f(x)$ is the classical gradient of $f$ seen as a scalar field on the embedding Euclidean space (Equation (2.3)), the Riemannian Hessian is given by

$$\mathrm{Hess} f(x)[\xi] = \mathrm{Proj}_x\left(\mathrm{D}(x \mapsto \mathrm{Proj}_x \nabla f(x))[\xi]\right). \tag{2.6}$$

Effectively, it requires to compute the classical gradient $\nabla f(x)$, project it on the tangent space, compute the directional derivative of the result and project once more.

### 2.4.2 Riemannian quotient manifolds

The next theorem expresses the link between the Riemannian connection of a Riemannian manifold and the Riemannian connection of one of its Riemannian quotient manifolds.

**Theorem 2.4.** *Let $\overline{\mathcal{M}}$ be a Riemannian manifold and $\mathcal{M} = \overline{\mathcal{M}}/\sim$ be a Riemannian quotient manifold of $\overline{\mathcal{M}}$. Let $\nabla$ and $\overline{\nabla}$ be the Riemannian connections on $M$ and $\overline{\mathcal{M}}$, respectively. Then,*

$$\overline{(\nabla_X Y)_x} = \mathrm{Proj}^h_{\overline{x}}\left(\overline{\nabla}_{\overline{X}} \overline{Y}\right)_{\overline{x}}$$

*for all $X, Y \in \mathfrak{X}(\mathcal{M}), x \in \mathcal{M}$ and any $\overline{x} \in \pi^{-1}(x)$. Overlines denote horizontal lifts and* $\mathrm{Proj}^h_{\overline{x}}$ *is the orthogonal projection onto the horizontal space at $\overline{x} \in \overline{\mathcal{M}}$.*

In the particular case where $\overline{\mathcal{M}}$ is a Euclidean space, this reduces to

$$\overline{(\nabla_X Y)_x} = \mathrm{Proj}^h_{\overline{x}}\left(\mathrm{D}\overline{Y}(\overline{x})[\overline{X}_{\overline{x}}]\right),$$

that is, a classical directional derivative of the horizontal vector field $\overline{Y}$ followed by a projection onto the horizontal space. This allows to write the Riemannian Hessian of a function $f\colon \mathcal{M} \to \mathbb{R}$ as

$$\mathrm{Hess}f(x)[\xi] = \mathrm{Proj}^h_{\overline{x}}\left(\mathrm{D}\overline{\nabla f}(\overline{x})[\overline{\xi}]\right), \tag{2.7}$$

where $\overline{\nabla f}$ is the classical gradient of $f$ seen as a function on the total space $\overline{\mathcal{M}}$ (which is naturally a horizontal vector field, since $f$ is invariant along vertical directions).

## 2.5 Optimality conditions

Consider the optimization problem

$$\min_x f(x) \text{ subject to } x \in \mathcal{M}, \tag{2.8}$$

where $f\colon \mathcal{M} \to \mathbb{R}$ is smooth and $\mathcal{M}$ is a Riemannian manifold. We state necessary optimality conditions of first- and second-order for local minimizers of (2.8). A proof of the next two results can be found in (Boumal, 2020, Section 4.2 and 6.1).

**Proposition 2.5** (First-order critical point on manifolds)**.** *If $x \in \mathcal{M}$ is a local minimizer for Problem* (2.8)*, then*

$$\mathrm{grad}f(x) = 0. \tag{2.9}$$

**Proposition 2.6** (Second-order critical point on manifolds)**.** *If $x \in \mathcal{M}$ is a local minimizer for Problem* (2.8)*, then*

$$\mathrm{grad}f(x) = 0 \qquad \textit{and} \qquad \mathrm{Hess}f(x) \succeq 0. \tag{2.10}$$

Implementations of Riemannian optimization methods usually aim to find approximate critical points, that is, points on the manifold which satisfy first- or second-order conditions up to a predefined tolerance. Approximate first- and second-order critical points on manifolds are defined as follows.

**Definition 2.17** (Approximate first-order critical points on manifolds)**.** *The point $x \in \mathcal{M}$ is an $\varepsilon_1$–first-order critical point of $f$ on $\mathcal{M}$ if*

$$\|\mathrm{grad}f(x)\|_x \leq \varepsilon_1. \tag{2.11}$$

**Definition 2.18** (Approximate second-order critical points on manifolds)**.** *The point $x \in \mathcal{M}$ is an $(\varepsilon_1, \varepsilon_2)$–second-order critical point of $f$ on $\mathcal{M}$ if*

$$\|\mathrm{grad}f(x)\|_x \leq \varepsilon_1 \qquad \textit{and} \qquad \mathrm{Hess}f(x) \succeq -\varepsilon_2 \,\mathrm{Id}\,. \tag{2.12}$$

In the definitions above, the norm is induced by the Riemannian metric on $\mathcal{M}$.

## 2.6 Retractions

In $\mathbb{R}^n$, it is straightforward to travel from a point in a given direction. Riemannian optimization algorithms aim to travel from a given point in a direction prescribed by a tangent vector. This requires the use of a mapping, called a *retraction*, which returns a point on the manifold in the direction of a tangent vector. To arrive at the definition of retractions, we take a small detour to define geodesics and the exponential map.

A geodesic is a curve $\gamma\colon \mathbb{R} \to \mathcal{M}$ with zero acceleration, as defined in (Absil et al., 2008, Section 5.4). The notion of acceleration is defined by the connection $\nabla$ on $\mathcal{M}$, hence geodesics depend on the choice of connection. The exponential map travels along geodesics on the manifold.

**Definition 2.19** (Exponential map)**.** *Let $\mathcal{M}$ be a manifold endowed with a connection $\nabla$ and let $x \in \mathcal{M}$. For every $\xi \in \mathrm{T}_x\mathcal{M}$, there exists an open interval $I \ni 0$ and a geodesic $\gamma(t; x, \xi)\colon I \to \mathcal{M}$ such that $\gamma(0) = x$ and $\dot{\gamma}(0) = \xi$. Moreover, we have the homogeneity property $\gamma(t; x, a\xi) = \gamma(at; x, \xi)$. The mapping*

$$\mathrm{Exp}_x\colon \mathrm{T}_x\mathcal{M} \to \mathcal{M}\colon \xi \mapsto \mathrm{Exp}_x(\xi) = \gamma(1; x, \xi)$$

*is called the* exponential map *at $x$. In particular, $\mathrm{Exp}_x(0) = x$, for all $x \in \mathcal{M}$.*

We note the definition of injectivity radius of a manifold, which we come across in Section 4.5.

**Definition 2.20** ((Boumal, 2020) Definition 10.16)**.** *The* injectivity radius *of a Riemannian manifold $\mathcal{M}$ at a point $x$, denoted by $\mathrm{inj}(x)$, is the supremum over radii $r > 0$ such that $\mathrm{Exp}_x$ is defined and is a diffeomorphism on the open ball*

$$B(x, r) = \{v \in \mathrm{T}_x\mathcal{M} : \|v\|_x < r\}.$$

*By the inverse function theorem, $\mathrm{inj}(x) > 0$.*

Geodesics are not readily computable on most manifolds, hence, the exponential map is usually approximated in optimization algorithms. At a given point $x \in \mathcal{M}$, a retraction $\mathrm{R}_x\colon \mathrm{T}_x\mathcal{M} \to \mathcal{M}$ is a first-order approximation of the exponential map.

**Definition 2.21** (Retraction (Absil et al., 2008))**.** *A retraction on a manifold $\mathcal{M}$ is a smooth mapping $\mathrm{R}$ from the tangent bundle $T\mathcal{M}$ onto $\mathcal{M}$ with the following properties. Let $\mathrm{R}_x$ denote the restriction of $\mathrm{R}$ to $\mathrm{T}_x\mathcal{M}$.*
*(i) $\mathrm{R}_x(0_x) = x$, where $0_x$ denotes the zero element of $T_x\mathcal{M}$.*
*(ii) With the canonical identification $\mathrm{T}_{0_x}\mathrm{T}_x\mathcal{M} \simeq \mathrm{T}_x\mathcal{M}$, $\mathrm{R}_x$ satisfies*

$$\mathrm{DR}_x(0_x) = \mathrm{Id}_{\mathrm{T}_x\mathcal{M}},$$

*where $\mathrm{Id}_{\mathrm{T}_x\mathcal{M}}$ denotes the identity mapping on $\mathrm{T}_x\mathcal{M}$.*

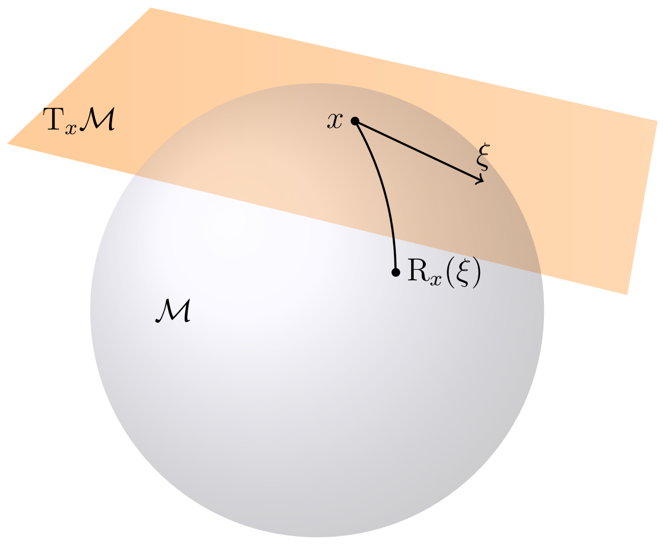


Figure 2.2: The retraction map

Retractions, illustrated in Figure 2.2, are designed to approximate the exponential map in a way that is cheap to compute but does not hinder convergence of algorithms. When the projection of a point from the ambient space to the manifold is available, it allows to define a *retraction by projection* (Absil and Malick, 2012), given for $\xi \in \mathrm{T}_x\mathcal{M}$ by

$$\mathrm{R}_x(\xi) = \mathrm{P}_{\mathcal{M}}(x + \xi).$$

For a given manifold, there may be several ways to define a retraction. One important aspect of Riemannian optimization is to compute retractions efficiently.

## 2.7 Parallel transport

In Section 4.5, we encounter the notion of parallel transport on a manifold, which we describe here by following (Boumal, 2020, Section 10.3). Consider a manifold $\mathcal{M}$ equipped with a connection $\nabla$ and a tangent vector $u \in \mathrm{T}_x\mathcal{M}$. In some situations, it is desirable to compare $u$ with vectors in the tangent space at a different point $y \in \mathcal{M}$. We describe how the vector $u$ may be transported along a curve in a meaningful way. Consider a smooth curve $c\colon I \to \mathcal{M}$ such that

$$c(0) = x \qquad\qquad c(1) = y.$$

Consider a smooth field on this curve, $Z \in \mathfrak{X}(c)$, with $Z(c(0)) = u$. Let $\frac{\mathrm{D}}{\mathrm{d}t}\colon \mathfrak{X}(c) \to \mathfrak{X}(c)$ denote the *covariant derivative on the curve* $c$ induced by the connection $\nabla$, as defined in (Boumal, 2020, Theorem 5.28). We require that $Z$ be *parallel* with respect to the affine connection $\nabla$. The upcoming result ensures that such vector field exists and is unique.

**Definition 2.22.** *A vector field $Z \in \mathfrak{X}(c)$ such that $\frac{\mathrm{D}}{\mathrm{d}t} Z = 0$ is parallel.*

**Theorem 2.7.** *On a manifold $\mathcal{M}$ with a connection and induced covariant derivative $\frac{\mathrm{D}}{\mathrm{d}t}$, for any smooth curve $c\colon I \to \mathcal{M}$, $t_0 \in I$ and $u \in \mathrm{T}_{c(t_0)}\mathcal{M}$, there exists a unique parallel vector field $Z \in \mathfrak{X}(c)$ such that $Z(c(t_0)) = u$.*

This justifies the following definition.

**Definition 2.23** (parallel transport)**.** *Given a smooth curve $c$ on $\mathcal{M}$, the* parallel transport *of tangent vectors at $c(t_0)$ to the tangent space at $c(t_1)$ along $c$,*

$$\mathrm{PT}^{c}_{t_1 \leftarrow t_0} \colon \mathrm{T}_{c(t_0)}\mathcal{M} \to \mathrm{T}_{c(t_1)}\mathcal{M}, \tag{2.13}$$

*is defined by $\mathrm{PT}^{c}_{t_1 \leftarrow t_0}(u) = Z(c(t_1))$, where $Z \in \mathfrak{X}(c)$ is the unique parallel vector field such that $Z(c(t_0)) = u$.*

## 2.8 Algorithms and implementations

We introduce some classical Riemannian optimization algorithms for the minimization of a smooth function $f\colon \mathcal{M} \to \mathbb{R}$, where $\mathcal{M}$ is a Riemannian manifold.

### 2.8.1 Riemannian gradient descent

From the notions of Riemannian gradient and retraction, we are able to define a simple Riemannian optimization algorithm, namely, Riemannian gradient descent (Algorithm 1). This algorithm uses the negative gradient as a search direction and a retraction to update the iterate while remaining on the manifold. In order to ensure convergence for an arbitrary $x_0 \in \mathcal{M}$, the algorithm must be augmented with a line search to compute the step sizes. We do not detail such procedure and, instead, choose to use trust-region methods.

**Algorithm 1** Riemannian gradient descent, (Absil et al., 2008)

```
1: Given: $f\colon \mathcal{M} \to \mathbb{R}$, a retraction R on $\mathcal{M}$, $x_0 \in \mathcal{M}$ and $\varepsilon > 0$.
2: $k = 0$
3: while $\|\mathrm{grad} f(x_k)\|_{x_k} > \varepsilon$ do
4:     $\eta_k = -\mathrm{grad} f(x_k) \in \mathrm{T}_{x_k}\mathcal{M}$
5:     Find step size $\alpha_k > 0$
6:     $x_{k+1} = \mathrm{R}_{x_k}(\alpha_k \eta_k)$
7:     $k = k + 1$
8: end while
```

### 2.8.2 Riemannian trust-region

Trust-region methods constitute a popular family of optimization algorithms that have been studied in great depths for unconstrained and constrained problems alike (Jorge Nocedal, 1999; Conn et al., 2000). In this section, we present the Riemannian version of trust-region methods.

At each iterate $x_k \in \mathcal{M}$, the Riemannian trust-region builds a local model of the function on the tangent space $\mathrm{T}_{x_k}\mathcal{M}$. The method minimizes this model under a ball constraint that prevents undesirably large steps, where the model does not accurately represent the function. The Riemannian trust-region subproblem takes the following form around $x_k \in \mathcal{M}$:

$$\min_{\substack{\eta \in \mathrm{T}_{x_k}\mathcal{M} \\ \|\eta\|_{x_k} \le \Delta_k}} \hat{m}_{x_k}(\eta) := f(x_k) + \langle \eta, \mathrm{grad} f(x_k) \rangle_{x_k} + \frac{1}{2}\langle \eta, H_k[\eta] \rangle_{x_k}, \tag{2.14}$$

where $H_k \colon \mathrm{T}_{x_k}\mathcal{M} \to \mathrm{T}_{x_k}\mathcal{M}$ is a symmetric operator on $\mathrm{T}_{x_k}\mathcal{M}$, $\Delta_k$ is the trust-region radius and the model $\hat{m}_{x_k} \colon \mathrm{T}_{x_k}\mathcal{M} \to \mathbb{R}$ is a quadratic approximation of the *pullback* $\hat{f}_{x_k} = f \circ \mathrm{R}_{x_k}$, defined on the tangent space at $x_k \in \mathcal{M}$ for some retraction $\mathrm{R}_{x_k} \colon \mathrm{T}_{x_k}\mathcal{M} \to \mathcal{M}$. The (usually approximate) minimization of the subproblem gives a candidate step $\eta_k$, whose quality is assessed using the following ratio:

$$\rho_k = \frac{f(x_k) - f(\mathrm{R}_{x_k}(\eta_k))}{\hat{m}_{x_k}(0_{x_k}) - \hat{m}_{x_k}(\eta_k)} \tag{2.15}$$

where $0_{x_k} \in \mathrm{T}_{x_k}\mathcal{M}$ is the zero vector of the tangent space at $x_k$. If $\rho_k$ is large enough, which indicates an agreement between the function decrease and the model decrease, the step $\eta_k$ is accepted and the next iterate is defined as $x_{k+1} = \mathrm{R}_{x_k}(\eta_k)$; otherwise the step is rejected and the trust-region radius is reduced. If $\rho_k$ is large enough and $\|\eta_k\|_{x_k} = \Delta_k$, which suggests that the model is a faithful representation of $f$, the trust-region radius may also be increased.

**First-order Riemannian trust-region** When the Hessian of the cost function is not available, or in order to obtain a cheap first-order method, the identity is used in the quadratic model, $H_k = \mathrm{Id}$. This gives a method similar to Riemannian gradient descent, where the trust-region is used to ensure global convergence, as opposed to a line search.

**Second-order Riemannian trust-region** When the true Hessian of the cost function is available, the classical second-order trust-region method is obtained with the choice $H_k = \mathrm{Hess} f(x_k)$. It is also possible to use an approximation of the true Hessian for $H_k$. Algorithm 2 defines a version of the Riemannian trust-region (RTR) which is designed to reach approximate second-order critical points, defined in Equation (2.10). A first-order version of the algorithm is given by the choice of parameter $\varepsilon_H = \infty$.

**Algorithm 2** Riemannian trust-region (RTR) (Boumal et al., 2019)

```
1: Given:  $x_0 \in \mathcal{M}$ and $0 < \Delta_0 < \bar{\Delta}$, $\varepsilon_g > 0$, $\varepsilon_H > 0$ and $0 < \rho' < 1/4$
2: Init:  $k = 0$
3: while true do
4:     if $\|\mathrm{grad} f(x_k)\|_{x_k} > \varepsilon_g$ then
5:         Obtain $\eta_k \in \mathrm{T}_{x_k}\mathcal{M}$ satisfying A5
6:     else if $\varepsilon_H < \infty$ then
7:         if $\lambda_{\min}(H_k) < -\varepsilon_H$ then
8:             Obtain $\eta_k \in \mathrm{T}_{x_k}\mathcal{M}$ satisfying A6
9:         else
10:            return $x_k$
11:        end if
12:    else
13:        return $x_k$
14:    end if
15:    $x_k^+ = \mathrm{R}_{x_k}(\eta_k)$
16:    $\rho = f(x_k) - f(x_k^+)/(\hat{m}_{x_k}(0) - \hat{m}_{x_k}(\eta_k))$
17:    if $\rho < 1/4$ then
18:        $\Delta_{k+1} = \Delta_k/4$
19:    else if $\rho > 3/4$ and $\|\eta_k\|_{x_k} = \Delta_k$ then
20:        $\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta})$
21:    else
22:        $\Delta_{k+1} = \Delta_k$
23:    end if
24:    if $\rho > \rho'$ then
25:        $x_{k+1} = x_k^+$
26:    end if
27:    $k = k + 1$
28: end while
```

**Worst-case complexity of the Riemannian trust-region**

The study of global complexity in optimization establishes the number of iterations that a particular algorithm takes in the worst-case to reach some approximate criticality condition from an arbitrary initial guess of the solution. Classical first- and second-order optimization algorithms on Riemannian manifolds have global worst-case complexities with rates matching their unconstrained counterparts, as was shown in (Boumal et al., 2019). To state this result for Algorithm 2, we introduce some assumptions.

**A1.** *There exists $f^* > -\infty$ such that $f(x) \geq f^*$ for all $x \in \mathcal{M}$.*

Assumptions A2 and A3 impose some Lipschitz continuity conditions on the pullback of $f$. In Section 4.5, we discuss the practicality of these assumptions for our formulation of nonlinear matrix recovery problems.

**A2** (Lipschitz gradient of the pullback)**.** *There exists $L_g \geq 0$ such that for all $x \in \mathcal{M}$, the pullback $\hat{f}_x = f \circ \mathrm{R}_x$ has Lipschitz continuous gradient with constant $L_g$, that is, for all $\eta \in \mathrm{T}_x\mathcal{M}$, it holds that*

$$\left|\hat{f}_x(\eta) - [f(x) + \langle \eta, \mathrm{grad} f(x)\rangle]\right| \leq \frac{L_g}{2}\left\|\eta\right\|^2. \tag{2.16}$$

**A3** (Lipschitz Hessian of the pullback)**.** *There exists $L_H \geq 0$ such that, for all $x \in \mathcal{M}$, the pullback $\hat{f}_x = f \circ \mathrm{R}_x$ has Lipschitz continuous Hessian with constant $L_H$, that is, for all $\eta \in \mathrm{T}_x\mathcal{M}$, it holds that*

$$\left|\hat{f}_x(\eta) - \left[f(x) + \langle \eta, \mathrm{grad} f(x)\rangle + \frac{1}{2}\left\langle \eta, \nabla^2 \hat{f}_x(0_x)[\eta]\right\rangle\right]\right| \leq \frac{L_H}{6}\left\|\eta\right\|^3. \tag{2.17}$$

**A4.** *There exists $c_0 \geq 0$ such that, for all first-order steps, $\|H_k\| \leq c_0$ and $H_k$ is radially linear, that is, $\forall \alpha \geq 0$ and $\forall \eta \in \mathrm{T}_{x_k}\mathcal{M}$, $H_k[\alpha\eta] = \alpha H_k[\eta]$.*

The assumptions A5 and A6 impose some sufficient decrease on the model for first- and second-order steps which are required for the convergence analysis of RTR. In practical implementations of Algorithm 2, these conditions may not be enforced, and we describe below a truncated conjugate gradient scheme used in practice to minimize the RTR subproblems.

**A5.** *There exists $c_2 > 0$ such that all first-order steps $\eta_k$ satisfy*

$$\hat{m}_k(0_{x_k}) - \hat{m}_k(\eta_k) \geq c_2 \min\left(\Delta_k, \frac{\varepsilon_g}{c_0}\right)\varepsilon_g. \tag{2.18}$$

**A6.** *There exists $c_3 > 0$ such that all second-order steps $\eta_k$ satisfy*

$$\hat{m}_k(0_{x_k}) - \hat{m}_k(\eta_k) \geq c_3 \Delta_k^2 \varepsilon_H. \tag{2.19}$$

Finally, A7 ensures that $H_k$ is close to the Hessian of the pullback in directions selected by the algorithm.

**A7.** *There exists $c_1 \geq 0$ such that, for all second-order steps,*

$$\left|\left\langle \eta_k, \left(\nabla^2 \hat{f}_{x_k}(0_{x_k}) - H_k\right)[\eta_k]\right\rangle\right| \leq \frac{c_1 \Delta_k}{3} \|\eta_k\|^2 .$$

*In addition, for all second-order steps, $H_k$ is linear and symmetric.*

Define the following constants

$$\lambda_g = \frac{1}{4} \min\left(\frac{1}{c_0}, \frac{c_2}{L_g + c_0}\right) \qquad \text{and} \qquad \lambda_H = \frac{3}{4} \frac{c_3}{L_H + c_1}.$$

The following (sharp) worst-case bound for Riemannian trust-region was recently established.

**Theorem 2.8** (Global complexity of RTR (Boumal et al., 2019))**.** *Under A1, A2, A5, A4 and assuming $\varepsilon_g \leq \frac{\Delta_0}{\lambda_g}$, Algorithm 2 produces an iterate $x_{N_1}$ satisfying $\|\mathrm{grad} f(x_{N_1})\| \leq \varepsilon_g$ with*

$$N_1 \leq \mathcal{O}(1/\varepsilon_g^2).$$

*Furthermore, if $\varepsilon_H < \infty$ then under additionally A3, A6, A7 and assuming $\varepsilon_g \leq \frac{c_2}{c_3} \frac{\lambda_H}{\lambda_g^2}$ and $\varepsilon_H \leq \frac{c_2}{c_3 \lambda_g}$, Algorithm 2 also produces an iterate $x_{N_2}$ satisfying $\|\mathrm{grad} f(x_{N_2})\| \leq \varepsilon_g$ and $\lambda_{\min}(H_{N_2}) \geq -\varepsilon_H$ with*

$$N_1 \leq N_2 \leq \mathcal{O}\left(\frac{1}{\varepsilon_g^2 \varepsilon_H}\right).$$

**Truncated conjugate gradient method for the RTR subproblem**

In implementations of the Riemannian trust-region, it is typical to neglect A5 and A6 to approximately minimize the RTR subproblem using a truncated conjugate gradients method (Algorithm 3). This is the method of choice in the Manopt toolbox for Riemannian optimization, which we introduce in the next section.

### 2.8.3 Riemannian optimization toolboxes: Manopt and PyManopt

For the implementation of Riemannian optimization methods, we use the Manopt toolbox in Matlab (Boumal et al., 2014) and Pymanopt, its adaptation in Python (Townsend et al., 2016). Both toolboxes implement a number of classical solvers, including the Riemannian trust-region which we use as the default solver for smooth optimization problems on manifolds. The toolboxes also implement the geometry of the Riemannian manifolds

**Algorithm 3** truncated Conjugate Gradient method (Absil et al., 2008, page 144)

```
1: Given:  $x_k \in \mathcal{M}$ and $\Delta, \theta, \kappa > 0$
2: Init:  $\eta_0 = 0 \in T_{x_k}\mathcal{M}$, $r_0 = \mathrm{grad}f(x_k)$, $\delta_0 = -r_0$
3: for $j = 0, \dots,$ max inner iterations $- 1$ do
4:     if $\langle \delta_j, H_k[\delta_j] \rangle_{x_k} \leq 0$ then
5:         Compute $\tau$ such that $\eta = \eta_j + \tau\delta_j$ minimizes $\hat{m}_{x_k}$ in (2.14) and satisfies $\|\eta\|_{x_k} = \Delta$ as in (Conn et al., 2000, Eqs. (7.5.5)-(7.5.7))
6:         return $\eta_k := \eta$
7:     end if
8:     Set $\alpha_j = \langle r_j, r_j \rangle_{x_k} / \langle \delta_j, H_k[\delta_j] \rangle_{x_k}$
9:     Set $\eta_{j+1} = \eta_j + \alpha_j \delta_j$
10:    if $\|\eta_{j+1}\|_{x_k} \geq \Delta$ then
11:        Compute $\tau \geq 0$ such that $\eta = \eta_j + \tau\delta_j$ satisfies $\|\eta\|_{x_k} = \Delta$
12:        return $\eta_k := \eta$
13:    end if
14:    $r_{j+1} = r_j + \alpha_j H_k[\delta_j]$
15:    if $\|r_{k+1}\|_{x_k} \leq \|r_0\|_{x_k} \min(\|r_0\|_{x_k}^{\theta}, \kappa)$ then
16:        return $\eta_k := \eta_{j+1}$          ▷ This approximate solution is good enough
17:    end if
18:    $\beta_{j+1} = \langle r_{j+1}, r_{j+1} \rangle_{x_k} / \langle r_j, r_j \rangle_{x_k}$
19:    $\delta_{j+1} = -r_{j+1} + \beta_{j+1}\delta_j$
20: end for
```

that we encounter in this thesis, with the exception of affine subspaces (Section 2.9.2) which we use for matrix sensing and have implemented ourselves. The only sensible difference between the Matlab and Python toolboxes resides in the way the derivatives of the cost function are computed. In Matlab, the toolbox expects to receive an expression for the (Euclidean) gradient of the cost function and its Hessian if needed. If they are not provided, finite differences are used, which is typically slow. Pymanopt benefits from a powerful automatic differentiation library in Python, and there is no need to provide the derivatives; the cost function and the manifold on which is it defined are enough to define an optimization problem and call a solver. The latest version of Manopt introduced automatic differentiation, but we did not use it for the results in this thesis.

## 2.9 Manifolds of interest

We provide a list of the Riemannian manifolds that appear throughout the thesis, and describe their geometries. When we encounter these manifolds in the chapters to come, we reference the reader back to this section for a detailed description.

### 2.9.1 Euclidean spaces

First of all, it is worth noting that a Euclidean space $\mathcal{E}$ with inner product $\langle \cdot, \cdot \rangle$ is a Riemannian manifold. At each point $x \in \mathcal{E}$, the tangent space is the Euclidean space itself, $\mathrm{T}_x\mathcal{E} = \mathcal{E}$, with the inner product $\langle \cdot, \cdot \rangle$ as the Riemannian metric. The exponential map is given by $\mathrm{Exp}_x(v) = x+v$ for $v \in \mathcal{E}$, which provides an easily computable retraction. This trivial identification of a Euclidean space as a Riemannian manifold has the benefit that unconstrained optimization problems and methods are part of the formalism of Riemannian optimization, and do not require particular attention.

### 2.9.2 Affine subspaces

We describe an affine subspace in $\mathbb{R}^{n\times s}$ as a Riemannian manifold. This set appears in Chapter 4, where we solve matrix recovery problems using Riemannian optimization. Naturally, the formalism of Riemannian manifolds is not strictly needed to deal with affine subspaces, as they are flat spaces. Nevertheless, using the Riemannian approach gives a straightforward way to implement feasible methods for affine constraints, and is useful when these constraints appear in conjunction with other (more complex) Riemannian manifolds.

Consider $m$ linearly independent matrices $A_1, \ldots, A_m \in \mathbb{R}^{n\times s}$ and the linear operator $\mathcal{A}\colon \mathbb{R}^{n\times s} \to \mathbb{R}^m$ defined by $\mathcal{A}(X)_i = \langle A_i, X\rangle$ for $i = 1, \ldots, m$ where $\langle \cdot, \cdot \rangle$ is the usual trace inner product in $\mathbb{R}^{n\times s}$. For some vector $b \in \mathbb{R}^m$, define the set

$$\mathrm{L}_{\mathcal{A},b} = \{X \in \mathbb{R}^{n\times s} : \mathcal{A}(X) = b\}, \tag{2.20}$$

an embedded submanifold of $\mathbb{R}^{n\times s}$. At any point $X \in \mathrm{L}_{\mathcal{A},b}$, the tangent space is the null space of $\mathcal{A}$,

$$\mathrm{T}_X\mathrm{L}_{\mathcal{A},b} = \ker(\mathcal{A}) = \{\Delta \in \mathbb{R}^{n\times s} : \mathcal{A}(\Delta) = 0\}.$$

Since the tangent space does not depend on $X$, we write $\mathrm{TL}_{\mathcal{A},b}$. The tangent space inherits an inner product from the embedding space $\mathbb{R}^{n\times s}$,

$$\langle \Delta_1, \Delta_2\rangle = \mathrm{trace}(\Delta_1^\top \Delta_2) \ \text{ for all } \Delta_1, \Delta_2 \in \mathrm{TL}_{\mathcal{A},b}.$$

The Riemannian gradient of a function $f\colon \mathrm{L}_{\mathcal{A},b} \to \mathbb{R}$ is given by the orthogonal projection of the Euclidean gradient onto the tangent space $\mathrm{TL}_{\mathcal{A},b}$, as indicated by Equation (2.3). From the fundamental theorem of algebra, $\ker(\mathcal{A}) = \mathrm{range}(\mathcal{A}^\top)^\perp$. Therefore we can express $\mathrm{P}_{\ker(\mathcal{A})} = \mathrm{Id} - \mathrm{P}_{\mathrm{range}(\mathcal{A}^\top)}$. The application $\mathcal{A}$ is represented by a flat matrix $A \in \mathbb{R}^{m\times ns}$ such that $\mathcal{A}(X) = AX(:) \in \mathbb{R}^m$, where $X(:)$ is a vector of length $ns$ made of the columns of $X$ taken from left to right and stacked on top of each other. Concretely this gives,

$$\mathcal{A}(X) = \begin{pmatrix} \langle A_1, X\rangle \\ \langle A_2, X\rangle \\ \vdots \\ \langle A_m, X\rangle \end{pmatrix} = \begin{pmatrix} A_1(:)^\top \\ A_2(:)^\top \\ \vdots \\ A_m(:)^\top \end{pmatrix} X(:) =: AX(:).$$

The tall matrix $A^\top$ represents the linear application $\mathcal{A}^\top$. Viewing $A^\top$ as the matrix of an overdetermined system of linear equations convinces us that $\mathrm{P}_{\mathrm{range}(\mathcal{A}^\top)} = A^\top(AA^\top)^{-1}A$. Equivalently, if $Q \in \mathbb{R}^{m\times m}$ is an orthogonal basis for range$(\mathcal{A}^\top)$ (which can be obtained by a reduced QR factorization of $A^\top$), we can apply $\mathrm{P}_{\mathrm{range}(\mathcal{A}^\top)} = QQ^\top$. The projection is given by

$$\mathrm{P}_{\ker(\mathcal{A})} = \mathrm{I}_{ns} - A^\top(AA^\top)^{-1}A = \mathrm{I}_{ns} - QQ^\top. \tag{2.21}$$

Therefore,

$$\mathrm{P}_{\mathrm{TL}_{\mathcal{A},b}} \colon \mathbb{R}^{n\times s} \to \mathrm{TL}_{\mathcal{A},b} \colon \Delta \mapsto (\mathrm{I}_{ns} - QQ^\top)\Delta \tag{2.22}$$

where $Q \in \mathbb{R}^{m\times m}$ is an orthogonal basis for range$(\mathcal{A}^\top)$. The Riemannian Hessian of a function $f \colon \mathrm{L}_{\mathcal{A},b} \to \mathbb{R}$ is computed using Equation (2.6).

As Chapter 3 details, in matrix completion problems, the operator $\mathcal{A}$ selects the known entries of some matrix $M \in \mathbb{R}^{n\times s}$. In that case, the matrices $A_i$ have all entries equal to zero, except one entry with value 1. Let $\Omega$ denote the set of indices $(i, j)$ selected by the operator $\mathcal{A}$. The description of the feasible subspace is simplified. We write $\mathcal{L}_{\Omega,b} = \{X \in \mathbb{R}^{n\times s} : X_{ij} = M_{ij}, ij \in \Omega\}$ to make explicit that the measurements correspond to matrix completion. The tangent space is $\mathrm{TL}_{\Omega,b} = \{\Delta \in \mathbb{R}^{n\times s} : \Delta_{ij} = 0 \text{ for } ij \in \Omega\}$ and the projection onto $\mathrm{TL}_{\Omega,b}$ simply amounts to setting the entries in $\Omega$ to zero,

$$\mathrm{P}_{\mathrm{T}\mathcal{L}_{\Omega,b}}(\Delta) = \begin{cases} 0 & \text{for } ij \in \Omega \\ \Delta_{ij} & \text{for } ij \notin \Omega. \end{cases}$$

The natural retraction on $\mathrm{L}_{\mathcal{A},b}$ for $X \in \mathrm{L}_{\mathcal{A},b}$ and $\Delta \in \mathrm{TL}_{\mathcal{A},b}$ is given by

$$\mathrm{R}_X \colon \mathrm{TL}_{\mathcal{A},b} \to \mathrm{L}_{\mathcal{A},b} \colon \Delta \mapsto X + \Delta, \tag{2.23}$$

because the manifold is flat. Hence this simple retraction also happens to be the exponential map.

### 2.9.3 The Stiefel manifold: orthonormal matrices

For $p \leq n$, the set of orthogonal matrices of size $n \times p$ is called the Stiefel manifold:

$$\mathrm{St}(n, p) = \{X \in \mathbb{R}^{n\times p} : X^\top X = \mathrm{I}_p\},$$

where $\mathrm{I}_p$ is the identity matrix of size $p$. The defining function for the Stiefel manifold is given by

$$h\colon \mathbb{R}^{n\times p} \to \mathrm{Sym}(p)\colon X \mapsto h(X) = X^\top X - \mathrm{I}_p,$$

where $\mathrm{Sym}(p)$ is the space of symmetric matrices of size $p$. The application of Theorem 2.1 shows that $\mathrm{St}(n,p)$ is an embedded submanifold of $\mathbb{R}^{n\times p}$. We verify that $h$ is smooth and $h^{-1}(0) = \mathrm{St}(n,p)$. A direct computation gives

$$\mathrm{D}h(X)\colon \mathbb{R}^{n\times p} \to \mathrm{Sym}(p)\colon V \mapsto \mathrm{D}h(X)[V] = X^\top V + V^\top X.$$

One can verify that $\mathrm{D}h(X)$ is surjective and has rank $p(p+1)/2$ for all $X \in \mathrm{St}(n,p)$. Therefore, $\dim \mathrm{St}(n,p) = \dim \mathbb{R}^{n\times p} - \dim \mathrm{Sym}(p) = np - p(p+1)/2$. The tangent space to $\mathrm{St}(n,p)$ at $X$ is given by the null space of $\mathrm{D}h(X)$, that is,

$$\mathrm{T}_X\mathrm{St}(n,p) = \ker \mathrm{D}h(X) = \{V \in \mathbb{R}^{n\times p} : X^\top V + V^\top X = 0\}. \tag{2.24}$$

It is useful to parametrize the tangent space explicitly. For $X \in \mathrm{St}(n,p)$, consider $X^\perp \in \mathbb{R}^{n\times(n-p)}$ such that $\begin{bmatrix} X & X^\perp \end{bmatrix}$ is orthogonal. Since $\begin{bmatrix} X & X^\perp \end{bmatrix}$ is invertible, any matrix $V \in \mathbb{R}^{n\times p}$ can be written as

$$V = \begin{bmatrix} X & X^\perp \end{bmatrix} \begin{bmatrix} \Omega \\ B \end{bmatrix} = X\Omega + X^\perp B,$$

for a unique choice of $\Omega \in \mathbb{R}^{p\times p}$ and $B \in \mathbb{R}^{(n-p)\times p}$. The matrix $V \in \mathbb{R}^{n\times p}$ is a tangent vector at $X \in \mathrm{St}(n,p)$ if and only if $\mathrm{D}h(X)[V] = 0$. This gives

$$0 = \mathrm{D}h(X)[V] = X^\top(X\Omega + X^\perp B) + (X\Omega + X^\perp B)^\top X = \Omega + \Omega^\top.$$

This restricts $\Omega$ to be skew-symmetric, while $B$ is free. We have reached a new description of the tangent space:

$$\mathrm{T}_X\mathrm{St}(n,p) = \left\{X\Omega + X^\perp B\colon \Omega \in \mathrm{Skew}(p), B \in \mathbb{R}^{(n-p)\times p}\right\}, \tag{2.25}$$

where $\mathrm{Skew}(p) = \left\{\Omega \in \mathbb{R}^{p\times p} : \Omega^\top = -\Omega\right\}$. The projection of a vector $V \in \mathbb{R}^{n\times p}$ onto the tangent space $\mathrm{T}_X\mathrm{St}(n,p)$ is given by

$$\begin{aligned} \mathrm{Proj}_X(V) &= V - X\frac{X^\top V + V^\top X}{2} && (2.26)\\ &= (\mathrm{I} - XX^\top)V + X\frac{X^\top V - V^\top X}{2}. && (2.27) \end{aligned}$$

The manifold $\mathrm{St}(n,p)$ is naturally turned into a Riemannian manifold by making it a Riemannian submanifold of $\mathbb{R}^{n\times p}$. The Riemannian metric is the usual trace inner product.

Consider a smooth real-valued function, $f\colon \mathrm{St}(n,p) \to \mathbb{R}$, with a smooth extension on $\mathbb{R}^{n\times p}$, labelled $\overline{f}\colon \mathbb{R}^{n\times p} \to \mathbb{R}$. We wish to find a formula for the Riemannian gradient and Riemannian Hessian of $f$ using the derivatives of $\overline{f}$. These follow from the results in Sections 2.3.1 and 2.4.1, and the fact that $\mathrm{St}(n,p)$ is a Riemannian submanifold of $\mathbb{R}^{n\times p}$. Equation (2.3) gives the gradient of $f$ as the projection of the gradient of $\overline{f}$ on the tangent space:

$$\mathrm{grad}f(X) = \mathrm{Proj}_X\left(\mathrm{grad}\overline{f}(X)\right) = \mathrm{grad}\overline{f}(X) - X\mathrm{sym}\left(X^\top \mathrm{grad}\overline{f}(X)\right),$$

where $\mathrm{sym}(M) = (M^\top + M)/2$ is the symmetric part of a matrix. Covariant derivatives on $\mathrm{St}(n,p)$ coincides with the usual vector field derivative in $\mathbb{R}^{n\times p}$, followed by orthogonal projection to the tangent space (Equation (2.5)):

$$\begin{aligned}\mathrm{Hess}f(X)[V] &= \nabla_V \mathrm{grad}f(X)\\ &= \mathrm{Proj}_X\left(\mathrm{D}\mathrm{grad}f(X)[V]\right)\\ &= \mathrm{Proj}_X\left(\mathrm{Hess}\overline{f}(X)[V] - V\mathrm{sym}(X^\top\mathrm{grad}\overline{f}(X)) - XS\right)\\ &= \mathrm{Proj}_X\left(\mathrm{Hess}\overline{f}(X)[V] - V\mathrm{sym}(X^\top\mathrm{grad}\overline{f}(X))\right),\end{aligned}$$

where $S = \mathrm{sym}\big(V^\top\mathrm{grad}\overline{f}(X) + X^\top\mathrm{Hess}\overline{f}(X)[V]\big)$, and $XS$ vanishes through $\mathrm{Proj}_X$.

A popular retraction for the Stiefel manifold is the Q-factor retraction. For $X \in \mathrm{St}(n,p)$ and $V \in \mathrm{T}_X\mathrm{St}(n,p)$:

$$\mathrm{R}_X(V) = Q, \tag{2.28}$$

where $QR = X+V$ is a QR decomposition: $Q \in \mathrm{St}(n,p)$ and $R \in \mathbb{R}^{p\times p}$ is upper triangular with nonnegative diagonal entries.

### 2.9.4 The orthogonal group and rotation matrices

The *orthogonal group* consists of square matrices with orthonormal columns, that is

$$\mathrm{O}(n) = \left\{X \in \mathbb{R}^{n\times n} : X^\top X = XX^\top = \mathrm{I}_n\right\},$$

whose dimension is

$$\dim \mathrm{O}(n) = n^2 - n(n+1)/2 = n(n-1)/2.$$

Naturally, the orthogonal group correspond to square Stiefel matrices, that is, $\mathrm{O}(n) = \mathrm{St}(n,n)$. Therefore, the expression of tangent spaces on $\mathrm{O}(n)$ follows from the tangent spaces on $\mathrm{St}(n,p)$ (Equation (2.25)):

$$\mathrm{T}_X\mathrm{O}(n) = \{X\Omega : \Omega \in \mathrm{Skew}(n)\} = X\mathrm{Skew}(n), \tag{2.29}$$

where $\mathrm{Skew}(n) = \left\{X \in \mathbb{R}^{n\times n} : X^\top = -X\right\}$. The orthogonal projection onto tangent spaces is identical to the projection onto $\mathrm{St}(n,p)$, but for square matrices, the fact that $XX^\top = \mathrm{I}_n$ brings a simplification:

$$\begin{aligned}\mathrm{Proj}_X(V) &= V - X\mathrm{sym}(X^\top V)\\ &= X\,\mathrm{skew}(X^\top V),\end{aligned}$$

where $\mathrm{skew}(M) = \dfrac{M - M^\top}{2}$ extracts the skew-symmetric part of a matrix. We turn $\mathrm{O}(n)$ into a Riemannian submanifold by restricting the canonical inner product on $\mathbb{R}^{n\times n}$ to tangent spaces of $\mathrm{O}(n)$. Using that $\mathrm{O}(n)$ is a Riemannian submanifold of $\mathbb{R}^{n\times n}$, we get a formula for the Riemannian gradient of a function $f\colon \mathrm{O}(n) \to \mathbb{R}$ based on the gradient of a smooth extension $\overline{f}\colon \mathbb{R}^{n\times n} \to \mathbb{R}$. Equation (2.3) yields

$$\begin{aligned}\mathrm{grad}f(X) &= \mathrm{Proj}_X\big(\nabla\overline{f}(X)\big)\\ &= X\,\mathrm{skew}(X^\top \mathrm{grad}\overline{f}(X)).\end{aligned}$$

The orthogonal group is made of two distinct connected components, corresponding to matrices of determinant $+1$ and $-1$. Orthogonal matrices represent rotations in $\mathbb{R}^n$, composed with a reflection for those with determinant $-1$. The set of pure rotations is called the *special orthogonal group* :

$$\mathrm{SO}(n) = \{X \in \mathrm{O}(n) : \det(X) = +1\}\,.$$

Optimization algorithms on manifolds stay in the connected component that they are initialized in. Hence, it matters in which component the algorithm is initialized.

For the Hessian of a real-valued function, the formula established for $\mathrm{St}(n,p)$ is also valid on $\mathrm{O}(n)$ and $\mathrm{SO}(n)$:

$$\mathrm{Hess}f(X)[V] = \mathrm{Proj}_X\big(\mathrm{Hess}\overline{f}(X)[V] - V\mathrm{sym}\big(X^\top\mathrm{grad}\overline{f}(X)\big)\big).$$

The Q-factor retraction (Equation (2.28)), which is described in the previous section for Stiefel manifolds, is also a retraction on $\mathrm{O}(n)$ and $\mathrm{SO}(n)$.

### 2.9.5 The Grassmann manifold

The Grassmann manifold, written $\mathrm{Grass}(N,r)$, is the set of all linear subspaces of dimension $r$ in $\mathbb{R}^N$. This manifold appears in Chapters 4 and 5, for the rank minimization of a feature matrix. We follow the geometry described in (Boumal and Absil, 2015) and implemented in Manopt. A point $\mathcal{U} \in \mathrm{Grass}(N,r)$ is represented by a matrix $U \in \mathrm{St}(N,r)$ such that $\mathrm{range}(U) = \mathcal{U}$. For any orthogonal matrix $Y \in \mathrm{O}(r)$, the matrix $UY$ is also a valid representation of $\mathcal{U}$ since range$(UY) = $ range$(U)$. The orthogonal group induces an equivalence relation on the Stiefel manifold, where any two matrices with the same column space are equivalent. In this regard, the Grassmann is a quotient manifold

$$\mathrm{Grass}(N,r) = \mathrm{St}(N,r)/\mathrm{O}(r), \tag{2.30}$$

and inherits properties as a Riemannian quotient manifold of the ambient space $\mathrm{St}(N,r)$ as described in Sections 2.2.2 and 2.3.2. As a manifold, the Grassmann admits a tangent space at every point $\mathcal{U}$. Once the representation $U$ has been chosen for $\mathcal{U}$, tangent vectors $\mathcal{H} \in \mathrm{T}_{\mathcal{U}}\mathrm{Grass}(N,r)$ are represented by their horizontal lift at $U \in \mathrm{St}(N,r)$, namely, as matrices $H \in \mathbb{R}^{N\times r}$. The tangent space $\mathrm{T}_{\mathcal{U}}\mathrm{Grass}(N,r)$ is represented by the horizontal space at $U \in \mathrm{St}(N,r)$, which we denote by

$$\mathrm{T}_U\mathrm{Grass}(N,r) = \{H \in \mathbb{R}^{N\times r} : U^\top H = 0\} = \mathrm{range}(U_\perp). \tag{2.31}$$

This is a slight abuse of notation since the matrix $U$ cannot belong to the Grassmann manifold. The projection onto the horizontal space $\mathrm{T}_U\mathrm{Grass}(N,r)$ is given by

$$\mathrm{Proj}_U^h(H) = (\mathrm{I} - UU^\top)H, \tag{2.32}$$

with $U \in \mathrm{St}(N,r)$. The horizontal space is endowed with the usual inner product inherited from the ambient space $\mathbb{R}^{N\times r}$,

$$\langle H_1, H_2\rangle_U = \mathrm{trace}(H_1^\top H_2), \quad \forall H_1, H_2 \in \mathrm{T}_U\mathrm{Grass}(N,r). \tag{2.33}$$

We define the *polar retraction* on $\mathrm{Grass}(N,r)$, given for $\mathcal{U} \in \mathrm{Grass}(N,r)$ and $\Delta \in \mathrm{T}_U\mathrm{Grass}(N,r)$ by

$$\mathrm{R}_U \colon \mathrm{T}_U\mathrm{Grass}(N,r) \to \mathrm{Grass}(N,r) \colon \mathrm{R}_U(H) = \mathrm{range}\left(\mathrm{polar}(U+H)\right), \tag{2.34}$$

where $\mathrm{polar}(A) \in \mathrm{St}(N,r)$ denotes the orthogonal factor of the polar decomposition of $A \in \mathbb{R}^{N\times r}$. This can be computed from a (thin) singular value decomposition: let $A = U\Sigma V^\top$, then $\mathrm{polar}(A) = UV^\top$.

Consider a smooth function $\overline{f} : \mathbb{R}_*^{N\times r} \to \mathbb{R}$, where $\mathbb{R}_*^{N\times r}$ is the set of full-rank matrices of size $N \times r$. Let $\overline{f}|_{\mathrm{St}}$ denote its restriction to the Stiefel manifold and further assume that $\overline{f}|_{\mathrm{St}}$ is only a function of the column space of its argument:

$$\overline{f}|_{\mathrm{St}}(U) = \overline{f}|_{\mathrm{St}}(UQ) \qquad \text{for all } U \in \mathrm{St}(N,r),\ Q \in \mathrm{O}(r).$$

Under this assumption, the function

$$f \colon \mathrm{Grass}(N,r) \to \mathbb{R} \colon \mathcal{U} \mapsto f(\mathcal{U}) = \overline{f}|_{\mathrm{St}}(U)$$

is well defined. It is possible to express the Riemannian derivatives of $f$ using the derivatives of $\overline{f}$. Using that $\mathrm{St}(N,r)$ is a Riemannian submanifold of $\mathbb{R}_*^{N\times r}$ and that $\mathrm{Grass}(N,r)$ is a Riemannian quotient manifold of $\mathrm{St}(N,r)$, we combine Equations (2.3) and (2.4) to reach

$$\mathrm{grad} f(U) = \mathrm{Proj}_U^h(\mathrm{grad}\overline{f}(U)) = (I - UU^\top)\mathrm{grad}\overline{f}(U),$$

as presented in (Boumal and Absil, 2015, Equation 11). In practice, this means that we compute the gradient of $\overline{f}$ in the usual way and then project it with $I-UU^\top$. Additionally, the Hessian of $f$ follows from Equations (2.6) and (2.7); and is given by

$$\mathrm{Hess} f(U)[H] = \mathrm{Proj}_U^h(\mathrm{D}\overline{F}(U)[H]) = (I - UU^\top)\mathrm{D}\overline{F}(U)[H],$$

with

$$\overline{F}(U) \colon \mathbb{R}_*^{N\times r} \to \mathbb{R}^{N\times r} \colon U \mapsto \overline{F}(U) = (I - UU^\top)\mathrm{grad}\overline{f}(U),$$

as presented in (Boumal and Absil, 2015, Equation 12).

### 2.9.6 Manifolds defined by $h(x) = 0$

Working over a Euclidean space $\mathcal{E}$ with inner product $\langle\cdot,\cdot\rangle$ and associated norm $\|\cdot\|$, define the set

$$\mathcal{M} = \{x \in \mathcal{E} : h(x) = 0\},$$

for some smooth function $h \colon \mathcal{E} \to \mathbb{R}^m$. This is the setting that we consider in Chapter 6: generic smooth equality constraints, for which there is no closed-form expression to generate a feasible point. We follow the presentation in (Boumal, 2020, Section 7.7). The following proposition gives a condition for the set $\mathcal{M}$ to be a smooth manifold, which follows directly from Theorem 2.1.

**Proposition 2.9.** *The set $\mathcal{M} = \{x \in \mathcal{E} : h(x) = 0\}$ is an embedded submanifold of $\mathcal{E}$ if and only if $\mathrm{D}h(x)$ has full rank $m$ for all $x \in \mathcal{M}$.*

Using the notation $h(x) = (h_1(x), \ldots, h_m(x))^\top$ to denote the $m$ components of $h$, we find the linear operator

$$\mathrm{D}h(x)[v] = (\langle \nabla h_1(x), v\rangle, \ldots, \langle \nabla h_m(x), v\rangle)^\top,$$

and its adjoint

$$\mathrm{D}h(x)^*[\alpha] = \sum_{i=1}^m \alpha_i \nabla h_i(x).$$

The tangent spaces are given by

$$\mathrm{T}_x\mathcal{M} = \ker \mathrm{D}h(x) = \{v \in \mathcal{E} : \langle v, \nabla h_i(x)\rangle = 0 \text{ for all } i\}.$$

The fact that $\mathrm{D}h(x)$ has full rank means that the gradients $\nabla h_i(x)$ are linearly independent. They form a basis of the normal space at $x$:

$$\mathrm{N}_x\mathcal{M} = (\ker \mathrm{D}h(x))^{\perp} = \operatorname{span}(\nabla h_1(x), \ldots, \nabla h_m(x)).$$

The manifold $\mathcal{M}$ is then naturally turned into a Riemannian manifold by restricting the inner product $\langle \cdot, \cdot \rangle$ to $\mathrm{T}\mathcal{M}$. For any vector $x \in \mathcal{E}$, there exists a unique $\alpha \in \mathbb{R}^m$ such that

$$v = \operatorname{Proj}_x(v) + \mathrm{D}h(x)^*[\alpha],$$

where $\alpha$ is the unique solution to the least squares problem

$$\alpha = \operatorname*{arg\,min}_{\alpha \in \mathbb{R}^m} \|v - \mathrm{D}h(x)^*[\alpha]\| = (\mathrm{D}h(x)^*)^{\dagger}[v],$$

where the dagger denotes the Moore–Penrose pseudo-inverse. This gives the projection onto the tangent space as

$$\operatorname{Proj}_x(v) = v - \mathrm{D}h(x)^* \left[(\mathrm{D}h(x)^*)^{\dagger}[v]\right].$$

Consider the equality constrained optimization problem

$$\min_x f(x) \text{ subject to } h(x) = 0, \tag{2.35}$$

where $f\colon \mathcal{E} \to \mathbb{R}$ and $h\colon \mathcal{E} \to \mathbb{R}^m$ are smooth. Using that $\mathcal{M}$ is a Riemannian submanifold of $\mathcal{E}$, a formula for the Riemannian gradient of $f$ on $\mathcal{M}$ follows from (2.3):

$$\operatorname{grad} f(x) = \operatorname{Proj}_x(\nabla f(x)) = \nabla f(x) - \mathrm{D}h(x)^*[\lambda(x)] \tag{2.36}$$

where $\lambda(x) = (\mathrm{D}h(x)^*)^{\dagger}[\nabla f(x)]$. In other words,

$$\nabla f(x) = \operatorname{grad} f(x) + \sum_{i=1}^m \lambda_i(x) \nabla h_i(x).$$

For the second derivatives, we get from (2.6) that $\operatorname{Hess} f(x)[v]$ is the orthogonal projection of $\mathrm{D}(\operatorname{grad} f(x))[v]$ to $\mathrm{T}_x\mathcal{M}$. We find

$$\mathrm{D}(\operatorname{grad} f(x))[v] = \nabla^2 f(x)[v] - \sum_{i=1}^m \mathrm{D}\lambda_i(x)[v] \cdot \nabla h_i(x) - \sum_{i=1}^m \lambda_i(x) \nabla^2 h_i(x)[v].$$

Since each $\nabla h_i(x)$ is orthogonal to $\mathrm{T}_x\mathcal{M}$, it follows that

$$\operatorname{Hess} f(x)[v] = \operatorname{Proj}_x\left(\nabla^2 f(x)[v] - \sum_{i=1}^m \lambda_i(x) \nabla^2 h_i(x)[v]\right).$$

We have reached the conclusion that, for all $x \in \mathcal{M}$:

$$\operatorname{Hess} f(x) = \operatorname{Proj}_x \circ \left(\nabla^2 f(x) - \sum_{i=1}^m \lambda_i(x) \nabla^2 h_i(x)\right) \circ \operatorname{Proj}_x. \tag{2.37}$$

**Optimality conditions on manifolds defined by $h(x) = 0$.** Consider the Lagrangian function

$$\mathcal{L}\colon \mathcal{E} \times \mathbb{R}^m \to \mathbb{R}\colon (x, \lambda) \mapsto \mathcal{L}(x, \lambda) = f(x) - \langle \lambda, h(x) \rangle .$$

First- and second-order necessary optimality conditions for (2.35) are well known in the constrained optimization literature. The first-order condition expresses that at any minimizer, the gradient of the Lagrangian vanishes.

**Proposition 2.10.** *If $x$ is a local minimizer of Problem* (2.35)*, then there exists a vector $\lambda \in \mathbb{R}^m$ such that*

$$\nabla f(x) = \sum_{i=1}^{m} \lambda_i \nabla h_i(x) \text{ and } h(x) = 0. \tag{2.38}$$

Points who satisfy (2.38) are also often called KKT points. At the second-order, it is additionally necessary that the Hessian of the Lagrangian be positive semidefinite on $\mathrm{T}_x\mathcal{M}$.

**Proposition 2.11.** *If $x$ is a minimizer of Problem 2.35, there exists a vector $\lambda \in \mathbb{R}^m$ such that Equation* (2.38) *holds with*

$$\left\langle \nabla^2_{xx}\mathcal{L}(x, \lambda)[v], v \right\rangle \geq 0 \text{ for all } v \in \ker \mathrm{D}h(x). \tag{2.39}$$

Conditions (2.38) and (2.39) are equivalent to their Riemannian analogues in (2.9) and (2.10), which state that a local minimizer $x \in \mathcal{M}$ of problem (2.35), satisfies $\mathrm{grad} f(x) = 0$ and $\mathrm{Hess} f(x) \succeq 0$.

# Chapter 3

# Matrix recovery problems

In this chapter, we introduce and give an overview of the field of matrix recovery. Section 3.1 gives a brief summary of standard low-rank matrix recovery approaches. This serves as a reference for the origins of the problem tackled in this thesis. We present the main formulations of the problem, some of which can be extended to the nonlinear setting. In Section 3.2, we introduce the nonlinear matrix recovery problem, the task of recovering particular classes of high-rank matrices, and describe the case studies that we further consider in Chapters 4 and 5, namely, the algebraic variety model, unions of subspaces and clustering. Finally, we give an up-to-date overview of related work on nonlinear matrix recovery.

For some integer $m < ns$, consider a matrix $M \in \mathbb{R}^{n\times s}$ that satisfies $m$ linear equations $\langle A_i, M\rangle = b_i$ for given matrices $A_i \in \mathbb{R}^{n\times s}$ and a given vector $b \in \mathbb{R}^m$, where we use the usual inner product $\langle A_i, M\rangle = \operatorname{trace}(A_i^\top M)$. We define the linear operator

$$\mathcal{A}\colon \mathbb{R}^{n\times s} \to \mathbb{R}^m \text{ where } \mathcal{A}(M)_i = \langle A_i, M\rangle, \tag{3.1}$$

so as to have the compact notation $\mathcal{A}(M) = b$ for the measurements. The matrices $A_i$ are assumed to be linearly independent. When each matrix $A_i$ has exactly one non-zero entry which is equal to 1, this is known as a *matrix completion* problem. The matrix $M$ is then known on a subset $\Omega$ of the complete set of entries $\{1,\dots,n\} \times \{1,\dots,s\}$. Throughout this chapter, $M \in \mathbb{R}^{n\times s}$ denotes the matrix to be recovered, that is, the solution of the problem. We use $X$ as a variable to denote a generic matrix in $\mathbb{R}^{n\times s}$.

## 3.1 Low-rank matrix recovery

The task of recovering a matrix $M$ based on $\mathcal{A}$ and $b$ alone is ill-posed, as the system is underdetermined. A popular regularization is to seek a low-rank matrix $M$ that satisfies $\mathcal{A}(M) = b$. In this section, we review the main existing approaches of low-rank matrix recovery, the problem of finding a low-rank matrix using only partial (linear) measurements. Low-rank matrix completion has become a very active area of research over the past two decades, given the wide range of applications that are connected to low-rank matrix completion, such as recommendation problems, collaborative filtering and system identification. Surveys on low-rank matrix completion include (Davenport and Romberg, 2016; Nguyen et al., 2019). We explore formulations that have been proposed to solve the low-rank matrix recovery problem and describe algorithms to solve them.

**Problem formulation** The formulation of the problem that is most faithful to practical settings is to look for the matrix of smallest possible rank that satisfies the observations, that is,

$$\begin{cases} \min\limits_{X} & \operatorname{rank}(X) \\ & \mathcal{A}(X) = b. \end{cases} \tag{3.2}$$

Problem (3.2) is NP hard in general (Harvey et al., 2006). Additionally, the rank is an impractical function to minimize as it is not continuous. If the measurements are noisy, (3.2) is not robust and it is more appropriate to minimize a least-squares residual of the measurements, as in

$$\begin{cases} \min\limits_{X} & \|\mathcal{A}(X) - b\|_2^2 \\ & \operatorname{rank}(X) \le r, \end{cases} \tag{3.3}$$

where the rank has been encoded as a constraint using the set

$$\mathcal{M}_{\le r} = \left\{X \in \mathbb{R}^{n \times s} : \operatorname{rank}(X) \le r\right\}.$$

Amongst the approaches that we describe below, some try to solve (3.3) directly, while others work with a convex relaxation of (3.2).

## Summary of existing approaches

**Nuclear norm minimization** This approach aims to solve a relaxation of (3.2) and leverage the power of convex optimization. Formulation (3.2) has a nonconvex cost function and a convex set of constraints. As defined in (Recht et al., 2010), the convex envelope of a function $f \colon \mathcal{C} \to \mathbb{R}$ defined on a convex set $\mathcal{C}$, is the largest convex function $g$ such that $g(x) \le f(x)$ for all $x \in \mathcal{C}$. Recall that the nuclear norm of a matrix $X \in \mathbb{R}^{n \times s}$ is defined as the sum of its singular values,

$$\|X\|_* = \sum_{i=1}^{\min(n,s)} \sigma_i(X).$$

Recht et al. (2010) show that the nuclear norm is the convex envelope of the rank function, and the following problem is the tightest convex relaxation to (3.2):

$$\begin{cases} \min\limits_X & \|X\|_* \\ & \mathcal{A}(X) = b. \end{cases} \tag{3.4}$$

Candès and Tao (2010) show that for matrices with so-called low incoherence, the solution of (3.4) coincides with the global minimizer of (3.2), provided that there are enough measurements, $m > \mathcal{O}(sr\,\mathrm{polylog}(s))$ for $s \geq n$. The convex relaxation (3.4) can be transformed into an equivalent semidefinite program (SDP) whose global minimum can be found in polynomial time using interior point methods (Nesterov, 2004). However, this approach scales poorly in the matrix dimension and most computers run out of memory when $n$ and $s$ reach a few thousands. Hence, the applicability of this relaxation for large-scale matrices is related to the development of scalable SDP solvers, which is an ongoing field of research. First-order methods have also been investigated for nuclear norm minimization, including (Toh and Yun, 2010; Recht and Ré, 2013). For example, Toh and Yun (2010) propose an accelerated proximal gradient method for nuclear norm minimization with a least-squares penalty term for the measurements. An advantage of the convex relaxation (3.4) is that the resulting algorithms are easier to analyse. Various works prove exact recovery guarantees for algorithms based on nuclear norm minimization in the noiseless setting as well as stable recovery in the noisy setting. Scalability concerns have prompted the search for nonconvex formulations that exploit the low-rank structure of the problem and reduce the storage complexity of the algorithms.

**Iterative hard thresholding** Iterative hard thresholding (IHT) is inspired from compressed sensing and the nonconvex formulation (3.3). It is a first-order method which was introduced for matrix completion in (Jain et al., 2010). It consists in taking a gradient step of the objective and then projecting back onto the set $\mathcal{M}_{\leq r}$, so that every iterate satisfies the constraints. The gradient is given by

$$\nabla \left( \|\mathcal{A}(X) - b\|_2^2 \right) = 2\mathcal{A}^*(\mathcal{A}(X) - b)$$

where the adjoint operator is defined as

$$\mathcal{A}^* \colon \mathbb{R}^m \to \mathbb{R}^{n \times s} \colon y \mapsto \mathcal{A}^*(y) = \sum_{i=1}^{m} y_i A_i,$$

with the matrices $A_i$ that define the operator $\mathcal{A}$ (Equation (3.1)). A projection onto the non-convex set $\mathcal{M}_{\leq r}$ can be computed as follows. Let $X = \sum_{i=1}^{\min(n,s)} \sigma_i u_i v_i^\top$ be the singular value decomposition of $X$, with $\sigma_1 \geq \cdots \geq \sigma_{\min(n,s)} \geq 0$. The projection is given by the truncated singular value decomposition

$$\mathrm{P}_{\mathcal{M}_{\leq r}}(X) = \sum_{i=1}^{r} \sigma_i u_i v_i^\top.$$

Note that this projection is not unique if $\sigma_r = \sigma_{r+1}$. The iterative hard thresholding iteration, for some constant step size $\alpha > 0$, is given by

$$X_{k+1} = \mathrm{P}_{\mathcal{M}_{\leq r}} \left( X_k - 2\alpha \mathcal{A}^*(\mathcal{A}(X_k) - b) \right) \qquad \text{for } k = 0, 1, 2, \ldots$$

When the step size varies across iterations, the method is called *normalised* iterative hard thresholding (Tanner and Wei, 2013). Since the iterates produced by IHT have a prescribed rank, they can be stored in a low-rank decomposition, which is critical for large-scale problems as it avoids storing matrices of size $n \times s$.

**Optimization on the fixed-rank manifold** Another approach is to use the fact that the set of matrices of size $n \times s$ and rank $r$,

$$\mathcal{M}_r = \{X \in \mathbb{R}^{n \times s} : \operatorname{rank}(X) = r\},$$

is a smooth manifold, which allows the use of Riemannian optimization methods. The manifold $\mathcal{M}_r$ can be described using the singular value decomposition of a rank $r$ matrix

$$\begin{aligned} \mathcal{M}_r = \{U\Sigma V^\top : U \in \mathbb{R}^{n \times r}, \Sigma \in \mathbb{R}^{r \times r}, V \in \mathbb{R}^{s \times r}, U^\top U = \mathrm{I}_r, V^\top V = \mathrm{I}_r, \\ \Sigma = \operatorname{diag}(\sigma_i),\ \sigma_1 \geq \cdots \geq \sigma_r > 0\}. \end{aligned}$$

Vandereycken (2013) defines a Riemannian geometry for $\mathcal{M}_r$ as an embedded submanifold of $\mathbb{R}^{n \times s}$, and shows how to represent tangent vectors as low-rank matrices and compute retractions efficiently. The author proposes to use a Riemannian conjugate gradient method on $\mathcal{M}_r$ to solve the following matrix completion problem:

$$\begin{cases} \min\limits_{X} & \|\mathrm{P}_\Omega(X - M)\|_\mathrm{F}^2 \\ & X \in \mathcal{M}_r. \end{cases} \tag{3.5}$$

The conjugate gradient method requires the definition of a vector transport on $\mathcal{M}_r$, which allows to compare tangent vectors that belong to tangent spaces at different points on the manifold. An advantage of using optimization methods on $\mathcal{M}_r$, is that matrices and tangent vectors can be stored as low-rank matrices, which keeps the computational costs proportional to the rank, and avoids computations in the ambient space $\mathbb{R}^{n \times s}$.

**Alternating minimization** A matrix $X \in \mathbb{R}^{n\times s}$ of rank at most $r$ can be factorized as $X = UV^\top$ with $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{s\times r}$. This allows to rewrite (3.3) using the variables $U$ and $V$, which gives

$$\begin{cases} \min\limits_{U,V} & \|\mathcal{A}(UV^\top) - b\|_2^2, \\ & U \in \mathbb{R}^{n\times r}, \\ & V \in \mathbb{R}^{s\times r}. \end{cases} \tag{3.6}$$

The strategy used in (Koren et al., 2009) is to minimize (3.6) over each variable separately in an alternating fashion. Each subproblem is a simple linear least-squares:

$$\begin{cases} V_{k+1} = \arg\min\limits_{V\in\mathbb{R}^{s\times r}} \|\mathcal{A}(U_k V^\top) - b\|_2^2 \\ U_{k+1} = \arg\min\limits_{U\in\mathbb{R}^{n\times r}} \|\mathcal{A}(UV_{k+1}^\top) - b\|_2^2. \end{cases}$$

The LMaFit method applies a similar strategy to matrix completion with a dynamic estimate of the rank (Wen et al., 2012).

**Grassmann manifold optimization** One issue in formulation (3.6) is that the decomposition $X = UV^\top$ is not unique. For any invertible matrix $Q \in \mathbb{R}^{r\times r}$, $X$ satisfies $X = (UQ)(Q^{-1}V^\top)$. All matrices of the form $UQ$ have the same column space as $U$ and the cost function does not depend on a specific matrix $U$. To remove this unnecessary degree of freedom in the factorization, Dai et al. (2012) introduce a variable on the Grassmann manifold, written $\mathrm{Grass}(n, r)$, the set of all subspaces of dimension $r$ in $\mathbb{R}^n$, which we describe in Section 2.9.5. The Grassmann manifold is a quotient manifold, where a subspace $\mathcal{U} \in \mathrm{Grass}(n, r)$ is represented by a matrix on the Stiefel manifold $U \in \mathrm{St}(n, p)$, such that $\mathrm{range}(U) = \mathcal{U}$. The formulation in (Dai et al., 2012) is

$$\begin{cases} \min\limits_{\mathcal{U},V} & \sum\limits_{(i,j)\in\Omega} \big((UV^\top)_{ij} - Y_{ij}\big)^2 \\ & \mathcal{U} \in \mathrm{Grass}(n, r) \\ & V \in \mathbb{R}^{s\times r}. \end{cases} \tag{3.7}$$

In (3.7), the function $\mathcal{U}$ might not be continuous with respect to the variable $U$, if the minimization over $V$ does not have a unique solution (Boumal and Absil, 2011). In OptSpace (Keshavan and Oh, 2009), the authors decide to use two variables on the Grassmann manifold, for the column space and row space:

$$\begin{cases} \min\limits_{\mathcal{U},\mathcal{V},S} & \sum\limits_{(i,j)\in\Omega} \big((USV^\top)_{ij} - M_{ij}\big)^2 + \lambda \left\|USV^\top\right\|_{\mathrm{F}}^2 \\ & \mathcal{U} \in \mathrm{Grass}(n, r) \\ & \mathcal{V} \in \mathrm{Grass}(s, r) \\ & S \in \mathbb{R}^{r\times r}, \end{cases} \tag{3.8}$$

where $U$ and $V$ are orthonormal basis of $\mathcal{U}$ and $\mathcal{V}$, and $\lambda > 0$ is a regularization parameter. The variable $S \in \mathbb{R}^{r\times r}$ is introduced so that $USV^\top$ may have a rank less than $r$. In (Keshavan et al., 2010), a similar approach without regularization is used. In (Boumal and Absil, 2011), the authors combine previous approaches by using a single variable on the Grassmann and regularizing only the entries outside of $\Omega$:

$$\begin{cases} \min\limits_{\mathcal{U},\mathcal{V},S} & \sum\limits_{(i,j)\in\Omega} \left((UV^\top)_{ij} - M_{ij}\right)^2 + \lambda \sum\limits_{(i,j)\notin\Omega} (UV^\top)_{ij}^2 \\ & \mathcal{U} \in \operatorname{Grass}(n,r) \\ & V \in \mathbb{R}^{s\times r}. \end{cases} \tag{3.9}$$

Boumal and Absil (2011) use the Riemannian trust-region (Algorithm 2) with a preconditioning, which uses an easily invertible approximation of the Hessian, to reach state-of-the-art performance. Balzano et al. (2010) present an online algorithm to estimate a subspace from partial measurements, which performs gradient descent steps on the Grassmann manifold. Their formulation reduces to (3.7) when applied to matrix completion. The approach is especially relevant for large instances, where one gets access to the observed entries of each column independently, in an online fashion. In (Eftekhari et al., 2019), the authors propose an online algorithm to conduct a principal component analysis from incomplete data which is based on the formulation

$$\begin{cases} \min\limits_{\mathcal{U},X} & \|X - \mathrm{P}_{\mathcal{U}}(X)\|_{\mathrm{F}}^2 \\ & \mathcal{U} \in \operatorname{Grass}(n,r) \\ & \mathcal{A}(X) = b \\ & X \in \mathbb{R}^{n\times s}, \end{cases} \tag{3.10}$$

where the orthogonal projection onto the subspace $\mathcal{U}$ is given by $\mathrm{P}_{\mathcal{U}} = UU^\top$ for some $U \in \operatorname{St}(n,r)$ such that $\operatorname{range}(U) = \mathcal{U}$. The objective is a least-squares residual which penalizes the columns of $X$ that do not belong to the r-dimensional subspace $\mathcal{U}$. Alternatively, the cost may be written $\left\|U^\perp X\right\|_{\mathrm{F}}^2$ where $U^\perp \in \operatorname{St}(n, n-p)$ spans the orthogonal complement of $\mathcal{U}$. We reconsider formulation (3.10) in Chapter 4 and adapt it to the nonlinear matrix recovery framework.

The performance of a low-rank matrix recovery algorithm consists in its runtime on benchmark problems as well as the measurement regime in which it is able to recover the desired matrix. These performances are expected to vary with the dimensions and the conditioning of the matrix to recover, which makes it difficult to draw general conclusions on the relative performances of the algorithms presented above. The following references include numerical comparisons of several state-of-the-art algorithms for low-rank matrix recovery (Boumal and Absil, 2015; Cambier and Absil, 2016; Nguyen et al., 2019).

## 3.2 Nonlinear matrix recovery

In this section, we introduce the nonlinear matrix recovery problem, the extension of low-rank matrix recovery to particular classes of high-rank matrices. Our results in Chapter 4 focus on solving the problem which is introduced here. We describe the concept of feature space and classes of high-rank matrices to which the framework applies. We consider three cases studies: algebraic varieties, unions of subspaces and clusters, that are introduced as Case studies 1, 2 and 3 below. Unions of subspaces are a particular type of algebraic variety which is particularly prevalent in applications (Eldar and Mishali, 2009; Elhamifar and Vidal, 2013). The algebraic variety model is also used in Chapter 5 for denoising and registration problems. Section 3.2.4 introduces clustering with missing information as a novel application of nonlinear matrix recovery, which uses the Gaussian kernel. Section 3.2.5 gives an overview of the related work on nonlinear matrix completion. As always, $M \in \mathbb{R}^{n\times s}$ denotes the original matrix to be recovered and $X$ is a generic variable in $\mathbb{R}^{n\times s}$.

### 3.2.1 Problem description and feature map

Traditional approaches to matrix completion described in the previous section do not apply if the matrix $M$ is high-rank. Recovering high-rank matrices requires making assumptions on the structure of $M$. To this end, let $m_1, \dots, m_s$ denote the columns of $M$. Low-rank matrix recovery methods can be applied when the points $m_i \in \mathbb{R}^n$ belong to a low-dimensional affine subspace in $\mathbb{R}^n$. Nonlinear matrix recovery attempts to recover $M$ when the columns $m_i$ of $M$ are related in a nonlinear way. This relies on a function that maps the columns of a matrix $X \in \mathbb{R}^{n\times s}$ to a higher-dimensional space (Fan and Chow, 2018). The *feature map* is defined as

$$\varphi\colon \mathbb{R}^n \to \mathcal{F}\colon v \mapsto \varphi(v),$$

where $\mathcal{F}$ is a Hilbert space. If $\mathcal{F}$ is finite dimensional, we write $\mathcal{F} = \mathbb{R}^N$ where $N$ is the dimension of the feature space, with $N \geq n$. We obtain the *feature matrix* $\Phi(X)$ by applying $\varphi$ to each column of $X$,

$$\Phi(X) = \begin{bmatrix}\varphi(x_1) & \dots & \varphi(x_s)\end{bmatrix} \in \mathbb{R}^{N\times s}. \tag{3.11}$$

The map $\varphi$ is chosen using a priori knowledge of the matrix $M$ so that the features of the data points $\varphi(m_i)$ for $i = 1, \dots, s$, all belong to the same affine subspace in $\mathbb{R}^N$. The nonlinear structure in $M$ causes a rank deficiency in the feature matrix $\Phi(M)$, even though $M$ may be full-rank, as illustrated in Figure 3.1.

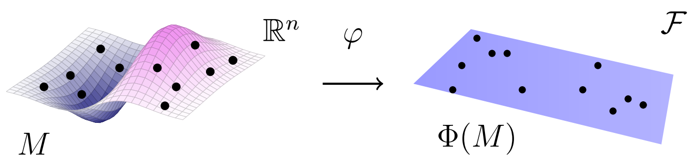


Figure 3.1: The feature map $\varphi$ is chosen to exploit the nonlinear structure.

If the features are infinite dimensional or that $N$ is very large, the feature map is represented using a kernel. This is known as the "kernel trick". The set $\mathcal{F}$ is then called a reproducing kernel Hilbert space. The kernel map represents the inner product between elements in the feature space $\mathcal{F}$:

$$k\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}\colon (x, y) \mapsto \langle \varphi(x), \varphi(y) \rangle_{\mathcal{F}}. \tag{3.12}$$

This allows to define the *kernel matrix* $\mathrm{K}(X, X) \in \mathbb{R}^{s \times s}$, by $\mathrm{K}(X, X)_{ij} = k(x_i, x_j)$. When $\mathcal{F}$ is finite dimensional, we have $\mathrm{K}(X, X) = \Phi(X)^\top \Phi(X)$ and therefore, $\operatorname{rank}(\mathrm{K}(X, X)) = \operatorname{rank}(\Phi(X))$ for all $X \in \mathbb{R}^{n \times s}$. Indeed, for any matrix $A \in \mathbb{R}^{N \times s}$, it holds that $\operatorname{rank}(A) = \operatorname{rank}(A^\top A)$, as $A$ and $A^\top A$ have the same null space. We use the term *lifting* to denote a kernel or a feature map, indistinctively.

For an appropriately chosen feature map, the nonlinear matrix recovery problem can be formulated as the rank minimization of the feature matrix subject to the measurements constraint (Equation (3.1)):

$$\begin{cases} \min\limits_{X \in \mathbb{R}^{n \times s}} & \operatorname{rank}(\Phi(X)) \\ & \mathcal{A}(X) = b. \end{cases} \tag{3.13}$$

This seeks to find a matrix which satisfies the measurements using a minimum number of independent features. This is an extension of Problem (3.2) for low-rank matrix recovery, where the map $\Phi$ is the identity. As is the case in low-rank problems, it is necessary to find a suitable approximation to the rank function. Section 3.2.5 describes approximations of (3.13) that have been proposed in recent related work.

### 3.2.2 The algebraic variety model

We introduce our first case study to which the nonlinear matrix recovery framework applies: algebraic varieties.

**Case study 1** (Algebraic variety model (Cox et al., 1994))**.** *Let $\mathbb{R}_d[x]$ be the set of real-valued polynomials of degree at most $d$ over $\mathbb{R}^n$. A real (affine)* algebraic variety *of degree $d$ is defined as the roots of a system of polynomials $P \subset \mathbb{R}_d[x]$:*

$$V(P) = \{x \in \mathbb{R}^n : p(x) = 0 \textit{ for all } p \in P\}.$$

*We say that the matrix $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_s \end{bmatrix} \in \mathbb{R}^{n\times s}$ follows an algebraic variety model if every column of $X$ belongs to the same algebraic variety, i.e., $x_i \in V(P)$ for all $i = 1, \dots, s$.*

Let

$$N(n,d) = \binom{n+d}{d} = \frac{(n+d)!}{d!\,n!}, \tag{3.14}$$

be the dimension of the space of polynomials in $n$ variables of degree at most $d$. The *monomial features* $\varphi_d$ for some degree $d$ are defined as

$$\varphi_d \colon \mathbb{R}^n \to \mathbb{R}^{N(n,d)} \colon \varphi_d(x) = \begin{pmatrix} x^{\boldsymbol{\alpha}^1} \\ x^{\boldsymbol{\alpha}^2} \\ \vdots \\ x^{\boldsymbol{\alpha}^N} \end{pmatrix} \tag{3.15}$$

where $\boldsymbol{\alpha}^i$ for $i = 1, 2, \dots, N(n,d)$ is a multi-index of nonnegative integers $(\alpha_1^i, \alpha_2^i, \dots, \alpha_n^i)$ so that $x^{\boldsymbol{\alpha}^i} := x_1^{\alpha_1^i} x_2^{\alpha_2^i} \dots x_n^{\alpha_n^i}$ and $\alpha_1^i + \alpha_2^i + \cdots + \alpha_n^i \leq d$. We obtain the matrix of monomial features $\Phi_d(X)$ by applying $\varphi_d$ to each column of $X \in \mathbb{R}^{n\times s}$,

$$\Phi_d(X) = \begin{bmatrix} \varphi_d(x_1) & \cdots & \varphi_d(x_s) \end{bmatrix} \in \mathbb{R}^{N\times s}.$$

The next example illustrates that the monomial features are rank deficient when the columns of a matrix belong to an algebraic variety.

**Example 3.1** (Monomial features). *Consider a set of $s$ points in $\mathbb{R}^2$ that satisfy one quadratic polynomial equation. Let*

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1s} \\ x_{21} & x_{22} & \cdots & x_{2s} \end{pmatrix} \in \mathbb{R}^{2\times s}$$

*satisfy for some coefficients $a = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \end{pmatrix}^\top$, the equation*

$$a_0 + a_1 x_{1i} + a_2 x_{2i} + a_3 x_{1i} x_{2i} + a_4 x_{1i}^2 + a_5 x_{2i}^2 = 0 \qquad \text{for all } i = 1, \dots, s.$$

*The monomial features of degree $d = 2$ applied to $X$ give*

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1s} \\ x_{21} & x_{22} & \cdots & x_{2s} \end{pmatrix} \mapsto \Phi_d(X) = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_{11} & x_{12} & \cdots & x_{1s} \\ x_{21} & x_{22} & \cdots & x_{2s} \\ x_{11}x_{21} & x_{12}x_{22} & \cdots & x_{1s}x_{2s} \\ x_{11}^2 & x_{12}^2 & \cdots & x_{1s}^2 \\ x_{21}^2 & x_{22}^2 & \cdots & x_{2s}^2 \end{pmatrix}.$$

*We observe that $\Phi_d(X)$ is rank deficient since $a^\top \Phi_d(X) = 0$.*

More generally, suppose the algebraic variety $V(P) \subset \mathbb{R}^n$ is defined by the set of linearly independent polynomials $P = \{p_1, \dots, p_q\}$, where each polynomial $p_i$ is at most of degree $d$. Let $x_1, \dots, x_s$ denote the columns of $X \in \mathbb{R}^{n \times s}$, then

$$x_i \in V(P) \text{ for all } i = 1, \dots, s \qquad \text{if and only if} \qquad \Phi_d(X)^\top C = 0, \tag{3.16}$$

where the columns of $C \in \mathbb{R}^{N \times q}$ define the coefficients of the polynomials $p_1, \dots, p_q$ in the monomial basis. Thus,

$$\operatorname{rank}(\Phi_d(X)) \le \min(N - q, s). \tag{3.17}$$

This ensures that $\Phi_d(X)$ is rank deficient when $X$ follows an algebraic variety model of degree $d$, provided that $s > N - q$.

The dimension of the lifted space $N(n, d)$ is approximately exponential in $d$, since $(n + d)!/n!d! \ge n^d/d!$. Therefore, a kernel implementation is usually used in practice for moderate and large values of $n$, or more precisely, whenever $s \le N(n, d)$. The monomial kernel of degree $d$ is defined as

$$\mathrm{K}_d \colon \mathbb{R}^{n \times s} \times \mathbb{R}^{n \times s} \to \mathbb{R}^{s \times s} \colon (X, Y) \mapsto (X^\top Y + c\mathbf{1}_{s \times s})^{\odot d}, \tag{Monomial kernel}$$

where the value $c \in \mathbb{R}$ is a parameter of the kernel, $\mathbf{1}_{s\times s}$ is a square matrix of size $s$ full of ones and $\odot$ is an entry-wise exponent. If the equations describing the algebraic variety are known to be homogeneous, one can set $c = 0$. Note that the monomial kernel satisfies $\mathrm{K}_d(X, X) = \tilde{\Phi}_d(X)^\top \tilde{\Phi}_d(X)$ for map of monomials $\tilde{\Phi}_d$ with non-unitary coefficients. Therefore, $\Phi_d(X)$ and $\mathrm{K}_d(X, X)$ have the same rank.

### 3.2.3 Union of subspaces

The second case study presents a union of subspaces as a particular type of algebraic variety.

**Case study 2** (Union of subspaces)**.** *Given two affine subspaces $\mathcal{S}_1$, $\mathcal{S}_2 \subset \mathbb{R}^n$ of dimension $r_1$ and $r_2$ respectively, we can write $\mathcal{S}_1 = \{x \in \mathbb{R}^n : q_i(x) = 0 \text{ for } i = 1, \dots, n - r_1\}$ and $\mathcal{S}_2 = \{x \in \mathbb{R}^n : p_j(x) = 0 \text{ for } j = 1, \dots, n - r_2\}$ where the $q_i$ and $p_j$ are affine functions. The union $\mathcal{S}_1 \cup \mathcal{S}_2$ can be expressed as the set where all possible products $q_i(x)p_j(x)$ vanish. Therefore, $\mathcal{S}_1 \cup \mathcal{S}_2$ is the solution of a system of $(n - r_1)(n - r_2)$ quadratic polynomial equations. Similarly, a union of $\ell$ affine subspaces of dimensions $r_1, \dots, r_\ell$ is an algebraic variety described by a system of $\Pi_{i=1}^{\ell}(n - r_i)$ polynomial equations of degree $\ell$.*

The rank of the monomial features is difficult to estimate a priori. It depends on the degree $d$ and the intrinsic dimension of the variety. The following result gives an upper bound when the matrix $X$ follows the union of subspaces model.

**Proposition 3.1** (Rank of monomial features (Ongie et al., 2017))**.** *If the columns of a matrix $X \in \mathbb{R}^{n\times s}$ belong to a union of $\ell$ affine subspaces each of dimension at most $\kappa$, then for all $d \geq 1$, the matrix $\Phi_d(X) \in \mathbb{R}^{N(n,d)\times s}$ of monomial features satisfies*

$$\operatorname{rank} \Phi_d(X) \leq \ell \binom{\kappa + d}{d},$$

*with $N(n,d)$ the dimension of the feature space* (3.14)*.*

The next example shows the rank of the monomial features and kernel for a matrix whose columns belong to a union of subspaces.

**Example 3.2.** *Consider $X \in \mathbb{R}^{10\times 100}$ following a union of subspaces model (Case study 2). The columns of $X$ are divided across 5 subspaces of dimension 2 in $\mathbb{R}^{10}$, with 20 points on each subspace. Hence, the matrix $X$ has full rank,* $\operatorname{rank}(X) = 10$*. Computing the monomial features of degree $d = 2$ gives $\Phi_2(X) \in \mathbb{R}^{66\times 100}$ with* $\operatorname{rank}(\Phi_2(X)) = 26$*. The monomial kernel $\mathrm{K}_2(X, X) \in \mathbb{R}^{100\times 100}$ has the same rank. Figure 3.2 shows the singular values of $\Phi_2(X)$ and $\mathrm{K}_2(X, X)$ in logarithmic scale. The bound from Proposition 3.1 applied to this example gives* $\operatorname{rank}(\Phi_2(X)) \leq 30$.

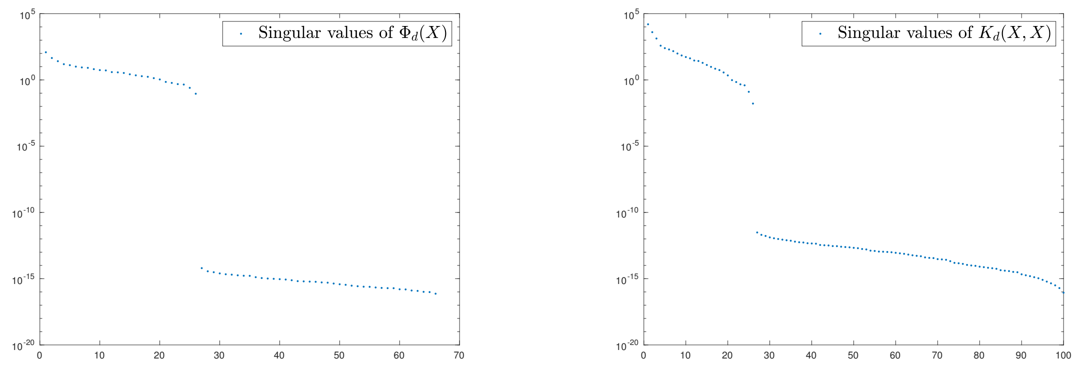


Figure 3.2: Singular values of the monomial maps for a union of subspaces

**Choice of the degree $d$** Choosing the degree $d$ of the monomial kernel for a given problem is a nontrivial task. In Section 4.6, we discuss the practical choice of this degree, how it impacts the rank of the feature matrix and the possibility to recover a matrix whose columns belong to a union of subspaces. Previous works using the monomial kernel to recover high-rank matrices all restricted themselves to degrees two or three (Ongie et al., 2017; Fan and Chow, 2018). Using monomial features of a larger degree seems helpful to capture all the nonlinearity in some data sets. In particular, the union of $d$ subspaces is an algebraic variety of degree $d$. Unfortunately, increasing the degree implies that the dimension of the optimization problem grows exponentially. Indeed, the lifted dimension $N(n,d) \approx n^d/d!$ blows up with $d$ for even moderate values of $n$. Using the monomial kernel, which belongs to $\mathbb{R}^{s\times s}$, does not solve this issue entirely, even though the dimension $N(n,d)$ does not appear explicitly. Recovery of $M \in \mathbb{R}^{n\times s}$ requires $\Phi_d(M)$ to be rank deficient, which only happens if $s > N(n,d) - q$, where $q$ is the number of polynomial equations that define the algebraic variety (Equation (3.17)). If $\operatorname{rank}(\Phi_d(M)) = N(n,d)$, for any $s > N(n,d)$, the monomial kernel is rank deficient at $M$ but also at every $X \in \mathbb{R}^{n\times s}$, which gives no hope of recovery. In short, choosing a larger degree increases the dimension of the feature space $N(n,d)$, which in turn increases the number of columns necessary for the recovery to be possible, in order to satisfy $s > N(n,d) - q$.

### 3.2.4 Clustering and the Gaussian kernel

We now define the Gaussian kernel, whose use for clustering with missing data is novel in the context of nonlinear matrix recovery.

**Case study 3** (Clusters). *For $X, Y \in \mathbb{R}^{n\times s}$, the entry $(i,j)$ of the Gaussian kernel $\mathrm{K}_\sigma\colon \mathbb{R}^{n\times s} \times \mathbb{R}^{n\times s} \to \mathbb{R}^{s\times s}$ is defined as*

$$\mathrm{K}_\sigma(X,Y)_{ij} = e^{-\frac{\|x_i - y_j\|_2^2}{2\sigma^2}}, \qquad \text{(Gaussian kernel)}$$

*where the parameter $\sigma > 0$ is the width of the kernel. The Gaussian kernel acts as a proximity measure. For two columns of $X \in \mathbb{R}^{n\times s}$, labelled $x_i$ and $x_j$, we observe that $\mathrm{K}_\sigma(X,X)_{ij} \approx 1$ if $x_i$ is close to $x_j$, and $\mathrm{K}_\sigma(X,X)_{ij} \approx 0$ if $x_i$ is far from $x_j$. Therefore, the Gaussian kernel is close to a low-rank matrix whose rank is the number of clusters formed by the columns of $X$, as illustrated in Figure 3.3. The value of $\sigma$ should be chosen appropriately depending on the size of the clusters (Singer, 2006). The Gaussian kernel is a representation of features that belong to an infinite dimensional Hilbert space.*

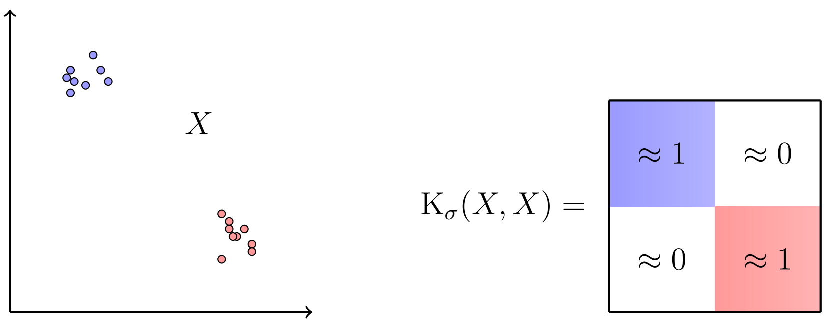


Figure 3.3: The Gaussian kernel on clustered data

In Figure 3.4, we present a randomly generated data set $X \in \mathbb{R}^{2\times 100}$ divided into two clusters, along with the singular values of the Gaussian kernel in logarithmic scale. We see that the two largest singular values are much greater than the third one, and that the following singular values decay at an approximately exponential rate. This behaviour is observed for an arbitrary number of clusters, namely, $\sigma_{n_c}(\mathrm{K}_\sigma(M)) \gg \sigma_{n_c+1}(\mathrm{K}_\sigma(M))$ for $n_c$ clusters. In Case study 1 and 2, the rank of the monomial features can be unambiguously determined by looking at the graph of singular values. In the case of clusters, the rank of the Gaussian kernel is better replaced by the $\varepsilon$-rank of a matrix, as defined in (Golub and Van Loan, 1996, Section 2.5.5) by

$$\operatorname{rank}_\varepsilon(A) = \min_{\|A-B\|_2 \le \varepsilon} \operatorname{rank}(B). \tag{3.18}$$

The $\varepsilon$-rank of a matrix $A \in \mathbb{R}^{s\times s}$ can be determined using the fact that

$$\sigma_1(A) \ge \sigma_{\operatorname{rank}_\varepsilon(A)} > \varepsilon \ge \sigma_{\operatorname{rank}_\varepsilon(A)+1} \ge \cdots \ge \sigma_s(A).$$

The example in Figure 3.4 gives convincing evidence that when the columns of $X$ are divided in $n_c$ clusters, the matrix $\mathrm{K}_\sigma(X,X)$ has $\varepsilon$-rank equal to $n_c$ for an appropriate value of $\varepsilon$ (around one in the figure). For low-rank matrix completion, Vandereycken (2013) describes a rank adaptive strategy to recover matrices with exponentially decaying singular values, which we apply to the kernel matrix $\mathrm{K}_\sigma$ in Chapter 4. In (Fan and Cheng, 2018), the Gaussian kernel was also used on image inpainting and image denoising problems.

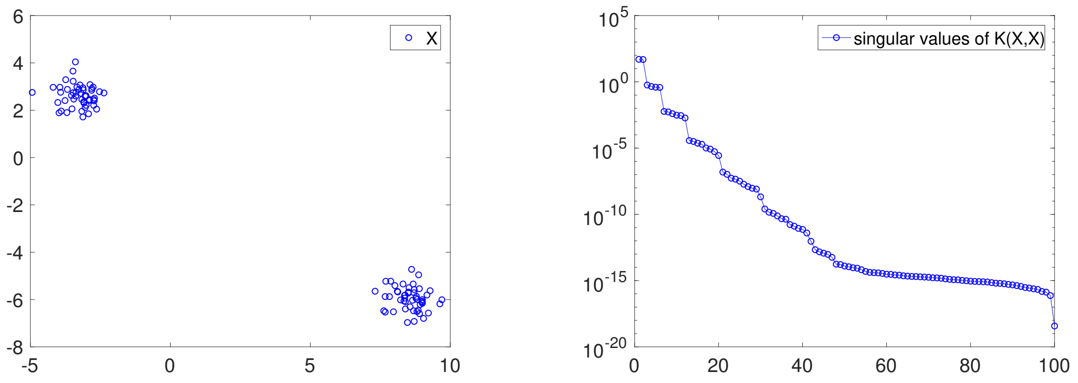


Figure 3.4: Clustered data and the singular values of the Gaussian kernel in log-scale.

### 3.2.5 Related work

We survey the existing literature on nonlinear matrix recovery, and begin in chronological order. Consider a matrix $M \in \mathbb{R}^{n\times s}$ whose columns as vectors in $\mathbb{R}^n$ belong to a union of

$\ell$ linear subspaces, each of dimension at most $\kappa$ strictly smaller than $n$. Let us consider the problem of recovering $M$ from a subset of its entries. This setting generalizes low-rank matrix completion where $\ell = 1$ and all the columns of $M$ belong to a single subspace. In the case of $\ell$ subspaces, it is expected that $\operatorname{rank}(M) = \ell\kappa$. It is known that low-rank matrix completion is proven to be successful at recovering a rank $r$ matrix from only $\mathcal{O}(rs\log^2 s)$ entries (assuming $n \leq s$) (Recht, 2011), which becomes $Cs\ell\kappa\log^2(s)$ for a union of subspaces. This means that nearly all the entries of the matrix are required if $M$ is high-rank, i.e., $\ell\kappa$ is close to $n$. Applying low-rank matrix completion methods is therefore inefficient when the number of subspaces $\ell$ is large.

In (Eriksson et al., 2012), the authors propose an algorithm that exploits the union of subspace structure of the problem described above. In several steps, they identify the subspaces by localized low-rank matrix completion subproblems. They show that each column of $M$ can be recovered with high probability if the number of entries observed is greater than $Cs\kappa\log^2(n)$. The constant $C > 1$ depends on an incoherence condition and the geometrical arrangement of the subspaces. This is better than the necessary $Cs\ell\kappa\log^2(s)$ which low-rank matrix completion requires, especially when the number of subspaces $\ell$ is large.

The recovery of a matrix whose columns belong to a union of subspaces is equivalent to *subspace clustering* with missing data. Given a matrix $X \in \mathbb{R}^{n\times s}$ for which each column $x_i \in \mathbb{R}^n$ belongs to one of $\ell$ subspaces in $\mathbb{R}^n$, subspace clustering consists in learning the subspaces from $X$ and assigning each $x_i$ to its corresponding subspace. With access to all the entries in $X$, subspace clustering can be solved with high probability (Lerman et al., 2011).

The concept of completing a high-rank matrix was generalized simultaneously in (Ongie et al., 2017) and (Fan and Chow, 2018) with the introduction of a feature map. In (Ongie et al., 2017), the applications go beyond union of subspaces as the authors consider matrices whose columns belong to an algebraic variety. In essence, Fan and Chow (2018) and Ongie et al. (2017) apply different minimization algorithms to

$$\begin{cases} \min\limits_{X} & \|\Phi_d(X)\|_{\mathcal{S}_p}^p \\ & X_{ij} = M_{ij},\ (i,j) \in \Omega, \end{cases} \tag{3.19}$$

where $\|.\|_{\mathcal{S}_p}$ is the Schatten-p quasi-norm defined as

$$\|\Phi_d(X)\|_{\mathcal{S}_p} = \left( \sum_{i=1}^{\min(N,s)} \sigma_i(\Phi_d(X))^p \right)^{1/p} \qquad \text{for } 0 < p \leq 1.$$

When $p = 1$, the sum of the singular values is the nuclear norm, which extends Problem (3.4) to the nonlinear case. The Schatten p-norm for $0 < p \leq 1$ is nonsmooth. This has the benefit of encouraging sparsity in the singular values, but it might prevent fast convergence near a minimizer. Both Fan and Chow (2018) and Ongie et al. (2017) use a kernel representation of the features, so that the features are never computed explicitly. In (Fan and Cheng, 2018), a quasi-Newton method is used to minimize (3.19).

Ongie et al. (2017) propose a nonlinear extension of an iterative reweighted least squares scheme, introduced for low-rank matrix completion in (Mohan and Fazel, 2012). Using the fact that

$$\|\Phi_d(X)\|_{\mathcal{S}_p} = \operatorname{trace}\left(\mathrm{K}_d(X,X)W\right) \text{ with } W = \mathrm{K}_d(X,X)^{\frac{p}{2}-1}, \tag{3.20}$$

the strategy is to approximately minimize (3.20) where $W$ is treated as a constant, then update $W$ separately, and repeat the process until convergence. This gives, for $k = 1, 2, \ldots$

$$\begin{aligned} W_k &= (\mathrm{K}_d(X_k, X_k) + \gamma_k \mathrm{I})^{\frac{p}{2}-1} \\ X_{k+1} &= \arg\min_X \operatorname{trace}(\mathrm{K}_d(X,X)W_k) \qquad \text{such that} \qquad X_{ij} = M_{ij},\ (i,j) \in \Omega, \end{aligned} \tag{3.21}$$

where $\gamma_k$ is a smoothing parameter, which is included to improve numerical stability; this is equivalent to minimizing a smooth approximation of the Schatten-$p$ norm (Mohan and Fazel, 2012). At each iteration, Ongie et al. (2017) performing one gradient step to approximately minimize (3.21). The authors present numerical results where this algorithm is applied to synthetic data, and demonstrate that their algorithm significantly outperforms traditional low-rank matrix completion methods to recover matrices whose columns belong to a union of subspaces. They numerically characterize the sampling regime and number of subspaces for which their algorithm gives successful recovery.

In (Fan et al., 2019) a truncated version of the Schatten norm is proposed, where only the smallest singular values are minimized:

$$\left( \sum_{i=r+1}^{\min(N,s)} \sigma_i(\Phi_d(X))^p \right)^{1/p} \qquad\qquad \text{for } 0 < p \leq 1,$$

where $r = \operatorname{rank}(\Phi_d(M))$. They use the kernel trick and propose an algorithm which alternates between a truncated singular value decomposition of the kernel matrix and a step of the stochastic gradient method. In (Fan et al., 2020), the authors propose an extension to handle outliers in the data. This is achieved by decomposing $M$ as the sum of a matrix who follows a union of subspaces model and a sparse matrix which absorbs the outliers. The method proposed in (Fan and Cheng, 2018) replaces the monomial kernel by a deep neural network who is trained to minimize the reconstruction error for the observable entries of $M$. This work showcases the applicability and performance of nonlinear matrix completion with numerous examples including image inpainting and collaborative filtering problems. For data drawn from multiple subspaces, Fan et al. (2018) propose a sparse factorization where each subspace is represented in a low rank decomposition. They solve this model with an alternating minimization algorithm in the spirit of the Proximal Alternating Linearized Minimization (Bolte et al., 2013). In (Fan and Udell, 2019), the authors propose a kernel factorization algorithm for matrix completion, which lends itself to online completion. In this setting, the columns of the matrix $M$ are accessible as a stream and the matrix $M$ is never stored in its entirety. They also develop a variant to deal with out of samples extensions, that is, how to complete a new column without recomputing the model. The offline formulation applies the kernel trick to

$$\begin{cases} \min\limits_{X,D,Z} & \|\Phi_d(X) - \Phi_d(D)Z\|_{\mathrm{F}}^2 + \alpha \|\Phi_d(D)\|_{\mathrm{F}}^2 + \beta \|Z\|_{\mathrm{F}}^2 \\ & X_{ij} = M_{ij},\ (i,j) \in \Omega, \\ & D \in \mathbb{R}^{n\times r}, Z \in \mathbb{R}^{r\times s}, X \in \mathbb{R}^{n\times s}. \end{cases}$$

The variable $D \in \mathbb{R}^{n\times r}$ aims to find $r$ points in $\mathbb{R}^n$ whose features form a basis for $\Phi_d(M)$ in the feature space. The last two terms in the objective are added as regularizers to improve the practical performances of the algorithm, as is common in low-rank matrix completion (Davenport and Romberg, 2016).

In (Ongie et al., 2021), the authors build a tensor representation of the data and apply known matrix completion techniques in the tensor space. For this algorithm, they are able to show that the sampling requirements nearly match the information theoretic lower bounds for recovery under a union of subspace model. This is remarkable as the sampling pattern in the tensor space is not random, and low-rank recovery results do not apply directly. Note that the approach in (Ongie et al., 2021) is only applicable to matrix completion problems, not matrix sensing.

In the next chapter, we aim to contribute to this literature by looking at new problem formulations, novel applications of algorithms to this problem and new use cases.

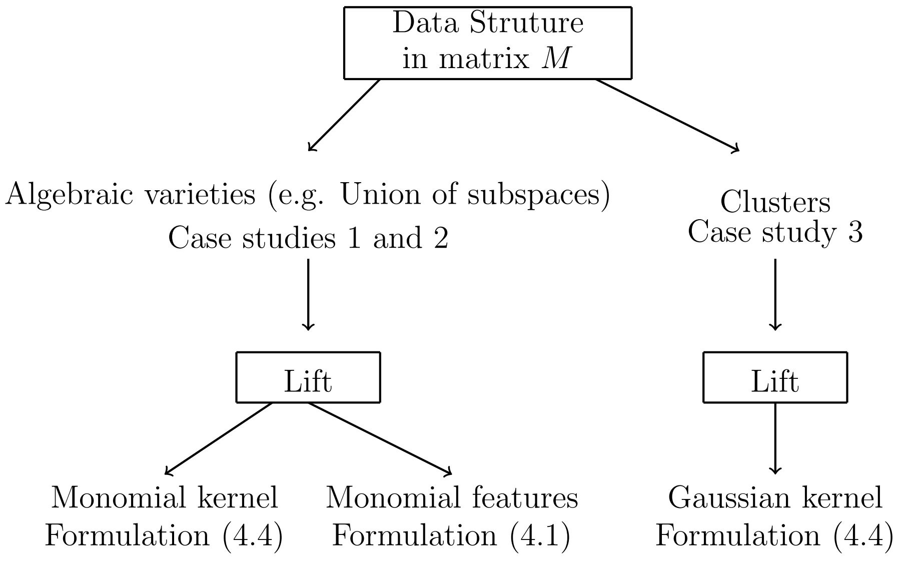
# Chapter 4

# Nonlinear matrix recovery on the Grassmann manifold

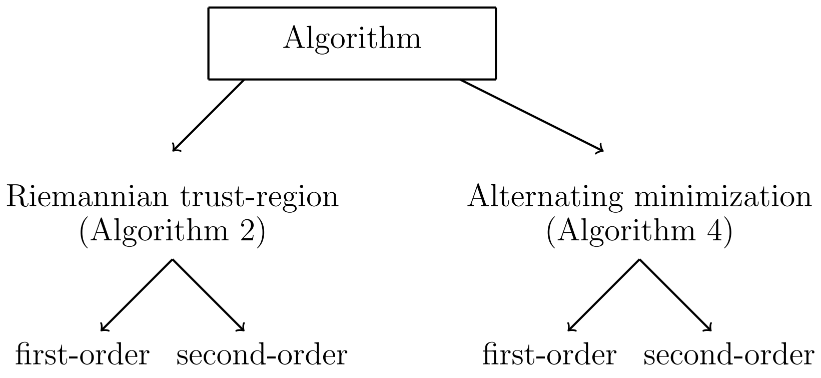
Our work in this chapter is based on the recent discovery that an extension of traditional methods for low-rank matrix recovery allows to recover specific classes of high-rank matrices. This problem, known as *nonlinear matrix recovery* (or high-rank matrix recovery), is introduced in Section 3.2 where the concepts of feature space and case studies are presented. The methods in this chapter are general, and in particular applicable to Case studies 1, 2 and 3, which refer to matrices whose columns belong to an algebraic variety, a union of subspaces and several clusters, respectively.

In Section 4.1, we propose a new formulation to solve the nonlinear matrix recovery problem. The rank minimization in (3.13) is replaced by a nonconvex constrained optimization problem on the Grassmann manifold. This extends the residual proposed by Eftekhari et al. (2019) in the context of low-rank matrix completion to the nonlinear case. The derivatives of the cost function for this residual and specific kernel maps are computed in Section 4.1.2. In Section 4.2, we propose to use Riemannian optimization methods to solve the recovery problem, which is new in the context of nonlinear matrix recovery. Riemannian optimization, as described in Chapter 2, provides a framework to design algorithms for problems with smooth constraints. This allows to seamlessly choose between standardized first- and second-order methods. The use of second-order methods allows to recover high-rank matrices up to high accuracy if desired. Section 4.3 presents an alternating minimization algorithm to solve the recovery problem. First- and second-order variants of the alternating minimization are discussed. We prove global convergence of the alternating minimization to first-order stationary points in Section 4.4 and give a global complexity bound on the number of iterations necessary to achieve an arbitrary accuracy on the gradient norm from an arbitrary initial guess. In Section 4.4.2, we also show convergence of the sequence of iterates to a unique limit point using the Kurdyka-Lojasiewicz property. Our alternating minimization method is a similar approach to the method proposed in (Fan et al., 2019), which does not provide a convergence analysis. We conclude with an extensive set of numerical experiments that compare the performances of the optimization approaches and the quality of the solutions that can be obtained (Section 4.6). We discuss the influence of the complexity of the data and the role of model parameters on the recovery. Moreover, we showcase that our approach is efficient at clustering with missing information (a novel application of nonlinear matrix recovery) and dealing with noisy measurements.

We summarize the different elements which compose the nonlinear matrix recovery problem in the next two diagrams. This acts as a roadmap of the chapter. When the matrix $M$ to be recovered belongs to Case study 1 or 2, we use the monomial features or monomial kernel. When the matrix $M$ belongs to Case study 3, we use the Gaussian kernel. When a feature map is used, we solve problem formulation (4.1), introduced in Section 4.1. If a kernel is used instead, we solve problem (4.4).


Data Struture in matrix $M$
Algebraic varieties (e.g. Union of subspaces) Case studies 1 and 2
Clusters Case study 3
Lift
Lift
Monomial kernel Formulation (4.4)
Monomial features Formulation (4.1)
Gaussian kernel Formulation (4.4)

In addition, one needs to choose an algorithm to solve (4.1) or (4.4). We propose two families of algorithms: Riemannian trust-regions (Algorithm 2) and Alternating minimizations (Algorithm 4). Each of these algorithms have first- and second-order variants.


Algorithm
Riemannian trust-region (Algorithm 2)
Alternating minimization (Algorithm 4)
first-order
second-order
first-order
second-order

## 4.1 Nonlinear matrix recovery as an optimization problem

We continue from the problem description in Section 3.2.1 and present a new formulation to solve the nonlinear matrix recovery problem. The notations and set up here are precisely as in Section 3.2.1, we use $\Phi(X)$ to denote a feature matrix (Equation (3.11)) and the linear measurements on a matrix $M \in \mathbb{R}^{n\times s}$ are represented by an operator $\mathcal{A}\colon \mathbb{R}^{n\times s} \to \mathbb{R}^m$ and vector $b \in \mathbb{R}^m$ (Equation (3.1)), in which the matrices $A_1, \dots, A_m \in \mathbb{R}^{n\times s}$ may represent matrix completion or dense matrix sensing.

**Noiseless measurements case** Consider Equation (3.13), the rank minimization of the feature matrix under exact measurement constraints. We assume that $r = \operatorname{rank}(\Phi(M))$ is known and smaller than $\min(N, s)$. As a nonconvex approximation to (3.13), we consider formulation (3.10) based on (Balzano et al., 2010; Boumal and Absil, 2015; Eftekhari et al., 2019), designed for online low-rank matrix completion, and extend it to the nonlinear case. This leads to the following formulation:

$$\begin{cases} \min\limits_{X,\mathcal{U}} & f(X,\mathcal{U}) := \|\Phi(X) - \mathrm{P}_{\mathcal{U}}\Phi(X)\|_{\mathrm{F}}^2 \\ & \mathcal{U} \in \operatorname{Grass}(N, r) \\ & \mathcal{A}(X) = b, \end{cases} \tag{4.1}$$

where $\operatorname{Grass}(N, r)$ is the Grassmann manifold, the set of all subspaces of dimension $r$ in $\mathbb{R}^N$, $\mathrm{P}_{\mathcal{U}}$ is the orthogonal projection on the subspace $\mathcal{U}$ and $\|.\|_{\mathrm{F}}$ denotes the Frobenius norm. Given $U \in \operatorname{St}(N, r)$ such that $\operatorname{range}(U) = \mathcal{U}$, the projection is given by $\mathrm{P}_{\mathcal{U}} = UU^\top$. The residual in (4.1) attempts to find an $r$-dimensional subspace in $\mathbb{R}^N$ such that the span of $\Phi(X)$ belongs to that subspace, effectively limiting the rank of $\Phi(X)$ to $r$. In (4.1), the objective function is expected to be nonconvex but smooth for practical choices of $\Phi$, such as Case study 1, 2 and 3. If the variable $\mathcal{U}$ is additionally constrained to be the range of the $r$ leading singular vectors of $\Phi(X)$, the cost function becomes $\sum_{i=r+1}^{\min(N,s)} \sigma_i(\Phi(X))^2$. It is interesting to wonder how the formulation performs if the rank is over-estimated – be it purposefully or not. Intuitively, estimating the subspace $\mathcal{U}$ from the $r + r' > r$ leading singular vectors of $\Phi(X)$ could bring some robustness. In the numerical section 4.6, we try overestimating the rank with disappointing results, which indicates that such approach would probably require more care to perform well. The advantage of the formulation in (4.1) over using singular values is that is it straightforward to express the former as a finite sum of $s$ terms over the data points. This leaves open the opportunity to use stochastic subsampling algorithms, which would be helpful in applications with a very large number of samples.

The advantage of using the Grassmann manifold, which is a quotient space, instead of the Stiefel manifold of orthogonal matrices $\mathrm{St}(N,r) := \{U \in \mathbb{R}^{N\times r} : U^\top U = \mathrm{I}_r\}$ is that due to the invariance of the cost function with respect to the matrix that represents the subspace $\mathcal{U}$, local optimizers cannot possibly be isolated in a formulation over $\mathrm{St}(N,r)$. Therefore, the fast local convergence rates of some second-order algorithms might not apply on $\mathrm{St}(N,r)$, while they would apply on the quotient manifold.

Consider $U^\perp$ a basis of $\mathcal{U}^\perp$, the orthogonal complement of $\mathcal{U}$ in $\mathbb{R}^N$. The variable $\mathcal{U}^\perp \in \mathrm{Grass}(N, N-r)$ has a nice interpretation since, if $f(X,\mathcal{U}) = 0$, then $U^\perp$ spans $\mathrm{null}(\Phi(X)^\top)$, the null space of $\Phi(X)^\top$. In the case of algebraic varieties (Case study 1), vectors in $\mathrm{null}(\Phi(X)^\top)$ give the coefficients of polynomials defining the algebraic variety in the basis given by $\Phi$. Recovering the equations of the variety is of interest in some applications, and we use that observation in Chapter 5.

**Noisy measurements case** When the measurements are known to be noisy, it is advisable to relax the constraint $\mathcal{A}(X) = b$ to allow for some misfit. We place the measurement constraint into the cost function as a penalty. This gives

$$\begin{cases} \min\limits_{X,\mathcal{U}} & f_\lambda(X,\mathcal{U}) := \|\Phi(X) - \mathrm{P}_{\mathcal{U}}\Phi(X)\|_\mathrm{F}^2 + \lambda\, \|\mathcal{A}(X) - b\|_2^2 \\ & \mathcal{U} \in \mathrm{Grass}(N,r), \end{cases} \tag{4.2}$$

where the parameter $\lambda > 0$ has to be adjusted. This allows to satisfy the measurements approximately.

## 4.1.1 Kernel representation of the features

When using a kernel to represent the feature map, as shown in (3.12), we want to find a cost function equivalent to (4.1) which uses the kernel matrix instead of the feature matrix. We find that they are related in the following way.

**Proposition 4.1.** *Given a feature map $\Phi$ and the associated kernel $\mathrm{K}\colon X \times X \mapsto \Phi(X)^\top\Phi(X)$, for $\mathcal{W} \in \mathrm{Grass}(s,r)$ we have*

$$\left\|\Phi(X)^\top - \mathrm{P}_{\mathcal{W}}\Phi(X)^\top\right\|_\mathrm{F}^2 = \mathrm{trace}\Big(\mathrm{K}(X,X) - \mathrm{P}_{\mathcal{W}}\mathrm{K}(X,X)\Big). \tag{4.3}$$

*Proof.* We write $\mathrm{P}_{\mathcal{W}^\perp} = \mathrm{I}_{s\times s} - \mathrm{P}_{\mathcal{W}}$ and find

$$\begin{aligned} \mathrm{trace}\Big(\mathrm{P}_{\mathcal{W}^\perp} K(X,X)\Big) &= \mathrm{trace}\Big(\mathrm{P}_{\mathcal{W}^\perp}\Phi(X)^\top\Phi(X)\Big) \\ &= \mathrm{trace}\Big(\Phi(X)\mathrm{P}_{\mathcal{W}^\perp}\Phi(X)^\top\Big) \\ &= \mathrm{trace}\Big(\Phi(X)\mathrm{P}_{\mathcal{W}^\perp}^\top\mathrm{P}_{\mathcal{W}^\perp}\Phi(X)^\top\Big) \\ &= \mathrm{trace}\Big((\mathrm{P}_{\mathcal{W}^\perp}\Phi(X)^\top)^\top\mathrm{P}_{\mathcal{W}^\perp}\Phi(X)^\top\Big) \\ &= \left\|\mathrm{P}_{\mathcal{W}^\perp}\Phi(X)^\top\right\|_\mathrm{F}^2. \end{aligned}$$

□

Using the kernel formula (4.3) corresponds to finding a subspace $\mathcal{W}$ of dimension $r$ which contains the row space of $\Phi(X)$. When a kernel is used, the following optimization problem is solved,

$$\begin{cases} \min\limits_{X,\mathcal{W}} & f(X,\mathcal{W}) := \operatorname{trace}(\mathrm{K}(X,X) - \mathrm{P}_{\mathcal{W}}\mathrm{K}(X,X)) \\ & \mathcal{W} \in \operatorname{Grass}(s,r) \\ & \mathcal{A}(X) = b. \end{cases} \tag{4.4}$$

Replacing the features $\Phi$ by the corresponding kernel K becomes beneficial when the dimension $N$ of the features is larger than the number of points $s$ and when a convenient formula is available to compute the kernel and its derivatives. For example, in the case of clusters (Case study 3), the features exist implicitly in an infinite dimensional space and we use the Gaussian kernel to represent them.

In the upcoming sections, we usually describe the algorithms and their properties using the notation of problem (4.1) with a feature map $\Phi$ and cost function $f(X,\mathcal{U})$. Unless specified otherwise, the developments also apply to problem (4.4) and the use of a kernel.

### 4.1.2 Derivatives of the cost function

In this section, we compute the (Euclidean) derivatives of the cost function in Problem (4.4). For the monomial kernel, we compute first- and second-order derivatives. For the Gaussian kernel, we compute the first-order derivative. To compute the derivative of a matrix-valued function, we write a Taylor expansion and identify the gradient by looking at first-order terms. The proofs can be found in Appendix A.1.

**Proposition 4.2.** *For the monomial kernel* $\mathrm{K}_d$ *(Case study 1 and 2), the Euclidean gradient of the cost function in* (4.4) *is given by*

$$\nabla_X f(X,W) = 2dX\left(\mathrm{K}_{d-1}(X) \odot \mathrm{P}_{W_\perp}\right) \qquad \text{and} \qquad \nabla_W f(X,W) = -2\mathrm{K}_d(X)W.$$

*For* $(\Delta_X, \Delta_W) \in \mathbb{R}^{n\times s} \times \mathbb{R}^{s\times r}$*, the application of the Hessian is given by*

$$\nabla^2 f(X,W)\begin{bmatrix}\Delta_X \\ \Delta_W\end{bmatrix} = \begin{pmatrix}\nabla_X^2 f(X,W)[\Delta_X] + \nabla_W\nabla_X f(X,W)[\nabla_W] \\ \nabla_X\nabla_W f(X,W)[\Delta_X] + \nabla_W^2 f(X,W)[\Delta_W]\end{pmatrix}.$$

*where*

$$\begin{aligned}
\nabla_W^2 f(X,W)[\Delta_W] &= -2\mathrm{K}_d(X)\Delta_W \\
\nabla_X^2 f(X,W)[\Delta_X] &= 2d(d-1)X\left(\mathrm{K}_{d-2}(X) \odot (X^\top\Delta_X + \Delta_X^\top X) \odot \mathrm{P}_{W_\perp}\right) + 2d\Delta_X\left(\mathrm{K}_{d-1}(X) \odot \mathrm{P}_{W_\perp}\right) \\
\nabla_X\nabla_W f(X,W)[\Delta_X] &= -2d\left(\mathrm{K}_{d-1}(X) \odot (X^\top\Delta_X + \Delta_X^\top X)\right)W \\
\nabla_W\nabla_X f(X,W)[\Delta_W] &= -2dX\left(\mathrm{K}_{d-1}(X) \odot (W\Delta_W^\top - \Delta_W W^\top)\right).
\end{aligned}$$

**Proposition 4.3.** *For the Gaussian kernel* $\mathrm{K}_\sigma$ *(Case study 3), the Euclidean gradient of the cost function in* (4.4) *is given by*

$$\nabla_X f(X,W) = -\frac{2}{\sigma^2} X \left(\operatorname{diag}\left(\operatorname{sum}(\mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}, 1)\right) - \mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}\right) \quad \textit{and} \quad \nabla_W f(X,W) = -2\mathrm{K}_\sigma(X)W.$$

*where* $\odot$ *denotes an entry-wise product and* $\operatorname{sum}(\mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}, 1)$ *is the vector whose entries are the sum of each column of the matrix* $\mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}$.

We do not compute the Hessian for the Gaussian kernel. We either use automatic differentiation (in Python) or finite differences of the gradient (in Matlab).

## 4.2 Riemannian optimization algorithms

In this section, we investigate the use of Riemannian optimization methods to solve (4.1) and (4.4). Riemannian optimization methods, which are introduced in Chapter 2, have proved to be efficient in low-rank matrix completion problems (Vandereycken, 2013; Boumal and Absil, 2015). In order to formally express (4.1) as a Riemannian optimization problem, we define a notation for the affine subspace that represents the measurements on the matrix $M$,

$$\mathrm{L}_{\mathcal{A},b} = \{X \in \mathbb{R}^{n\times s} : \mathcal{A}(X) = \mathcal{A}(M) = b\}.$$

We form the product manifold

$$\mathcal{M} = \mathrm{L}_{\mathcal{A},b} \times \operatorname{Grass}(N,r), \tag{4.5}$$

so that Problem (4.1) can be viewed as the unconstrained minimization of a smooth cost function defined on the manifold $\mathcal{M}$,

$$\begin{cases} \min\limits_{(X,\mathcal{U})} & \|\Phi(X) - \mathrm{P}_{\mathcal{U}}\Phi(X)\|_{\mathrm{F}}^2 \\ & (X,\mathcal{U}) \in \mathcal{M}. \end{cases} \tag{4.6}$$

We introduce the notation $z := (X,\mathcal{U}) \in \mathcal{M}$ to denote the pair of variables that appear in the optimization problem. The geometry of the set $\mathrm{L}_{\mathcal{A},b}$ is rather trivial because it is affine. Section 2.9.2 describes that $\mathrm{L}_{\mathcal{A},b}$ can be viewed as a Riemannian manifold, which gives a straightforward way to implement optimization methods with affine constraints. The Riemannian manifold structure of $\operatorname{Grass}(N,r)$ is covered in Section 2.9.5. We briefly recall some of these elements in order to define some notations. At $\mathcal{U} \in \operatorname{Grass}(N,r)$, choose some $U \in \operatorname{St}(N,r)$ such that $\operatorname{range}(U) = \mathcal{U}$. The tangent space $\mathrm{T}_{\mathcal{U}}\operatorname{Grass}(N,r)$ can be represented by the horizontal space, denoted by

$$\mathrm{T}_U\operatorname{Grass}(N,r) = \{\Delta_U \in \mathbb{R}^{N\times r} : U^\top \Delta_U = 0\} = \operatorname{range}(U^\perp).$$

The horizontal space is endowed with the usual inner product from the embedding space $\mathbb{R}^{N\times r}$,

$$\langle \Delta_1, \Delta_2 \rangle_U = \mathrm{trace}(\Delta_1^\top \Delta_2), \quad \forall \Delta_1, \Delta_2 \in \mathrm{T}_U\mathrm{Grass}(N,r).$$

The norm of a tangent vector $\Delta_{\mathcal{U}} \in \mathrm{T}_{\mathcal{U}}\mathrm{Grass}(N,r)$ is given by the norm of its horizontal lift $\Delta_U \in \mathrm{T}_U\mathrm{Grass}(N,r)$. Thus, we make sense of the notation

$$\|\Delta_{\mathcal{U}}\|_{\mathrm{F}} := \|\Delta_U\|_{\mathrm{F}} = \sqrt{\langle \Delta_U, \Delta_U \rangle_U}, \quad \forall \Delta_{\mathcal{U}} \in \mathrm{T}_{\mathcal{U}}\mathrm{Grass}(N,r).$$

Similarly, for $\xi = (\Delta_X, \Delta_{\mathcal{U}}) \in \mathrm{T}_z\mathcal{M}$, we write $\|\xi\|_{\mathrm{F}} := \sqrt{\|\Delta_X\|_{\mathrm{F}}^2 + \|\Delta_U\|_{\mathrm{F}}^2}$. There exists several ways to compute a retraction on the Grassmann manifold. We choose the polar retraction defined in Equation (2.34), which we denote by $\mathrm{R}_U$. This allows to define a retraction on $\mathcal{M}$, as the retraction on $\mathrm{L}_{\mathcal{A},b}$ is simply given by $\mathrm{R}_X(\Delta_X) = X + \Delta_X$. The description of $\mathrm{L}_{\mathcal{A},b}$ and $\mathrm{Grass}(N,r)$ as Riemannian manifolds (Section 2.9) shows how to compute the Riemannian gradient and Hessian of a function defined on those sets. The Euclidean derivatives, which do not take the constraints into account, computed using the results of Section 4.1.2 or automatic differentiation, are appropriately projected onto the tangent spaces of $\mathcal{M}$.

**Riemannian trust-region (RTR)** We apply the Riemannian trust-region (Algorithm 2, page 33), described in Chapter 2, to Problem (4.1) and (4.4). Recall that, at each iterate $z_k \in \mathcal{M}$, the Riemannian trust-region minimizes the following model of the cost function on the tangent space $\mathrm{T}_{z_k}\mathcal{M}$:

$$\min_{\substack{\eta \in \mathrm{T}_{z_k}\mathcal{M} \\ \|\eta\|_{z_k} \leq \Delta_k}} \hat{m}_{z_k}(\eta) := f(z_k) + \langle \eta, \mathrm{grad} f(z_k) \rangle_{z_k} + \frac{1}{2}\langle \eta, H_k[\eta] \rangle_{z_k},$$

where $H_k \colon \mathrm{T}_{z_k}\mathcal{M} \to \mathrm{T}_{z_k}\mathcal{M}$ is a symmetric operator on $\mathrm{T}_{z_k}\mathcal{M}$, $\Delta_k$ is the trust-region radius and the model $\hat{m}_{z_k} \colon \mathrm{T}_{z_k}\mathcal{M} \to \mathbb{R}$ is a quadratic approximation of the *pullback* $\hat{f}_{z_k} = f \circ \mathrm{R}_{z_k}$, defined on the tangent space at $z_k \in \mathcal{M}$ for some retraction $\mathrm{R}_{z_k} \colon \mathrm{T}_{z_k}\mathcal{M} \to \mathcal{M}$. A first-order version of the algorithm is given by the choice $H_k = \mathrm{Id}$, whereas a second-order version if obtained with $H_k = \mathrm{Hess} f(z_k)$ or an approximation of the Hessian. If a first-order critical point is sought, set $\varepsilon_H = \infty$. We discuss the implementation of RTR in Section 4.6. We note that there is no guarantee on the quality of the stationary point, due to the nonconvexity. Nonetheless, we see in Section 4.6 that the method performs well in practice for nonlinear matrix recovery.

The Riemannian trust-region also provably converges to second-order critical points for any initialization under a weak decrease condition in the subproblems and satisfies global worst-case complexity bounds matching their unconstrained counterparts, as was shown in (Boumal et al., 2019). These are stated in Theorem 2.8 on page 35, which says that, under A1, A2, A3, A5, A6, A4, A7, first-order RTR finds a point $z_{N_1} \in \mathcal{M}$ such that $\|\mathrm{grad} f(z_{N_1})\|_{\mathrm{F}} \leq \varepsilon_g$ in at most $\mathcal{O}(\varepsilon_g^{-2})$ iterations and a point $z_{N_2} \in \mathcal{M}$ such that $\|\mathrm{grad} f(z_{N_2})\|_{\mathrm{F}} \leq \varepsilon_g$ and $\lambda_{\min}(H_{N_2}) \geq -\varepsilon_H$ in at most $\mathcal{O}\left(1/\varepsilon_g^2 \varepsilon_H\right)$ iterations. In Section 4.5, we detail how the Lipschitz conditions A2 and A3 relate to the smoothness of the kernel and its derivatives, and discuss the practicality of these assumptions for problem (4.4) when the monomial and Gaussian kernels are used.

## 4.3 Alternating minimization algorithms

In this section, departing from the techniques in 4.2, we propose an alternating minimization algorithm to solve (4.1) and (4.4) (Algorithm 4). This comes from the natural separation of the variables into two blocks $X$ and $\mathcal{U}$, yielding two distinct minimization subproblems. Alternating minimization type methods have been popular in recent years to solve large-scale nonconvex problems (Wen et al., 2012; Bolte et al., 2013). This is due to their good practical performances and ease of implementation, as often one or both of the subproblems have a closed-form solution. Strictly speaking, this is still a Riemannian optimization approach, as all iterates are feasible, but this section describes a two-block coordinate minimization, whereas the previous section was considering both variables as a single block.

We set the initial guess $X_0 \in \mathbb{R}^{n \times s}$ as any solution of the underdetermined linear system $\mathcal{A}(X) = b$ and $\mathcal{U}_0$ as the span of the $r$ leading singular vectors of $\Phi(X_0)$. The framework is as follows, for $k \geq 0$:
With $\mathcal{U}_k$ fixed, solve

$$X_{k+1} = \begin{cases} \underset{X \in \mathbb{R}^{n \times s}}{\arg\min} \quad \|\Phi(X) - \mathrm{P}_{\mathcal{U}_k} \Phi(X)\|_{\mathrm{F}}^2 \\ \qquad\qquad \mathcal{A}(X) = b. \end{cases} \tag{4.7}$$

With $X_{k+1}$ fixed, solve

$$\mathcal{U}_{k+1} = \begin{cases} \underset{\mathcal{U}}{\arg\min} \quad \|\Phi(X_{k+1}) - \mathrm{P}_{\mathcal{U}} \Phi(X_{k+1})\|_{\mathrm{F}}^2 \\ \qquad\qquad \mathcal{U} \in \mathrm{Grass}(N, r). \end{cases} \tag{4.8}$$

This separation of the variables takes advantage of the fact that problem (4.8), even though nonconvex, is solved to global optimality by computing the $r$ leading left singular vectors of the matrix $\Phi(X_{k+1})$. The result is a consequence of the celebrated Eckart-Young-Mirsky theorem, which gives the best rank $r$ approximation in Frobenius norm of a matrix by the $r$ leading terms of the singular value decomposition (Eckart and Young, 1936; Mirsky, 1960). In particular, let $\Phi(X_{k+1}) = \sum_{i=1}^{\min(N,s)} \sigma_i u_i v_i^\top$, then

$$\mathcal{U}_{k+1} = \mathrm{span}(u_1, \ldots, u_r) \tag{4.9}$$

is a global minimizer of (4.8). Note that the solution need not be unique, in the case where $\sigma_r = \sigma_{r+1}$. The truncated singular value decomposition (4.9) is denoted by `truncate_svd` in Algorithm 4. The singular vectors can be obtained from the singular value decomposition, which can be computed in $\mathcal{O}(\max(N, s)^3)$ operations. Its numerical accuracy depends on the distribution of the singular spectrum, as described in (Demmel et al., 1999).

Problem (4.7) is in general hard to solve to global optimality. The difficulty comes from the nonconvexity of the cost function, which is due to $\Phi$. One can choose from a variety of first- or second-order methods to find an approximate first-or second-order critical point. We present the numerical merits of both possibilities in Section 4.6.

**i) First-order version of alternating minimization** When only gradient information is available, a first-order method is used to minimize subproblem (4.7). For the sake of illustration, in Algorithm 4 we present a projected gradient descent with line search for (4.7). The gradient of the cost function with respect to $X$ is projected onto the null space of $\mathcal{A}$. This ensures that the iterates remain in the feasible set $\mathrm{L}_{\mathcal{A},b}$. The line search is a classical backtracking with an Armijo condition for sufficient decrease. Variants in the line search or even constant step sizes are possible.

**ii) Second-order version of alternating minimization** In subproblem (4.7), it is possible to use a second-order method to speed up the local convergence and reach a higher accuracy. We apply the RTR on the affine manifold $\mathrm{L}_{\mathcal{A},b}$, where the quadratic model uses $H_k = \mathrm{P}_{\mathrm{TL}_{\mathcal{A},b}}(\nabla^2_{XX} f(X_k, \mathcal{U}_k))$, the Hessian of the cost function in $X$ restricted to the tangent space.

Algorithm 4 details a first-order version of the alternating minimization method, where gradient descent with an Armijo line search, a standard inexact procedure in nonconvex optimization, is applied to subproblem (4.7). The Armijo line search is described in Algorithm 5.

**iii) Accuracy of the subproblems solution** Algorithm (4) alternatively solves subproblems (4.7) and (4.8). For the solution of (4.7), there is no incentive to solve to high accuracy early on in the run of the algorithm, as we could still be far from convergence and the variable $\mathcal{U}$ might still change a lot. At iteration $k$, we use the following stopping criterion:

$$\|\mathrm{grad}_X f(X_{k+1}, \mathcal{U}_k)\|_\mathrm{F} \leq \varepsilon_{x,k} \qquad \text{for some } \varepsilon_{x,k} > 0.$$

**Algorithm 4** Alternating minimization scheme for Problem (4.1) and (4.4)

```
1: Given: The sensing matrix $A \in \mathbb{R}^{m \times ns}$, measurements $b \in \mathbb{R}^m$, tolerances $\varepsilon_u \geq 0, \varepsilon_x \geq 0$, an estimation of $r = \text{rank}(\Phi(M))$.
2: Set $k = 0$
3: Find $X_0$ that satisfies $AX_0 = b$
4: $U_0 =$ truncate_svd($\Phi(X_0)$)                                  ▷ Equation (4.9)
5: while $\|\text{grad}_X f(X_k, \mathcal{U}_k)\|_F > \varepsilon_x$ or $\|\text{grad}_{\mathcal{U}} f(X_k, \mathcal{U}_k)\|_F > \varepsilon_u$ do
6:     Set $X_k^{(0)} = X_k, i = 0$
7:     Choose $\varepsilon_{x,k}$ using Equation (4.10) or (4.11)
8:     while $\left\|\text{grad}_X f(X_k^{(i)}, \mathcal{U}_k)\right\|_F > \varepsilon_{x,k}$ do
9:         $\text{grad}_X f(X_k^{(i)}, \mathcal{U}_k) = \text{P}_{\text{TL}_{\mathcal{A},b}}\Big(\nabla_X f(X_k^{(i)}, \mathcal{U}_k)\Big)$        ▷ Equation (2.22)
10:        $\alpha_k^{(i)} = \text{Armijo}\Big((X_k^{(i)}, \mathcal{U}_k), -\text{grad}_X f(X_k^{(i)}, \mathcal{U}_k)\Big)$        ▷ Algorithm 5
11:        $X_k^{(i+1)} = X_k^{(i)} - \alpha_k^{(i)} \text{grad}_X f(X_k^{(i)}, \mathcal{U}_k)$
12:        $i = i + 1$
13:    end while
14:    $X_{k+1} = X_k^{(i)}$
15:    if $\|\text{grad}_{\mathcal{U}} f(X_{k+1}, \mathcal{U}_k)\|_F \leq \varepsilon_u$ then
16:        $U_{k+1} = U_k$
17:    else
18:        $U_{k+1} =$ truncate_svd($\Phi(X_{k+1})$)
19:    end if
20:    $k = k + 1$
21: end while
22: return $(X_k, \mathcal{U}_k)$ such that $\|\text{grad} f(X_k, \mathcal{U}_k)\|_F \leq \varepsilon_u + \varepsilon_x$.
```

**Algorithm 5** Armijo($z_k$, $d_k$): Line search with Armijo condition

**INPUT**: *Function $f$ and gradient* $\text{grad}_X f$, *current iterate* $(X_k^{(i)}, \mathcal{U}_k)$ *and a descent direction $d_k$ such that* $\langle \text{grad}_X f(X_k^{(i)}, \mathcal{U}_k), d_k \rangle < 0$, *a sufficient decrease coefficient* $\beta \in ]0, 1[$, *initial step* $\alpha_0 > 0$ *and* $\tau \in ]0, 1[$.

**OUTPUT**: *Step size* $\alpha_k^{(i)}$.

```
1: Set $\alpha = \alpha_0$.
2: while $f(X_k^{(i)} + \alpha d_k, \mathcal{U}_k) > f(X_k^{(i)}, \mathcal{U}_k) + \beta\alpha\langle \text{grad}_X f(X_k^{(i)}, \mathcal{U}_k), d_k \rangle$ do
3:     $\alpha = \tau\alpha$.
4: end while
5: Set $\alpha_k^{(i)} = \alpha$.
```

We propose the two following strategies for the choice of $\varepsilon_{x,k}$,

$$\varepsilon_{x,k} = \varepsilon_x \text{ for all } k, \tag{4.10}$$

or

$$\varepsilon_{x,k} = \max\left(\varepsilon_x, \theta \left\|\mathrm{grad}_X f(X_k, \mathcal{U}_k)\right\|_{\mathrm{F}}\right) \text{ for some user-chosen } 0 < \theta < 1. \tag{4.11}$$

To solve (4.8), it is possible to use a randomized SVD procedure. The randomized SVD is a stochastic algorithm that approximately computes the singular value decomposition of a matrix that exhibits a low-rank pattern (Halko et al., 2011). The matrix must be low rank or have a fast decay in its singular values for the random SVD to be accurate. As the iterates $X_k$ converge towards the solution $M$, the matrix $\Phi(X_k)$, for which we have to compute an SVD, becomes low-rank and therefore, it is natural to use a randomized SVD in Algorithm 4. In the early iterations, for a random starting point of the algorithm, the feature matrix $\Phi(X_0)$ is not expected to be low-rank and the random SVD should not be used. When the matrix $\Phi(X_k)$ is approaching a low-rank matrix, we can apply power iterations to make the singular values decrease faster. This makes the randomized decomposition more costly, but improves the accuracy of the randomized SVD.

Our strategy is as follows, choose two parameters $0 < \tau_1 \ll \tau_2 < 1$. As long as $f(X_{k+1}, \mathcal{U}_k) > \tau_2$, use an exact SVD algorithm, without randomization. When $\tau_1 < f(X_{k+1}, \mathcal{U}_k) \leq \tau_2$, the energy of $\Phi(X_{k+1})$ is approximately contained in the span of $\mathcal{U}_k$ which has dimension $r$. We use a randomized SVD, which we start up with a step of the power method to improve the accuracy. When $f(X_{k+1}, \mathcal{U}_k) \leq \tau_1$, the matrix $\Phi(X_{k+1})$ is close enough to a low-rank matrix to use a randomized SVD without a power iteration.

## 4.4 Convergence of the alternating minimization algorithm

In this section we present convergence results for Algorithm 4. We consider a first-order version where subproblem (4.7) is minimized using gradient descent and the Armijo backtracking line search (Algorithm 5). We first show asymptotic convergence of the gradient norms to zero. We also give a worst-case global complexity bound on the number of iterations necessary to achieve a small gradient from an arbitrary initial starting point. Note that we chose the Armijo line search for the sake of example, and minor adjustments to the proofs below allow to prove similar results for other minimization methods in subproblem (4.7). The convergence analysis of Algorithm 4 relies on the following assumption.

**A8.** *There exist constants $L_x$ and $L_u$ (which are both independent of $X$ and $\mathcal{U}$) such that for all $z = (X,\mathcal{U}) \in \mathcal{M}$, the pullback $\hat{f}_z = f \circ \mathrm{R}_z$ has a Lipschitz continuous gradient in $X$ and $\mathcal{U}$, with constants $L_x$ and $L_u$, respectively. That is, for all $(\eta_x, \eta_u) \in \mathrm{T}_z\mathcal{M}$,*

$$|f \circ \mathrm{R}_z(\eta_x, 0) - [f(X,\mathcal{U}) + \langle \mathrm{grad}_X f(X,\mathcal{U}), \eta_x\rangle]| \leq \frac{L_x}{2} \|\eta_x\|_{\mathrm{F}}^2 \tag{4.12}$$

*and*

$$|f \circ \mathrm{R}_z(0, \eta_u) - [f(X,\mathcal{U}) + \langle \mathrm{grad}_{\mathcal{U}} f(X,\mathcal{U}), \eta_u\rangle]| \leq \frac{L_u}{2} \|\eta_u\|_{\mathrm{F}}^2 . \tag{4.13}$$

*In other words, this means that, in each variable, the pullback is well approximated in a uniform way by its first-order Taylor approximation.*

**Remark 4.1.** *Note that if A2 holds, then A8 holds with $L_x = L_u = L_g$.*

### 4.4.1 Global convergence results

We carry on with the convergence analysis of Algorithm 4. The next lemma adapts the classical descent lemma for the SVD step.

**Lemma 4.4.** (Descent lemma based on (Boumal et al., 2019, Theorem 4)) *Let $f \colon \mathcal{M} \to \mathbb{R}$ be such that A8 holds and $f(z) \geq f^*$ for all $z \in \mathcal{M}$. Then, for any $k \geq 0$,*

$$f(X_{k+1},\mathcal{U}_k) - f^* \geq \frac{1}{2L_u} \|\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k)\|_{\mathrm{F}}^2 .$$

*where $L_u$ is the Lipschitz constant of the gradient of the pullback (A8) and $f^*$ is the lower bound of $f$ (A1).*

*Proof.* We follow the development of (Boumal et al., 2019, Theorem 4). By Lipschitz continuity of the gradient we have,

$$|f\left(X_{k+1}, \mathrm{R}_{\mathcal{U}_k}(\eta)\right) - [f(X_{k+1},\mathcal{U}_k) + \langle \mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k), \eta\rangle]| \leq \frac{L_u}{2} \|\eta\|_{\mathrm{F}}^2 \quad \forall \eta \in \mathrm{T}_{\mathcal{U}}\mathrm{Grass}(N, r).$$

Let $\eta = -\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k)/L_u$ and define $\mathcal{U}^+ = \mathrm{R}_{\mathcal{U}_k}(-\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k)/L_u)$, which gives

$$\begin{aligned} f(X_{k+1},\mathcal{U}^+) &\leq f(X_{k+1},\mathcal{U}_k) + \langle \mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k), -\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k)/L_u\rangle \\ &\quad + \frac{L_u}{2} \|-\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k)/L_u\|_{\mathrm{F}}^2 \\ &\leq f(X_{k+1},\mathcal{U}_k) - \frac{1}{2L_u} \|\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k)\|_{\mathrm{F}}^2 . \end{aligned}$$

We conclude that

$$f(X_{k+1},\mathcal{U}_k) - f^* \geq f(X_{k+1},\mathcal{U}_k) - f(X_{k+1},\mathcal{U}^+) \geq \frac{1}{2L_u} \|\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_k)\|_{\mathrm{F}}^2 . \qquad \square$$

Given two consecutive iterates $X_k$ and $X_{k+1}$ of Algorithm 4, we denote the intermediate iterates by

$$X_k = X_k^{(0)},\ X_k^{(1)},\ X_k^{(2)},\ \dots,\ X_k^{(n_k)} = X_{k+1},$$

where $n_k \geq 0$ is the number of gradient steps between $X_k$ and $X_{k+1}$. The next lemma gives upper and lower bounds on the step sizes returned by the Armijo line search. This is a standard result for line search methods (Nocedal and Wright, 2006) where the constraint $\mathcal{A}(X) = b$ is added.

**Lemma 4.5.** *Under A8, for the direction $-\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k) \in \mathrm{TL}_{\mathcal{A},b}$, Algorithm 5 returns a step size $\alpha_k^{(i)}$ that satisfies*

$$\underline{\alpha} := \min\left\{\alpha_0, \frac{2\tau(1-\beta)}{L_x}\right\} \leq \alpha_k^{(i)} \leq \alpha_0$$

*and ensures the following decrease*

$$f(X_k^{(i)},\mathcal{U}_k) - f(X_k^{(i+1)},\mathcal{U}_k) \geq \beta\underline{\alpha} \left\|\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k)\right\|_{\mathrm{F}}^2, \tag{4.14}$$

*where $X_k^{(i+1)} = X_k^{(i)} - \alpha_k^{(i)}\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k)$.*

*Proof.* It is clear from the line search that $\alpha_k^{(i)} \leq \alpha_0$. For any $\alpha > 0$, Lipschitz continuity of the gradient in $X$ (A8) gives

$$\begin{aligned} f\left(X_k^{(i)} - \alpha\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k),\mathcal{U}_k\right) \leq f(X_k^{(i)},\mathcal{U}_k) - \alpha \left\|\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k)\right\|_{\mathrm{F}}^2 \\ + \alpha^2\frac{L_x}{2}\left\|\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k)\right\|_{\mathrm{F}}^2. \end{aligned}$$

Hence the Armijo condition (4.14) is satisfied whenever

$$-\alpha\left\|\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k)\right\|_{\mathrm{F}}^2 + \alpha^2\frac{L_x}{2}\left\|\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k)\right\|_{\mathrm{F}}^2 \leq -\alpha\beta\left\|\mathrm{grad}_X f(X_k^{(i)},\mathcal{U}_k)\right\|_{\mathrm{F}}^2,$$

which simplifies to

$$\alpha \leq \frac{2(1-\beta)}{L_x} =: \alpha_{max}.$$

If $\alpha_0$ satisfies Armijo, then $\alpha_k^{(i)} = \alpha_0$. Otherwise, we have $\alpha_k^{(i)} = \tau\alpha_l$ where $\alpha_l > \alpha_{max}$ is the last $\alpha$ that does not satisfy Armijo and $\alpha_{l+1} = \tau\alpha_l$ satisfies Armijo. In this case we have $\alpha_k^{(i)} \geq \tau\alpha_{max} = \dfrac{2\tau(1-\beta)}{L_x}$. □

We are now ready to prove global convergence of the alternating minimization algorithm.

**Theorem 4.6** (Global convergence for Alternating minimization)**.** *Let A8 hold for* $f\colon \mathcal{M} \to \mathbb{R}$ *from* (4.1) *or* (4.4)*. Let* $\varepsilon_x = 0$*,* $\varepsilon_u = 0$ *and use Equation* (4.11) *to set* $\varepsilon_{x,k}$*. For any starting point* $(X_0,\mathcal{U}_0) \in \mathcal{M}$*, Algorithm 4 produces a sequence* $\Big(X_k,\mathcal{U}_k\Big)_{k\in\mathbb{N}}$ *such that*

$$\lim_{k\to\infty} \|\mathrm{grad} f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} = 0. \tag{4.15}$$

*Proof.* First note that $f$ is bounded below by $f_* = 0$. For any $k \geq 0$,

$$\begin{aligned} \big\|\big(\mathrm{grad}_X f(X_k,\mathcal{U}_k), \mathrm{grad}_{\mathcal{U}} f(X_k,\mathcal{U}_k)\big)\big\|_{\mathrm{F}} &\leq \|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} + \|\mathrm{grad}_{\mathcal{U}} f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} \\ &= \|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} \end{aligned} \tag{4.16}$$

since $\varepsilon_u = 0$. Given that each step is non-increasing,

$$\begin{aligned} f(X_k,\mathcal{U}_k) - f(X_{k+1},\mathcal{U}_{k+1}) &\geq f(X_k,\mathcal{U}_k) - f(X_{k+1},\mathcal{U}_k) \\ &\geq f(X_k,\mathcal{U}_k) - f(X_k^{(1)},\mathcal{U}_k) \\ &\geq \beta\alpha_k^{(0)} \|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}}^2 \\ &\geq \beta\underline{\alpha} \|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}}^2, \end{aligned}$$

where we used the Armijo decrease from Lemma 4.5. Summing over all iterations gives a telescopic sum on the left-hand side. For any $\bar{k} \geq 0$,

$$f(X_0,\mathcal{U}_0) - f^* \geq f(X_0,\mathcal{U}_0) - f(X_{\bar{k}},\mathcal{U}_{\bar{k}}) \geq \beta\underline{\alpha} \sum_{k=0}^{\bar{k}} \|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}}^2.$$

The series is convergent since it is bounded independently of $\bar{k}$. Letting $\bar{k} \to \infty$ gives $\sum_{k=0}^{\infty} \|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}}^2 < \infty$ and therefore

$$\lim_{k\to\infty} \|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} = 0.$$

We have $\|\mathrm{grad}_{\mathcal{U}} f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} = 0$ for all $k \geq 0$ since $\varepsilon_u = 0$. This corresponds to taking exact singular value decompositions. Taking $k \to \infty$ in (4.16) gives (4.15). □

**Theorem 4.7** (Global complexity for Alternating minimization)**.** *Let A8 hold for* $f\colon \mathcal{M} \to \mathbb{R}$ *from* (4.1) *or* (4.4)*. Let* $\varepsilon_x > 0$*,* $\varepsilon_u > 0$ *and* $\varepsilon_{x,k}$ *be given by* (4.10) *or* (4.11)*. For any starting point* $z_0 = (X_0,\mathcal{U}_0) \in \mathcal{M}$*, Algorithm 4 produces a sequence* $\Big(X_k,\mathcal{U}_k\Big)_{k\in\mathbb{N}}$ *such that*

$$\Big\|\Big(\mathrm{grad}_X f(X_k,\mathcal{U}_k), \mathrm{grad}_{\mathcal{U}} f(X_k,\mathcal{U}_k)\Big)\Big\|_{\mathrm{F}} \leq \varepsilon_x + \varepsilon_u,$$

*is achieved using at most* $N_{grad}$ *gradient steps and* $N_{svd}$ *singular value decompositions with*

$$N_{grad} \leq \frac{f(z_0) - f_*}{\underline{\alpha}\beta\varepsilon_x^2} \qquad \text{and} \qquad N_{svd} \leq \frac{2L_u(f(z_0) - f_*)}{\varepsilon_u^2},$$

*where* $\underline{\alpha} := \min\{\alpha_0, 2\tau(1-\beta)/L_x\}$ *is a constant depending on parameters of the line search (Algorithm 5).*

*Proof.* Note that $f$ is bounded below by $f_* = 0$. Due to Theorem 4.6, there exists $N_{iter} < \infty$ that gives the number of iterations performed by Algorithm 4, i.e, the smallest $k$ such that $\|\mathrm{grad}_X f(X_k, \mathcal{U}_k)\|_{\mathrm{F}} \leq \varepsilon_x$ and $\|\mathrm{grad}_{\mathcal{U}} f(X_k, \mathcal{U}_k)\|_{\mathrm{F}} \leq \varepsilon_u$. Let $N_{svd}$ be the number of singular value decompositions that have to be performed to reach $\|\mathrm{grad}_{\mathcal{U}} f(X_{k+1}, \mathcal{U}_k)\|_{\mathrm{F}} \leq \varepsilon_u$, at which point the algorithm would return without performing another computation. For any $k \leq N_{svd}$, from Lemma 4.4 we have

$$f(X_{k+1}, \mathcal{U}_k) - f_* \geq \frac{1}{2L_u} \|\mathrm{grad}_{\mathcal{U}} f(X_{k+1}, \mathcal{U}_k)\|_{\mathrm{F}}^2 \geq \frac{1}{2L_u} \varepsilon_u^2.$$

Summing from $k = 0$ to $N_{svd}$ gives,

$$f(z_0) - f_* \geq f(z_0) - f(z_{N_{svd}}) \geq \sum_{k=0}^{N_{svd}} \frac{\varepsilon_u^2}{2L_u} = \frac{\varepsilon_u^2 N_{svd}}{2L_u}.$$

Hence, this bounds the number of SVD to ensure $\|\mathrm{grad}_{\mathcal{U}} f(X_{k+1}, \mathcal{U}_k)\|_{\mathrm{F}} \leq \varepsilon_u$, as

$$N_{svd} \leq 2L_u \frac{(f(z_0) - f_*)}{\varepsilon_u^2}.$$

For $0 \leq i \leq n_k - 1$, we have $\left\|\mathrm{grad}_X f(X_k^{(i)}, \mathcal{U}_k)\right\|_{\mathrm{F}}^2 \geq \varepsilon_{x,k}^2$ by definition since the stopping criterion is $\left\|\mathrm{grad}_X f(X_k^{(n_k)}, \mathcal{U}_k)\right\|_{\mathrm{F}} \leq \varepsilon_{x,k}$. Combined with the Armijo decrease this gives

$$f(X_k^{(i)}, \mathcal{U}_k) - f(X_k^{(i+1)}, \mathcal{U}_k) \geq \alpha_k^{(i)} \beta \left\|\mathrm{grad}_X f(X_k^{(i)}, \mathcal{U}_k)\right\|_{\mathrm{F}}^2 \geq \alpha_k^{(i)} \beta \varepsilon_{x,k}^2. \tag{4.17}$$

We sum these bounds for the $n_k$ gradient steps from $X_k$ to $X_{k+1}$,

$$\sum_{i=0}^{n_k-1} \left[ f(X_k^{(i)}, \mathcal{U}_k) - f(X_k^{(i+1)}, \mathcal{U}_k) \right] \geq \sum_{i=0}^{n_k-1} \alpha_k^{(i)} \beta \varepsilon_{x,k}^2.$$

Using that the step sizes $\alpha_k^{(i)}$ are bounded below by $\underline{\alpha} = \tau(1-\beta)/L_x$ (Lemma 4.5),

$$f(X_k, \mathcal{U}_k) - f(X_{k+1}, \mathcal{U}_k) \geq n_k \underline{\alpha} \beta \varepsilon_{x,k}^2 \quad \forall k.$$

The SVD is nonincreasing, meaning $f(X_k, \mathcal{U}_k) - f(X_{k+1}, \mathcal{U}_{k+1}) \geq f(X_k, \mathcal{U}_k) - f(X_{k+1}, \mathcal{U}_k)$. This yields,

$$f(X_k, \mathcal{U}_k) - f(X_{k+1}, \mathcal{U}_{k+1}) \geq n_k \underline{\alpha} \beta \varepsilon_{x,k}^2 \geq n_k \underline{\alpha} \beta \varepsilon_x^2 \qquad \forall k \leq N_{iter},$$

as both (4.10) and (4.11) satisfy $\varepsilon_{x,k} \geq \varepsilon_x$. We sum once again over the iterations,

$$f(X_0, \mathcal{U}_0) - f_* \geq f(X_0, \mathcal{U}_0) - f(X_{N_{iter}+1}, \mathcal{U}_{N_{iter}+1}) \geq \sum_{k=0}^{N_{iter}} n_k \underline{\alpha} \beta \varepsilon_x^2.$$

We conclude that

$$\frac{(f(z_0) - f_*)}{\underline{\alpha} \beta \varepsilon_x^2} \geq \sum_{k=0}^{N_{iter}} n_k =: N_{grad}.$$ □

A similar algorithm using fixed step sizes for the update in $X$ also converges, provided the step sizes are small enough.

**Corollary 4.8.** *If the Armijo line search in Algorithm 4 is replaced by a gradient descent with constant step sizes $\alpha$ satisfying $\alpha < \dfrac{2}{L_x}$, Algorithm 4 converges*

$$\lim_{k\to\infty} \left\| \Big(\mathrm{grad}_X f(X_k,\mathcal{U}_k), \mathrm{grad}_{\mathcal{U}} f(X_k,\mathcal{U}_k)\Big) \right\|_{\mathrm{F}} = 0. \tag{4.18}$$

*We also have the worst-case bound*

$$N_{grad} \leq \frac{L_x(f_0 - f_*)}{\alpha\varepsilon_x^2}. \tag{4.19}$$

*Proof.* We derive the usual descent lemma from Lipschitz continuity of the gradient. This gives

$$f\big(X_k - \alpha \mathrm{grad}_X f(X_k,\mathcal{U}_k),\mathcal{U}_k\big) \leq f(X_k,\mathcal{U}_k) - \alpha \left\|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\right\|_{\mathrm{F}}^2 + \alpha^2 L_x/2 \left\|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\right\|_{\mathrm{F}}^2$$

which simplifies to

$$f(X_k,\mathcal{U}_k) - f(X_k^{(1)},\mathcal{U}_k) \geq (\alpha - \alpha^2 L_x/2) \left\|\mathrm{grad}_X f(X_k,\mathcal{U}_k)\right\|_{\mathrm{F}}^2.$$

This bound replaces the Armijo decrease of Equation (4.14). The rest of the proofs from Theorems 4.6 and 4.7 holds verbatim with step size $\alpha$ for every iteration. Note that for $\alpha > 0$, the factor $(\alpha - \alpha^2 L_x/2)$ is positive for $\alpha < 2/L_x$ and is maximized at $\alpha = 1/L_x$. $\square$

### 4.4.2 Convergence of the iterates using the Kurdyka-Lojasiewicz property

This section proves convergence of the sequence of iterates to a unique stationary point for a simplified version of the alternating minimization scheme. This section considers an algorithm where only one gradient step is performed in between the truncated singular value decompositions (Algorithm 6). This is similar to the algorithm described in (Fan et al., 2019) which does not provide theoretical convergence guarantees. Our observations indicate that Algorithm 6 is expected to behave similarly to Algorithm 4 in the limit. Asymptotically, there is usually only one gradient step needed between two truncated singular value decompositions. It is only in the early iterations that Algorithm 4 differs by making several gradient steps in between singular value decompositions. For the purpose of this theoretical section, we assume that the singular value decompositions in Algorithm 6 are exact and not approximated or randomized. This corresponds to setting $\varepsilon_u = 0$ in Algorithm 4. This section is written using the notation of a feature matrix $\Phi$ as in problem (4.1), but the results apply similarly to problem (4.4) if one assumes that the Lipschitz condition A10 applies to a kernel K instead of $\Phi$.

**Algorithm 6** A simple alternating minimization scheme for Problem (4.1) or (4.4)

```
1: Given: The sensing matrix $A \in \mathbb{R}^{m \times ns}$, measurements $b \in \mathbb{R}^m$, a tolerance $\varepsilon_x > 0$,
   estimation of $r = \mathrm{rank}(\Phi(M))$.
2: Set $k = 0$
3: Find $X_0$ that satisfies $AX_0 = b$.
4: $U_0 =$ truncate_svd($\Phi(X_0)$)                              ▷ Equation (4.9)
5: while $\|\mathrm{grad}_X f(X_k, \mathcal{U}_k)\|_\mathrm{F} > \varepsilon_x$ do
6:     $\mathrm{grad}_X f(X_k, \mathcal{U}_k) = \mathrm{P}_{\mathrm{TL}_{\mathcal{A},b}}(\nabla_X f(X_k, \mathcal{U}_k))$        ▷ Equation (2.22)
7:     $\alpha_k = \mathrm{Armijo}((X_k, \mathcal{U}_k), -\mathrm{grad}_X f(X_k, \mathcal{U}_k))$        ▷ Algorithm 5
8:     $X_{k+1} = X_k - \alpha_k \mathrm{grad}_X f(X_k, \mathcal{U}_k)$
9:     $U_{k+1} =$ truncate_svd($\Phi(X_{k+1})$)             ▷ exact SVD, not randomized
10: end while
11: return $(X_k, \mathcal{U}_k)$ such that $\|\mathrm{grad} f(X_k, \mathcal{U}_k)\|_\mathrm{F} \leq \varepsilon_x$.
```

We define a distance on the manifold $\mathcal{M} = \mathrm{L}_{\mathcal{A},b} \times \mathrm{Grass}(N, r)$.

**Definition 4.1** (Chordal distance on $\mathcal{M}$)**.** *Given two subspaces $\mathcal{U}_1, \mathcal{U}_2 \in \mathrm{Grass}(N, r)$, the canonical angles $\theta_i$ for $i = 1, \dots, r$ are defined as $\theta_i = cos^{-1}(\sigma_i)$ where $\sigma_i$ are the $r$ singular values of $U_1^\top U_2$, with* range$(U_1) = \mathcal{U}_1$ *and* range$(U_2) = \mathcal{U}_2$*. For all $\mathcal{U}_1, \mathcal{U}_2 \in \mathrm{Grass}(N, r)$ the* chordal distance *on* Grass$(N, r)$ *is defined by* $\mathrm{dist}(\mathcal{U}_1, \mathcal{U}_2) := \sqrt{\sum_{i=1}^r \sin^2 \theta_i}$*. For all $(X_1, \mathcal{U}_1), (X_2, \mathcal{U}_2) \in \mathcal{M}$, define*

$$\mathrm{dist}\Big((X_1, \mathcal{U}_1), (X_2, \mathcal{U}_2)\Big) := \sqrt{\|X_1 - X_2\|_\mathrm{F}^2 + \sum_{i=1}^r \sin^2 \theta_i}. \tag{4.20}$$

*as a distance on $\mathcal{M}$.*

In this section, we prove finite length of the sequence of iterates in $\mathcal{M}$ using the metric defined above. For two subspaces $\mathcal{U}_1$ and $\mathcal{U}_2$, with $\Theta = \mathrm{diag}(\theta_i)$ the diagonal matrix containing the principal angles, the geodesic distance between $\mathcal{U}_1$ and $\mathcal{U}_2$ is given by $\|\Theta\|_\mathrm{F}$. We prefer to use the chordal distance $\|\sin\Theta\|_\mathrm{F}$, because it is easier to derive perturbation bounds for the singular value decomposition in this metric. The two distances are equivalent, the geodesic distance takes values between 0 and $\pi\sqrt{r}/2$, while the chordal distance takes values between 0 and $\sqrt{r}$ (Dhillon et al., 2008).

The following assumption ensures a non-degeneracy of the spectrum of the feature matrix.

**A9** (Gap between the singular values)**.** *There exists $\delta > 0$ such that the sequence $(X_k)_{k \in \mathbb{N}}$ generated by Algorithm 6 satisfies, for all $k \geq 0$,*

$$\sigma_r(\Phi(X_k)) - \sigma_{r+1}(\Phi(X_k)) \geq \delta > 0.$$

This property ensures that the minimizer of the function $f(X,\cdot)\colon \mathrm{Grass}(N,r) \to \mathbb{R}$ is well defined, i.e., that the truncated SVD of $\Phi(X)$ is unique, which is necessary to establish convergence of the alternating minimization to a unique limit point. We use Assumption 9 to derive a Lipschitz continuity result on the truncated singular value decomposition. We now show two main lemmas (4.9 and 4.13), inspired by (Bolte et al., 2013).

**Lemma 4.9** (Gradient lower bound on iterates gap)**.** *Assume that Algorithm 6 generates a bounded sequence of iterates. Then, there exists $\rho_2 > 0$ such that, for all $k \in \mathbb{N}$,*

$$\|\mathrm{grad} f(X_{k+1},\mathcal{U}_{k+1})\|_{\mathrm{F}} \leq \rho_2 \mathrm{dist}\Big((X_{k+1},\mathcal{U}_{k+1}),(X_k,\mathcal{U}_k)\Big) \tag{4.21}$$

*with $\rho_2 := 2(L_g + 1/\underline{\alpha})$ for some $L_g \geq 0$.*

*Proof.* The expression

$$X_{k+1} = X_k - \alpha_k \mathrm{grad}_X f(X_k,\mathcal{U}_k)$$

implies

$$\mathrm{grad}_X f(X_{k+1},\mathcal{U}_{k+1}) = (X_k - X_{k+1})/\alpha_k + \mathrm{grad}_X f(X_{k+1},\mathcal{U}_{k+1}) - \mathrm{grad}_X f(X_k,\mathcal{U}_k).$$

Define the set $\tilde{S} = \mathrm{cl}\,(\mathrm{conv}((X_k)_{k\in\mathbb{N}}))$, the closure of the convex hull of the sequence of iterates, and $S = \tilde{S} \times \mathrm{Grass}(N,r)$. We show that the vector field $\mathrm{grad} f|_S\colon S \to \mathrm{T}\mathcal{M}$ is $L_g$-Lipschitz continuous in the sense of Definition 4.3 for some $L_g \geq 0$. Since $S$ is bounded and the Hessian is continuous, there exists $L_g \geq 0$ such that $\|\mathrm{Hess} f(x)\| \leq L_g$ for all $x \in S$. By Proposition 4.18, $\mathrm{grad} f|_S$ is $L_g$-Lipschitz continuous. Using the triangular inequality and the fact that $\underline{\alpha}$ is a lower bound of $\alpha_k$ for all $k$ gives

$$\begin{aligned}\|\mathrm{grad}_X f(X_{k+1},\mathcal{U}_{k+1})\|_{\mathrm{F}} &\leq \|X_{k+1} - X_k\|_{\mathrm{F}}/\underline{\alpha} + \|\mathrm{grad}_X f(X_{k+1},\mathcal{U}_{k+1}) - \mathrm{grad}_X f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} \\ &\leq \mathrm{dist}\Big((X_{k+1},\mathcal{U}_{k+1}),(X_k,\mathcal{U}_k)\Big)/\underline{\alpha} + L_g \mathrm{dist}\Big((X_{k+1},\mathcal{U}_{k+1}),(X_k,\mathcal{U}_k)\Big) \\ &\leq (1/\underline{\alpha} + L_g)\mathrm{dist}\Big((X_{k+1},\mathcal{U}_{k+1}),(X_k,\mathcal{U}_k)\Big).\end{aligned}$$

This gives (4.21) recalling that, since $\mathrm{grad}_{\mathcal{U}} f(X_{k+1},\mathcal{U}_{k+1}) = 0$,

$$\|\mathrm{grad} f(X_{k+1},\mathcal{U}_{k+1})\|_{\mathrm{F}} = \|\mathrm{grad}_X f(X_{k+1},\mathcal{U}_{k+1})\|_{\mathrm{F}}.$$ □

Further auxiliary results are needed.

**Lemma 4.10** (Wedin's theorem (Stewart, 1998))**.** *Let $Y, \check{Y} \in \mathbb{R}^{N\times s}$ with singular value decompositions*

$$Y = \sum_{i=1}^{\min(N,s)} \sigma_i u_i (v_i)^\top \qquad \textit{and} \qquad \check{Y} = \sum_{i=1}^{\min(N,s)} \check{\sigma}_i \check{u}_i (\check{v}_i)^\top,$$

*with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(N,s)}$ and similarly for $\check{Y}$. If there exists $\delta > 0$ such that*

$$\min_{\substack{1 \leq i \leq r \\ r+1 \leq j \leq \min(N,s)}} |\check{\sigma}_i - \sigma_j| \geq \delta \tag{4.22}$$

*and*

$$\check{\sigma}_r \geq \delta,$$

*then*

$$\|\sin \Theta\|_{\mathrm{F}}^2 \leq \frac{2 \left\|\check{Y} - Y\right\|_{\mathrm{F}}^2}{\delta^2} \tag{4.23}$$

*with $\Theta$ the matrix of the principal angles between $\begin{bmatrix} u_1 & u_2 & \cdots & u_r \end{bmatrix}$ and $\begin{bmatrix} \check{u}_1 & \check{u}_2 & \cdots & \check{u}_r \end{bmatrix}$.*

The following lemma is a direct consequence of Wedin's theorem.

**Lemma 4.11.** *Let $Y, \check{Y} \in \mathbb{R}^{N \times s}$. Consider the singular value decomposition of $Y = \sum_{i=1}^{\min(N,s)} \sigma_i u_i v_i^\top$, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(N,s)}$. Let us also write $U_r := \begin{bmatrix} u_1 & u_2 & \cdots & u_r \end{bmatrix}$, a matrix whose columns span the left principal subspace associated to the $r$ largest singular values. Similarly, $\sum_{i=1}^{\min(N,s)} \check{\sigma}_i \check{u}_i \check{v}_i^\top$, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(N,s)}$. Let us also write $\check{U}_r := \begin{bmatrix} \check{u}_1 & \check{u}_2 & \cdots & \check{u}_r \end{bmatrix}$. If there exists $\delta > 0$ such that $\sigma_r - \sigma_{r+1} \geq \delta$ and $\check{\sigma}_r - \check{\sigma}_{r+1} \geq \delta$, then*

$$\operatorname{dist}(\check{\mathcal{U}}_r, \mathcal{U}_r)^2 \leq \frac{2}{\delta^2} \left\|\check{Y} - Y\right\|_{\mathrm{F}}^2,$$

*where $\operatorname{dist}(\mathcal{U}_r, \check{\mathcal{U}}_r) = \sqrt{\sum_{i=1}^r \sin(\theta_i)^2}$ (with $\theta_i$ the principal angles between $\mathcal{U}_r$ and $\check{\mathcal{U}}_r$) is the distance between the subspaces $\mathcal{U}_r$ and $\check{\mathcal{U}}_r$.*

*Proof.* The result follows from the $\sin \Theta$ bound (4.23) in Wedin's theorem. Let us verify the assumptions. From the assumptions we know that $\sigma_r \geq \delta$ and $\check{\sigma}_r \geq \delta$. If Wedin's theorem does not apply, Assumption (4.22) is not satisfied and neither is it satisfied with the roles of $Y$ and $\check{Y}$ reversed. In that case, since there exists no $\delta > 0$ such that (4.22) holds, one must have $\sigma_i = \check{\sigma}_j$, for some $i \leq r$, $j \geq r+1$, and $\check{\sigma}_l = \sigma_m$, for $l \leq r, m \geq r+1$. However, since the singular values are ordered decreasingly, this gives:

$$\sigma_m \leq \sigma_i = \check{\sigma}_j \leq \check{\sigma}_l = \sigma_m,$$

which implies that there exists $i \leq r$ and $m \geq r+1$ such that

$$\sigma_m = \sigma_i = \check{\sigma}_j = \check{\sigma}_l.$$

This is a contradiction with $\sigma_r - \sigma_{r+1} \geq \delta$ and $\check{\sigma}_r - \check{\sigma}_{r+1} \geq \delta$. Therefore, these conditions guarantee that Wedin's theorem applies. $\square$

In the next lemma, we combine the previous bound with the Lipschitz continuity of $\Phi$.

**A10** (Lipschitz continuity of the features). *For Problem* (4.1)*, there exists $L_\Phi \geq 0$ such that for any $X_k, X_{k+1}$ produced by Algorithm 6, $\|\Phi(X_{k+1}) - \Phi(X_k)\|_{\mathrm{F}} \leq L_\Phi \|X_{k+1} - X_k\|_{\mathrm{F}}$. For Problem* (4.4)*, there exists $L_{\mathrm{K}} \geq 0$ such that $\|\mathrm{K}(X_{k+1}, X_{k+1}) - \mathrm{K}(X_k, X_k)\|_{\mathrm{F}} \leq L_{\mathrm{K}} \|X_{k+1} - X_k\|_{\mathrm{F}}$.*

If we assume that the sequence $(X_k)_{k\in\mathbb{N}}$ is bounded, which we do in the main result of this section (Theorem 4.16), then it is sufficient for the features and kernel to be locally Lipschitz continuous in order for A10 to hold. The monomial and Gaussian kernel are locally Lipschitz continuous. We also note that if the sublevel set $\{(X,\mathcal{U}) \in \mathcal{M} : f(X,\mathcal{U}) \leq f(X_0,\mathcal{U}_0)\}$ is bounded, then the iterates are contained in a bounded set since Algorithm 6 is a descent method.

**Lemma 4.12.** *Let A9 and A10 hold. It follows that, for all $k \geq 0$, the sequence $(X_k,\mathcal{U}_k)_{k\in\mathbb{N}}$ produced by Algorithm 6 satisfies*

$$\mathrm{dist}(\mathcal{U}_k,\mathcal{U}_{k+1})^2 \leq \frac{2L_\Phi^2}{\delta^2} \|X_{k+1} - X_k\|_{\mathrm{F}}^2 .$$

*Proof.* By definition of $\mathcal{U}_k$, Lemma 4.11 ensures that

$$\mathrm{dist}(\mathcal{U}_k,\mathcal{U}_{k+1})^2 \leq \frac{2}{\delta^2} \|\Phi(X_{k+1}) - \Phi(X_k)\|_{\mathrm{F}}^2 . \tag{4.24}$$

Indeed, $\mathcal{U}_k = \texttt{truncate-svd}(\Phi(X_k))$ is composed of the $r$ leading left singular vectors of $\Phi(X_k)$, which are uniquely defined due to A9. The result then follows from the Lipschitz continuity of $\Phi$. □

This lemma allows us to show the following result.

**Lemma 4.13** (Sufficient decrease property). *Assume that A9 and A10 hold. Then, there exists $\rho_1 > 0$, independent of k, such that the iterates of Algorithm 6 satisfy for all $k \geq 0$*

$$f(X_k,\mathcal{U}_k) - f(X_{k+1},\mathcal{U}_{k+1}) \geq \rho_1 \mathrm{dist}\Big((X_k,\mathcal{U}_k),(X_{k+1},\mathcal{U}_{k+1})\Big)^2 . \tag{4.25}$$

*Proof.* From the Armijo decrease of Lemma 4.5,

$$f(X_k,\mathcal{U}_k) - f(X_{k+1},\mathcal{U}_k) \geq \frac{\beta}{\alpha_0} \|X_k - X_{k+1}\|_{\mathrm{F}}^2 ,$$

where $\alpha_0$ is the largest step allowed by the backtracking. Set $M^2 = 2L_\Phi^2/\delta^2$. Using that $f(X_{k+1},\mathcal{U}_{k+1}) \leq f(X_{k+1},\mathcal{U}_k)$, we get

$$\begin{aligned}
f(\mathcal{U}_k, X_k) - f(\mathcal{U}_{k+1}, X_{k+1}) &\geq f(\mathcal{U}_k, X_k) - f(\mathcal{U}_k, X_{k+1}) \\
&\geq \frac{\beta}{\alpha_0} \|X_{k+1} - X_k\|_{\mathrm{F}}^2 = \frac{\beta}{\alpha_0(1+M^2)}(1+M^2) \|X_{k+1} - X_k\|_{\mathrm{F}}^2 \\
&\geq \frac{\beta}{\alpha_0(1+M^2)} \left(\|X_{k+1} - X_k\|_{\mathrm{F}}^2 + \mathrm{dist}^2(\mathcal{U}_k, \mathcal{U}_{k+1})\right),
\end{aligned}$$

where we use Lemma 4.12, thus establishing (4.25) with $\rho_1 := \dfrac{\beta}{\alpha_0(1+M^2)}$. □

We now show convergence of the gradient norms to zero for Algorithm 6.

**Corollary 4.14** (Global convergence for Algorithm 6)**.** *Set $\varepsilon_x = 0$, for any starting point $z_0 = (X_0,\mathcal{U}_0) \in \mathcal{M}$, Algorithm 6 applied to* (4.1) *or* (4.4) *produces a sequence $\Big(X_k,\mathcal{U}_k\Big)_{k\in\mathbb{N}}$ such that*

$$\lim_{k\to\infty} \|\mathrm{grad} f(X_k,\mathcal{U}_k)\|_{\mathrm{F}} = 0. \tag{4.26}$$

*Proof.* Using Lemmas 4.9 and 4.13 gives,

$$\begin{aligned}
f(X_k,\mathcal{U}_k) - f(X_{k+1},\mathcal{U}_{k+1}) &\geq \rho_1 \mathrm{dist}\Big((X_k,\mathcal{U}_k),(X_{k+1},\mathcal{U}_{k+1})\Big)^2 \\
&\geq \rho_1/\rho_2 \|\mathrm{grad} f(X_{k+1},\mathcal{U}_{k+1})\|_{\mathrm{F}}^2 .
\end{aligned}$$

We conclude by noticing that the left-hand side is a telescopic sum and that $f$ is nonnegative. For any $\bar{k} \geq 0$,

$$f(z_0) \geq \sum_{k=1}^{\bar{k}} \rho_1/\rho_2 \|\mathrm{grad} f(X_{k+1},\mathcal{U}_{k+1})\|_{\mathrm{F}}^2 .$$

The sum is bounded independently of $\bar{k} \geq 0$, hence $\lim\limits_{k\to\infty} \|\mathrm{grad} f(X_{k+1},\mathcal{U}_{k+1})\|_{\mathrm{F}} = 0$. □

The bounds in Lemmas 4.9 and 4.13 are standard and hold for most descent methods. The values $\rho_1, \rho_2$ depend on the specifics of the algorithm used (Bolte et al., 2013). We now define the Kurdyka-Lojasiewicz inequality on Riemannian manifolds, which was introduced in (Hosseini, 2015).

**Definition 4.2** (The Kurdyka-Lojasiewicz inequality)**.** *Let $\mathcal{M}$ be a Riemannian manifold. A locally Lipschitz function $f\colon \mathcal{M} \to \mathbb{R}$ satisfies the Kurdyka-Lajasiewicz inequality at $x \in \mathcal{M}$ if and only if there exist $\eta \in ]0,\infty[$, a neighbourhood $V \subset \mathcal{M}$ of $x$, and a continuous concave function $\kappa : [0,\eta] \to [0,\infty[$ such that*

- $\kappa(0) = 0,$

- *$\kappa$ is continuously differentiable on $]0, \eta[$,*
- *$\kappa' > 0$ on $]0, \eta[$,*
- *For every $y \in V$ with $f(x) < f(y) < f(x) + \eta$, we have*

$$\kappa'(f(y) - f(x)) \, \|\mathrm{grad} f(y)\| \geq 1. \tag{4.27}$$

*If $f$ satisfies the KL inequality at every point $x \in \mathcal{M}$, we call $f$ a KL function.*

Property (4.27) can be shown to imply

$$\left\|\mathrm{grad}_y \left(f \circ (\kappa(y) - \kappa(x)\right)\right\| \geq 1.$$

This property is easily shown to hold at points where $\mathrm{grad} f(x) \neq 0$. The intuition behind KL functions is that they can be re-parametrized to be sharp, even at critical points. They are *amenable to sharpness* (Bolte et al., 2013).

**Lemma 4.15** ((de Carvalho Bento et al., 2016) Lemma 4.1)**.** *Let $\{a_k\}_{k \in \mathbb{N}}$ be a sequence of nonnegative numbers. If $\sum\limits_{k=1}^{\infty} \dfrac{a_k^2}{a_{k-1}}$ converges, then $\sum\limits_{k=1}^{\infty} a_k$ converges as well.*

**Theorem 4.16.** *Set $\varepsilon_x = 0$. Assume that Algorithm 6, applied to problem* (4.1) *or* (4.4) *using the monomial or Gaussian kernel, generates a bounded sequence $(X_k, \mathcal{U}_k)_{k \in \mathbb{N}}$. If A9 and A10 hold, then, the sequence has finite length, that is*

$$\sum_{k=1}^{\infty} \mathrm{dist}\Big((X_k, \mathcal{U}_k), (X_{k+1}, \mathcal{U}_{k+1})\Big) < \infty. \tag{4.28}$$

*Additionally, $(X_k, \mathcal{U}_k)_{k \in \mathbb{N}}$ converges to a unique point $(X_*, \mathcal{U}_*)$, which is a critical point of $f$ on $\mathcal{M}$.*

*Proof.* The monomial and Gaussian kernels are algebraic or exponential function. These functions are known to be KL functions (Bolte et al., 2013), hence the cost function $f$ is a KL function (Definition 4.2). For convenience, we write $z_k = (X_k, \mathcal{U}_k)$. Let $\omega(z_0)$ denote the set of accumulation points for some starting point $z_0$. This is a closed set, which is bounded by assumption, therefore it is compact. Since the sequence $(z_k)_{k \in \mathbb{N}}$ is bounded, there exists a subsequence $(z_{k_q})_{q \in \mathbb{N}}$ which converges to some $\bar{z} \in \mathcal{M}$. We want to show that $\omega(z_0)$ is a singleton, i.e., $\omega(z_0) = \{\bar{z}\}$.

The function $f$ is continuous, which implies $\lim_{q \to \infty} f(X_{k_q}, \mathcal{U}_{k_q}) = f(\bar{z})$. Since $f(z_{k_q})_{q \in \mathbb{N}}$ is non-increasing, the function $f$ is also constant on $\omega(z_0)$. Since $f$ is a KL function, for every point $z \in \omega(z_0)$, there exists a neighbourhood $V_z$ of $z$ and a continuous concave function $\kappa_z\colon [0,\eta_z] \to [0,\infty[$ of class $C^1$ on $]0,\eta_z[$ with $\kappa_z(0)=0$ and $\kappa_z'>0$ on $]0,\eta_z[$ such that, for all $y \in V_z$ with $f(z) < f(y) < f(z)+\eta_z$, we have

$$\kappa_z'(f(y)-f(z))\,\|\mathrm{grad} f(y)\|_{\mathrm{F}} \geq 1.$$

By compactness of $\omega(z_0)$, we find a finite number of points $\bar{z}_1,\dots,\bar{z}_p$ in $\omega(z_0)$ such that $\cup_{i=1}^p V_{\bar{z}_i}$ covers $\omega(z_0)$. We choose $\varepsilon>0$, such that $V := \{y \in \mathcal{M} : \mathrm{dist}\big(y,\omega(z_0)\big) < \varepsilon\}$ is contained in $\cup_{i=1}^p V_{\bar{z}_i}$. Then, we set

$$\eta = \min_{i=1,\dots,p} \eta_{\bar{z}_i}, \qquad \kappa'(t) = \max_{i=1,\dots,p} \kappa_{\bar{z}_i}'(t) \qquad \text{and} \qquad \kappa(t) = \int_0^t \kappa'(\tau)d\tau.$$

We claim that for every $y \in V$ with $f(\bar{z}) < f(y) < f(\bar{z})+\eta$, we have

$$\kappa'(f(y)-f(\bar{z}))\,\|\mathrm{grad} f(y)\|_{\mathrm{F}} \geq 1.$$

To prove that claim we find some $\bar{z}_i$ such that $y \in V_{\bar{z}_i}$. From the definition of $\eta$ and the fact that $f$ is constant on $\omega(z_0)$, we have $f(\bar{z}_i) < f(y) < f(\bar{z}_i)+\eta_{\bar{z}_i}$. We reach our claim using the definition of $\kappa'$, as

$$\kappa'(f(y)-f(\bar{z}))\,\|\mathrm{grad} f(y)\|_{\mathrm{F}} \geq \kappa_{\bar{z}_i}'(f(y)-f(\bar{z}_i))\,\|\mathrm{grad} f(y)\|_{\mathrm{F}} \geq 1.$$

For $\eta>0$ given above, there exists $k_0$ such that for all $k>k_0$,

$$f(z_k) < f(\bar{z})+\eta.$$

By definition of accumulation point, there exists $k_1$ such that for all $k>k_1$,

$$\mathrm{dist}(z_k,\omega(z_0)) < \varepsilon.$$

Hence, for all $k > l = \max\{k_0,k_1\}$, we have

$$\kappa'(f(z_k)-f(\bar{z}))\,\|\mathrm{grad} f(z_k)\|_{\mathrm{F}} \geq 1.$$

Using $\|\mathrm{grad} f(z_{k+1})\|_{\mathrm{F}} \leq \rho_2 \mathrm{dist}\Big(z_k, z_{k+1}\Big)$ (Equation (4.21)), gives

$$\kappa'(f(z_k)-f(\bar{z})) \geq \frac{1}{\rho_2 \mathrm{dist}\Big(z_{k-1}, z_k\Big)}. \tag{4.29}$$

Concavity of $\kappa$ gives

$$\kappa\Big(f(z_k)-f(\bar{z})\Big) - \kappa\Big(f(z_{k+1})-f(\bar{z})\Big) \geq \kappa'\Big(f(z_k)-f(\bar{z})\Big)\Big(f(z_k)-f(z_{k+1})\Big). \tag{4.30}$$

By Equation (4.25), we have $\rho_1 \text{dist}^2\Big(z_k, z_{k+1}\Big) \leq f(z_k) - f(z_{k+1})$ for all $k \geq 0$. This fact, in addition to (4.29), plugged into (4.30) gives

$$\kappa\Big(f(z_k) - f(\bar{z})\Big) - \kappa\Big(f(z_{k+1}) - f(\bar{z})\Big) \geq \frac{1}{\rho_2 \text{dist}\Big(z_{k-1}, z_k\Big)} \rho_1 \text{dist}^2\Big(z_k, z_{k+1}\Big),$$

or equivalently,

$$\frac{\text{dist}^2\Big(z_k, z_{k+1}\Big)}{\text{dist}\Big(z_{k-1}, z_k\Big)} \leq \frac{\rho_2}{\rho_1} \kappa\Big(f(z_k) - f(\bar{z})\Big) - \kappa\Big(f(z_{k+1}) - f(\bar{z})\Big). \tag{4.31}$$

For any $N > l$, we sum (4.31) for all $l \leq k \leq N$, using that the right-hand side is a telescopic sum,

$$\begin{aligned}
\sum_{k \geq l}^{N} \frac{\text{dist}^2\Big(z_k, z_{k+1}\Big)}{\text{dist}\Big(z_{k-1}, z_k\Big)} &\leq \sum_{k \geq l}^{N} \frac{\rho_2}{\rho_1} \Big[\kappa\Big(f(z_k) - f(\bar{z})\Big) - \kappa\Big(f(z_{k+1}) - f(\bar{z})\Big)\Big] \\
&\leq \frac{\rho_2}{\rho_1} \Big[\kappa\Big(f(z_l) - f(\bar{z})\Big) - \kappa\Big(f(z_N) - f(\bar{z})\Big)\Big] \\
&\leq \frac{\rho_2}{\rho_1} \Big[\kappa\Big(f(z_l) - f(\bar{z})\Big) - \kappa\Big(f(\bar{z}) - f(\bar{z})\Big)\Big] \\
&= \frac{\rho_2}{\rho_1} \kappa\Big(f(z_l) - f(\bar{z})\Big),
\end{aligned} \tag{4.32}$$

where we used that $f(\bar{z}) \leq f(z_N)$ , $\kappa$ is increasing and $\kappa(0) = 0$. We deduce that the left-hand side of (4.32) converges as $N \to \infty$, since it is upper-bounded independently of $N$. By Lemma 4.15, $\sum_{k \geq l}^{\infty} \text{dist}\Big(z_k, z_{k+1}\Big)$ also converges and therefore

$$\sum_{k=1}^{\infty} \text{dist}\Big(z_k, z_{k+1}\Big) < \infty.$$

Thus, $(z_k)_{z \in \mathbb{N}}$ is a Cauchy sequence converging to $\bar{z} = \omega(z_0)$. This unique limit point is a critical point of $f$ on $\mathcal{M}$ according to Corollary 4.14. $\square$

## 4.5 Discussion of assumptions in convergence results

In this section, we discuss the Lipschitz smoothness conditions that are used in the complexity and convergence analysis of Algorithm 2 (Riemannian trust-region) and Algorithm 4 (Alternating minimization). We also investigate their practicality for the monomial and Gaussian kernel. More precisely, we establish under which conditions on the kernel one can ensure that A2, A3 and A8 are satisfied for the cost function of (4.4).

The following discussion requires the use of the exponential map (Definition 2.19) as the retraction. The exponential map follows geodesics along the manifold in directions prescribed by tangent vectors. Using the exponential map on $\mathcal{M}$ is not a restriction, as the exponential map on the Grassmann manifold is computable (Absil et al., 2004) and the exponential map on $\mathrm{L}_{\mathcal{A},b}$ is trivially given by (2.23).

We introduce some preliminary results in order to establish Lipschitz continuity of the vector field $\mathrm{grad} f$. Lipschitz continuity of a vector field on a smooth manifold is defined using the notion of parallel transport (Equation (2.13)). The injectivity radius at $x \in \mathcal{M}$, written $\mathrm{inj}(x)$, is introduced in Definition 2.20.

**Definition 4.3.** (Boumal, 2020, Definition 10.42) *A vector field $V$ on a connected manifold $\mathcal{M}$ is $L$-Lipschitz continuous if, for all $x, y \in \mathcal{M}$ with* $\mathrm{dist}(x,y) < \mathrm{inj}(x)$*,*

$$\|\mathrm{PT}^{\gamma}_{0\leftarrow 1} V(y) - V(x)\| \leq L \mathrm{dist}(x,y),$$

*where $\gamma : [0,1] \to \mathcal{M}$ is the unique minimizing geodesic connecting $x$ to $y$ and* $\mathrm{PT}^{\gamma}_{0\leftarrow 1}$ *denotes the parallel transport along $\gamma$.*

Functions with Lipschitz continuous gradient exhibit the following regularity condition for the pullback $\hat{f} = f \circ \mathrm{R}$, provided the retraction used is the exponential map, $\mathrm{R} = \mathrm{Exp}$.

**Proposition 4.17.** (Boumal, 2020, Corollary 10.52) *If $f : \mathcal{M} \to \mathbb{R}$ has $L$-Lipschitz continuous gradient, then*

$$|f(\mathrm{Exp}_x(\eta)) - f(x) - \langle \eta, \mathrm{grad} f(x) \rangle| \leq \frac{L}{2} \|\eta\|^2$$

*for all $(x, \eta)$ in the domain of the exponential map.*

In order to show Lipschitz continuity of the gradient, we use the following proposition, which is based on an upper bound on the operator norm of the Riemannian Hessian.

**Proposition 4.18.** (Boumal, 2020, Corollary 10.45) *If $f \colon \mathcal{M} \to \mathbb{R}$ is twice continuously differentiable on a manifold $\mathcal{M}$, then* $\mathrm{grad} f$ *is $L$-Lipschitz continuous if and only if* $\mathrm{Hess} f(x)$ *has operator norm bounded by $L$ for all $x \in \mathcal{M}$, that is,*

$$\|\mathrm{Hess} f(x)\| = \max_{\substack{\eta \in \mathrm{T}_x\mathcal{M} \\ \|\eta\|=1}} \|\mathrm{Hess} f(x)[\eta]\| \leq L. \tag{4.33}$$

We first compute the Euclidean gradient of (4.4) with respect to each variable,

$$\nabla_X f(X, \mathcal{W}) = \mathrm{DK}(X)^* \mathrm{P}_{\mathcal{W}^\perp}$$

and

$$\nabla_{\mathcal{W}} f(X, \mathcal{W}) = -2\mathrm{K}(X)\mathcal{W}.$$

This naturally gives, for $\Delta \in \mathrm{T}_{\mathcal{W}}\mathrm{Grass}(s,r)$

$$\nabla^2_{\mathcal{W}\mathcal{W}} f(X, \mathcal{W})[\Delta] = -2\mathrm{K}(X)\Delta.$$

**Proposition 4.19.** *Consider the cost function of* (4.4) *and assume that the retraction is the exponential map. If* $\mathrm{D}K(X)$ *is Lipschitz continuous over* $\mathrm{L}_{\mathcal{A},b}$*, then* (4.12) *holds where* $L_x$ *is the Lipschitz constant of* $\mathrm{D}K(X)$*. If* $\|K(X)\|_{\mathrm{F}} \leq C$ *for all* $X \in \mathrm{L}_{\mathcal{A},b}$*, condition* (4.13) *holds with* $L_u = 2C$*.*

*Proof.* For a given $X \in \mathrm{L}_{\mathcal{A},b}$, consider the function $f_{\mathcal{W}}(X, \cdot) : \mathrm{Grass}(s, r) \to \mathbb{R}$. Its Riemannian Hessian is such that for $\Delta \in \mathrm{T}_{\mathcal{W}}\mathrm{Grass}(s, r)$, $\mathrm{Hess} f_{\mathcal{W}}(\mathcal{W}, X)[\Delta] = -2\mathrm{P}_{\mathcal{W}_\perp} K(x)\Delta$. Hence $\|\mathrm{Hess} f_{\mathcal{W}}(\mathcal{W}, X)\| \leq 2\, \|K(X)\|_{\mathrm{F}}$. Using 4.18, the vector field $\mathrm{grad} f_{\mathcal{W}}(X, \cdot)$ is Lipschitz continuous with constant $L_{\mathcal{W}} = 2\, \|K(x)\|_{\mathrm{F}}$. If the kernel is upper-bounded for all $X \in \mathrm{L}_{\mathcal{A},b}$, the constant $L_{\mathcal{W}}$ is independent of $X$. This implies that (4.13) holds.

We also analyse Lipschitz continuity of the vector field $\mathrm{grad} f_X$.

$$\begin{aligned}
\|\mathrm{grad}_X f(X_1, \mathcal{W}) - \mathrm{grad}_X f(X_2, \mathcal{W})\|_{\mathrm{F}} &= \left\|\mathrm{P}_{\mathrm{TL}_{\mathcal{A},b}} \left(\nabla_X f(X_1, \mathcal{W}) - \nabla_X f(X_2, \mathcal{W})\right)\right\|_{\mathrm{F}} \\
&\leq \|\nabla_X f(X_1, \mathcal{W}) - \nabla_X f(X_2, \mathcal{W})\|_{\mathrm{F}} \\
&\leq \|(\mathrm{DK}(X_1)^* - \mathrm{DK}(X_2)^*)\, \mathrm{P}_{\mathcal{W}_\perp}\|_{\mathrm{F}} \\
&\leq \|\mathrm{DK}(X_1)^* - \mathrm{DK}(X_2)^*\|_2\, \|\mathrm{P}_{\mathcal{W}_\perp}\|_{\mathrm{F}} \\
&\leq \|\mathrm{DK}(X_1) - \mathrm{DK}(X_2)\|_2\, .
\end{aligned}$$

If $\mathrm{DK}(X)$ is $L_x$-Lipschitz over $\mathrm{L}_{\mathcal{A},b}$, we can write

$$\|\mathrm{grad}_X f(X_1, \mathcal{W}) - \mathrm{grad}_X f(X_2, \mathcal{W})\|_{\mathrm{F}} \leq L_x\, \|X_1 - X_2\|_{\mathrm{F}}$$

and $\mathrm{grad}_X f(., \mathcal{W})$ is also $L_x$-Lipschitz, where the constant $L_x$ is independent of $\mathcal{W} \in \mathrm{Grass}(s, r)$. This implies that (4.12) holds. □

The conditions listed in Proposition 4.19 on the kernel and its derivatives can be difficult to verify or satisfy in general. For instance, the Gaussian kernel $K_\sigma$ is bounded above on $\mathrm{L}_{\mathcal{A},b}$, but the monomial kernel $K_d$ is not for any degree $d \geq 1$. For the Gaussian kernel, the map $\mathrm{DK}_\sigma(X)$ is always Lipschitz continuous on $\mathrm{L}_{\mathcal{A},b}$. For the monomial kernel, the map $\mathrm{DK}_d(X)$ is Lipschitz continuous for $d \leq 2$, and only locally Lipschitz continuous for $d \geq 3$.

Fortunately, the picture is much simpler if the sequence of iterates $(X_k)_{k\in\mathbb{N}}$ generated by Algorithm 2 or Algorithm 4 is contained in a bounded set. This ensures that we can find Lipschitz constants such that the bounds in A2, A3 and A8 hold at every iterate of the algorithm (and trial points if any), which is all that is needed in the convergence analysis.

**Proposition 4.20.** *Consider the cost function of either* (4.1) *or* (4.4) *and apply Algorithm 4 or Algorithm 2 with the exponential map as the retraction. If the convex hull of the sequence of iterates* $(X_k)_{k\in\mathbb{N}}$ *and the trial points is a bounded set, then* (2.16)*,* (2.17) *and* (4.12)*-*(4.13) *hold at every iterate* $(X_k, \mathcal{U}_k)_{k\in\mathbb{N}}$ *and trial points of the algorithm.*

*Proof.* Provided the kernel is a smooth function, the derivatives of the cost function are continuous. As a consequence of the Weierstrass theorem, the derivatives are bounded on the closure of the convex hull of the iterates, which is compact (Grass$(s, r)$ is compact). If the Hessian is bounded on the closure of convex hull of the iterates, the gradient is Lipschitz continuous on that set (Proposition 4.18) and therefore A2 and A8 hold with the exponential map as the retraction (Proposition 4.17). The continuity of the third-order derivatives implies A3 in a similar way, which is required for the convergence analysis of the second-order RTR. □

## 4.6 Numerical experiments

In this section we validate our approach with numerical results on randomly generated test problems. We also compare the performances of the different algorithms we propose.

### 4.6.1 Implementation of the algorithms

Our code for nonlinear matrix recovery is available at `https://github.com/flgoyens/nonlinear-matrix-recovery` in both Matlab and Python. We use the Manopt (Boumal et al., 2014) and Pymanopt (Townsend et al., 2016) libraries for optimization on manifold solvers. We denote by `RTR1` and `RTR2` the first- and second-order versions of Algorithm 2, for which we use the default parameters of the Manopt solver. The maximum number of iterations is set at 500 for `RTR1` and `RTR2`. The RTR subproblems are solved using truncated conjugate gradients (Algorithm 3) and the final termination criterion is only a first-order condition (the norm of the gradient) which we set at $10^{-6}$ for `RTR1` and `RTR2`. Pymanopt uses automatic differentiation and does not require to input the derivatives of the cost function. In Manopt, we use the derivatives computed in Section 4.1.2.

`Altmin1` is a first-order version of alternating minimization (Algorithm 4) which uses gradient descent with Armijo line search to solve subproblem (4.7). In the second-order version of Algorithm 4 (`Altmin2`), a second-order trust-region is applied to the minimization of (4.7). The default values for the parameters of Algorithm 4 as well as the Gaussian and monomial kernels are presented in the table below.

| Parameter | Default value | Parameter | Default value |
|---|---|---|---|
| $\varepsilon_x$, $\varepsilon_u$ | $10^{-6}$ | $c$ in (Monomial kernel) | 1 |
| $\varepsilon_{x,k}$ | Equation (4.10) | $\alpha_0$ in Algorithm 5 | 2 |
| $\sigma$ in (Gaussian kernel) | 2.5 | $\tau$ in Algorithm 5 | 0.5 |
| $d$ in (Monomial kernel) | 2 | $\beta$ in Algorithm 5 | $10^{-4}$ |

### 4.6.2 Test problems

We describe the set of parameters that we want to vary and test the dependence of each algorithm with respect to these parameters.

**Union of subspaces** Case study 2 depends on the following parameters: ambient dimension $n$, number of subspaces, dimension of each subspace, number of points on each subspace. To generate a random union of subspaces, we place the same number of points on each subspace and take subspaces of the same dimension. We calculate a basis for a random subspace and generate each point on that subspace by taking a random combination of the columns of that basis.

**Clusters** For Case study 3, the generation of the test problem depends on the following parameters: the number of clusters, the number of points in each cluster and the standard deviation $\sigma_c$ of the clusters. We first generate random centers in $\mathbb{R}^n$. We then add to each center a cluster of points with multivariate Gaussian distribution with zero mean and covariance $\sigma_c^2 \,\mathrm{Id}$ with $\sigma_c = 0.5$.

### 4.6.3 Testing methodology

Throughout, we say that an algorithm successfully recovers the matrix $M \in \mathbb{R}^{n\times s}$ if it returns a matrix $X^*$ such that the root mean square error (RMSE) is below $10^{-3}$,

$$\mathrm{RMSE}(M, X^*) := \|X^* - M\|_{\mathrm{F}} / \sqrt{ns} \leq 10^{-3}.$$

Our goal is to test the ability of our methods to recover the original matrix $M$. We measure the performance against an increase in difficulty of the problem for several parameters. Parameters that increase the difficulty of the recovery include:

1. Reducing the number of measurements $m$;

2. Increasing the rank in the feature space.

In the case of unions of subspaces, for a fixed number of points, the rank of $\Phi_d(M)$ depends on the number and the dimension of the subspaces, as indicated by Proposition 3.1. For clusters, the $\varepsilon$-rank increases with the number of clusters. The undersampling ratio is defined as $\delta = \dfrac{m}{ns}$, it is the number of measurements over the number of entries in $M$. The examples presented in this section are matrix completion problems, for which the observations consist of entries of $M$ selected at random. We present phase transition results to numerically show which geometries can be recovered and which undersampling ratios are needed. Typical phase transition plots for matrix completion vary the undersampling ratio and the rank of the matrix (Tanner and Wei, 2013). For union of subspaces, the rank of the feature space is difficult to control, therefore we vary the number and dimension of the subspaces. For each value of the varying parameter, we generate 10 random matrices $M$ that follow the desired structure. We try to recover each with varying $\delta$ from 0.1 to 0.9 for a random initial guess. If the RMSE is below $10^{-3}$ in the maximum number of iterations allowed by the algorithm, we consider the recovery to be successful. The phase transition plots record which of the 10 random problems is solved for each configuration. In Figures 4.3 through 4.7 the grayscale indicates the proportion of problems solved, with white = 100% of instances solved and black = 0%.

### 4.6.4 Numerical results

**Comparing the performance of RTR and Alternating minimization** In Figure 4.1, we compare `RTR1` and `RTR2` to solve the recovery problem (4.4) of a union of subspaces and see that `RTR1` performs poorly, especially in comparison to the quadratic convergence rate of `RTR2`. This suggests that the conditioning number of the Hessian on $\mathcal{M} = \mathrm{L}_{\mathcal{A},b} \times \mathrm{Grass}(s,r)$ is large, hence first-order methods on $\mathcal{M}$ are not suitable. However, we see below that first-order alternating methods perform well, which suggests that the bad conditioning is a consequence of the product manifold.

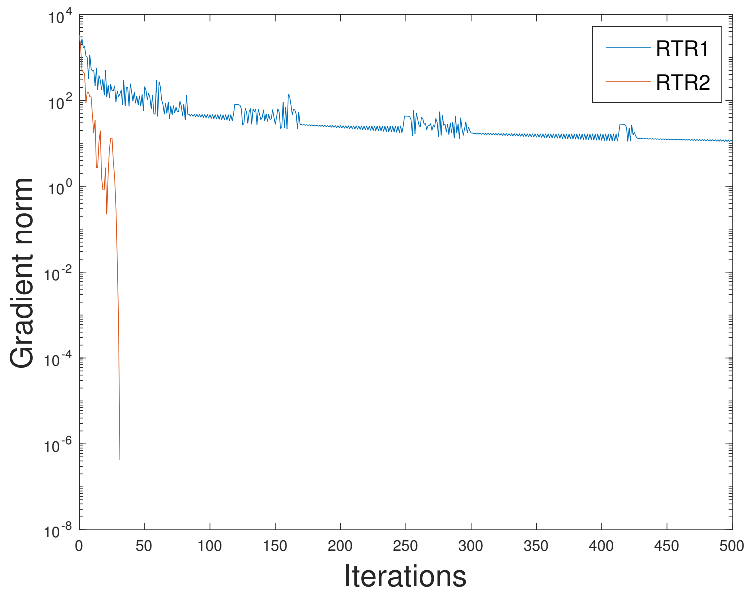


Figure 4.1: First- and second-order Riemannian trust-region

Figure 4.2 compares the performance of `RTR2` (Algorithm 2), `Altmin1` and `Altmin2` which are first- and second-order alternating minimization (Algorithm 4). We choose a problem of matrix completion over a union of subspaces. We find that `RTR2` has a local quadratic rate of convergence, which makes it the method of choice if we want to recover $M$ to high accuracy. Both `Altmin1` and `Altmin2` make faster progress during the early iterations; thus these methods should be considered if the required accuracy is low. Our observations indicated that, usually, the distance to the solution $M$ is of the same order of magnitude as the gradient norm. Hence, using an algorithm such as `RTR2` which terminates with a smaller gradient norm yields a greater accuracy in the recovery of $M$. We noticed that the first-order `Altmin2` and the second-order `Altmin2` typically stall numerically when the gradient norm is around $10^{-6}$. This is less frequent for `RTR2`, which may converge up to a gradient norm below $10^{-13}$.

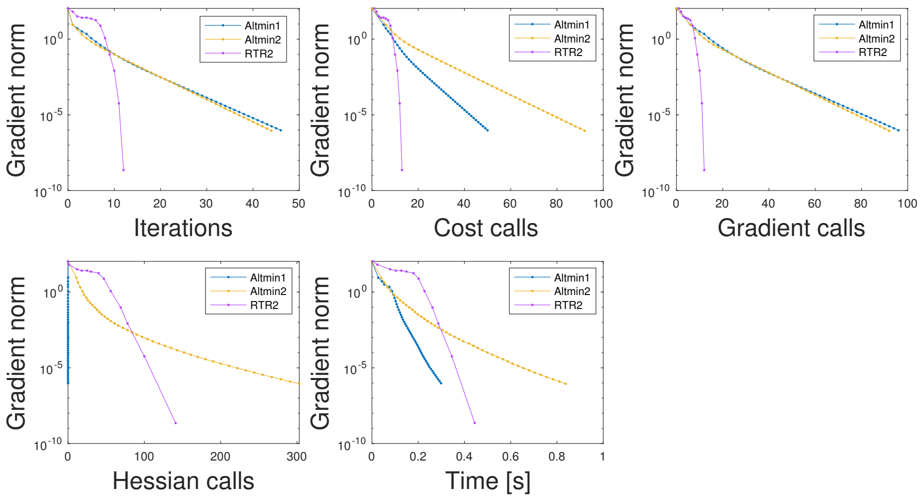


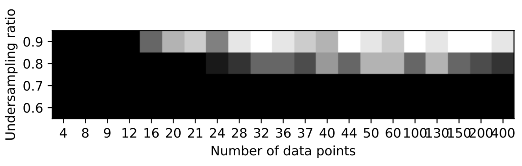
Figure 4.2: Comparing alternating minimization (first-order Altmin1 and second-order Altmin2) with the Riemannian trust-region algorithm (RTR2) for a union of subspaces recovery.

We now illustrate how the parameters at play affect the recovery for matrices that follows a union of subspaces model.

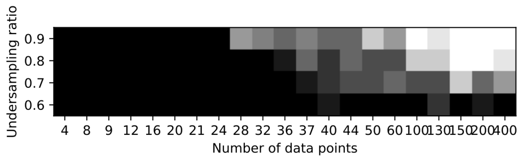
**Degree of the monomial features** Deciding which degree $d$ to use in practice requires consideration. Previous works limit themselves to $d = 2$ and $d = 3$. This is understandable because the dimension of the feature space $N(n, d)$ increases exponentially with $d$. Hence Problem (4.1) becomes practically intractable for even moderate values of $d$ and $n$, for example $N(n = 20, d = 5) = 53130$ and $N(n = 20, d = 2) = 231$.

The other natural option is to solve the kernel-based problem (4.4), where the lifted dimension $N$ does not appear explicitly and the Grassmann has dimension $s \times r$. This is attractive because, a priori, the number of columns $s$ may not be as large as $N$. However, as discussed on page 56, there are requirements on the number of samples (columns of $M$) needed to allow recovery. The matrix $M \in \mathbb{R}^{n\times s}$ must satisfy $s > N - q$, where $q$ is the number of linearly independent vectors $v$ such that $v^\top \Phi_d(M) = 0$. That is, $s$ needs to be large enough so that $\Phi_d(M)$ is rank-deficient. The analysis in (Ongie et al., 2017) shows that the number of points $s$ needed to allow recovery increases exponentially with $d$. For that reason, unless $d$ is small, solving problems where $s$ is large enough for recovery requires specific tools to handle very large scales. The monomial basis is also known to be ill-conditioned for large degrees. This gives two obstacles to the performances of these algorithms when the degree increases.

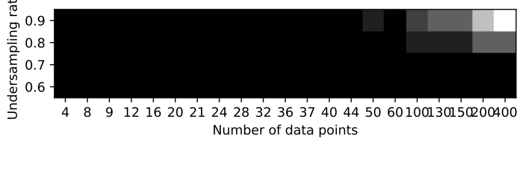
In Figure 4.3, we solve the recovery problems using `RTR2` for an increasing number of columns $s$, and we do that for a monomial kernel of degree one, two and three to compare the recovery that is possible for each degree. In Figure 4.3(a), the degree used is $d = 1$. For $n = 15$, the dimension of the lifted space is $N(15, 1) = 16$. For a large number of data points spread over 4 subspaces of dimension 2, the rank of the monomial kernel is 9. This explains why recovery is impossible when $s \leq 9$, since the kernel is not rank deficient at the solution $M$. In Figure 4.3(b), the degree used is $d = 2$. For $n = 15$, the dimension of the lifted space $N(15, 2) = 136$. For a large number of data points spread over 4 subspaces of dimension 2, the rank of the monomial kernel is 21. This explains why recovery is impossible when $s \leq 21$. In Figure 4.3(c), the degree used is $d = 3$. For $n = 15$, the dimension of the lifted space is $N(15, 3) = 816$. For a large number of data points spread over 4 subspaces of dimension 2, the rank of the monomial kernel is 37. This explains why recovery is impossible when $s \leq 37$. We notice that the recovery is still poor for $s > 37$. In general, $d = 2$ seems to give the best results for the majority of data sets.



(a) $d = 1$



(b) $d = 2$



(c) $d = 3$

Figure 4.3: Phrase transition for data belonging to a union of 4 subspaces of dimension 2 in $\mathbb{R}^{15}$ for an increasing number of data points spread across the subspaces. Each square gives the proportion of problems solved over 50 randomly generated problems, white = 100% of instances recovered and black = 0%,

The dimension of the subspaces that we aim to recover plays a role in the possibility to recover. In Figure 4.4, we increase the dimension of the subspaces while the other parameters of the data remain fixed. For a fixed number of data points $s$, increasing the dimension of the subspaces increases the rank of the monomial features (see Proposition 3.1), and therefore, if the dimension of the subspaces becomes too large, the recovery is compromised. For subspaces of dimension smaller than 4 in $\mathbb{R}^{10}$, we observe good recovery, which also depends on the undersampling ratio.

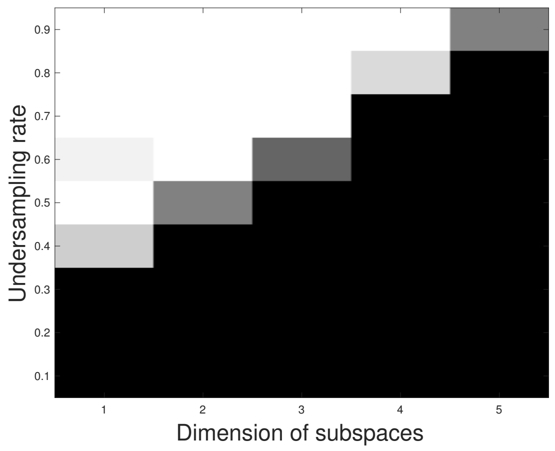
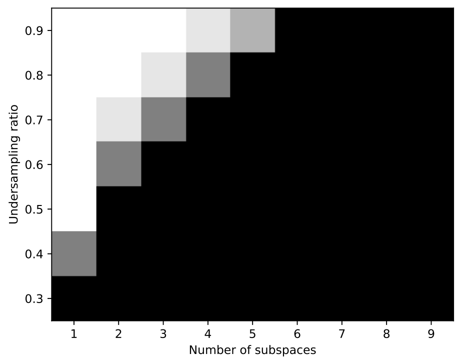


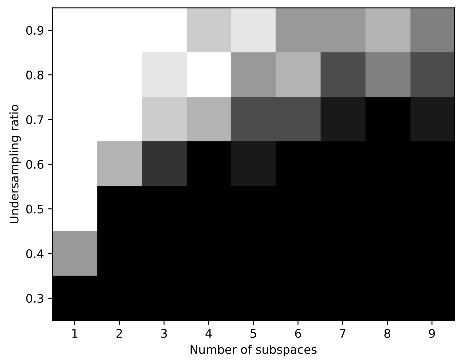
Figure 4.4: Phrase transition for data belonging to a union of 2 subspaces of increasing dimension in $\mathbb{R}^{10}$ with 20 points on each subspace. Each square gives the proportion of problems solved over 50 randomly generated problems, white = 100% of instances recovered and black = 0%, using the polynomial kernel of degree $d = 2$ as lifting.

The same phenomenon is observed when the number of subspaces is increased for a fixed number of data points, see Figure 4.5.

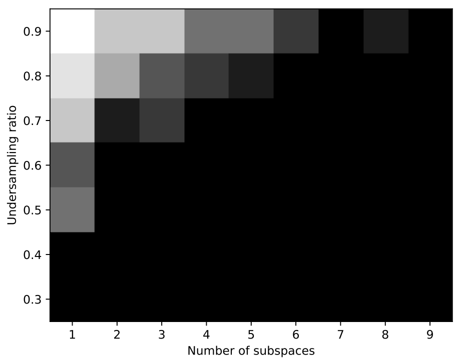
**Clustering with missing data** In the case of clusters (Case study 3 on page 57), there is a noise inherent to the model because the kernel at the solution is only approximately low-rank. More precisely, the matrix $\mathrm{K}_\sigma(M, M)$ has an exponential decay in the singular values, but none of them are exactly zero. Matrices with such singular value patterns are known to be difficult to recover in low-rank matrix completion. We adapt the homotopy strategy outlined in (Vandereycken, 2013) to our formulation. For $n_c$ clusters, we observe that $\sigma_{n_c}(\mathrm{K}_\sigma(M)) \gg \sigma_{n_c+1}(\mathrm{K}_\sigma(M))$, as shown in Figure 3.4, which makes it natural to say that the $\varepsilon$-rank (Equation (3.18)) of $M$ is the number of clusters. Problem (4.4) is first solved with a random variable $\mathcal{U} \in \mathrm{Grass}(s, r = 1)$ and random $X \in \mathrm{L}_{\mathcal{A},b}$. The problem is then solved several times in succession with values of the rank increased by one, using the previous solution to initialize the next solve. Let $(X, U_r) \in \mathrm{L}_{\mathcal{A},b} \times \mathrm{Grass}(s, r)$ denote the solution of the problem solved with rank $r$, the next problem is initialized as $(X, U_{r+1}) \in \mathrm{L}_{\mathcal{A},b} \times \mathrm{Grass}(s, r + 1)$ with $U_{r+1} = \begin{bmatrix} U_r & u \end{bmatrix}$ where $u \in \mathbb{R}^s$ has unit norm



(a) Monomial kernel degree $d = 1$



(b) Monomial kernel degree $d = 2$



(c) Monomial kernel degree $d = 3$

Figure 4.5: Phrase transition for data belonging to a union of an increasing number of subspaces of dimension 2 in $\mathbb{R}^{15}$ with 150 data points spread across the subspaces. Each square gives the proportion of problems solved over 10 randomly generated problems, white = 100% of instances recovered and black = 0%.

and is orthogonal to the vectors in $U_r$. The rank is increased until some value greater than $n_c$, when increasing brings no further progress. The recovery error can usually not be improved beyond a RMSE of $10^{-2}$, which is much less accurate than for union of subspace data (where is rank is exact). The completed matrix $X^*$ returned by this procedure allows to perform a clustering of the columns of $M$ starting from missing entries. We are interested in determining when the matrix $X^*$ yields the same clustering as the matrix $M$. We use the Rand index to measure the compatibility of two different clusterings of the same set (Rand, 1971). The clustering is established as follows. The entries of the matrix $\mathrm{K}_\sigma(X^*, X^*)$ are rounded to the nearest integer, which is either 0 or 1. This gives an adjacency matrix from which a clustering is inferred. That is, if $\mathrm{K}_\sigma(X^*, X^*)_{ij} \geq 0.5$, the points of index $i$ and $j$ are in the same cluster; otherwise they are not. We see in Figure 4.6 that, for 5 clusters or less, the original clustering can be recovered even though a significant percentage of the entries in the original matrix are missing.

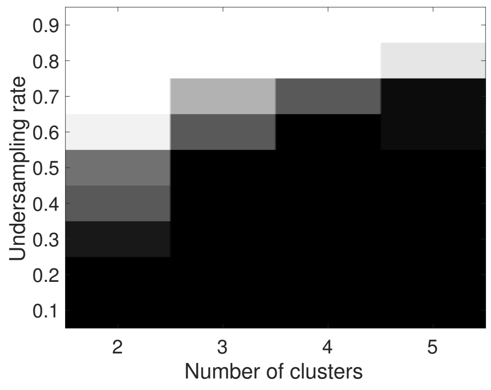


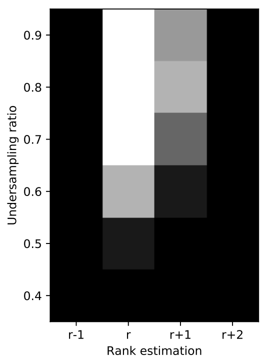
Figure 4.6: Percentage of problems correctly clustered for different sampling rates and increasing number of clusters. Fifty random instances generated in each case; white = 100% of instance correctly clustered and black = 0%. Clusters belong to $\mathbb{R}^5$ with 20 points in each cluster.

**Robustness to a bad estimate of the rank** In the case of a union of subspaces, the monomial features are exactly low-rank and it is important to have an accurate estimation of $\text{rank}(\Phi_d(X))$. Recovery is sometimes possible if the estimation is just above the correct value. If the estimated rank is less than the exact rank or much too large, recovery usually fails. This intuition is guided by the cost function that we use. If the variable $\mathcal{U}$ is artificially constrained to be the leading singular vectors of $\Phi(X)$, the cost function in (4.1) simplifies to

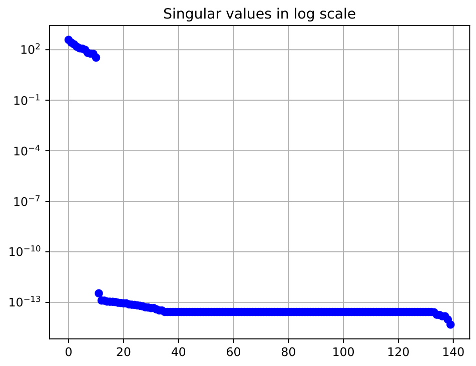
$$\min_X \sum_{i=r+1}^{\min(N,s)} \sigma_i^2(\Phi(X)).$$

The cost function represents the energy in the tail of the singular value decomposition, where $r$ is the estimation of the rank. In Figure 4.7(a), the data belongs to a union of 2 subspaces of dimension 2 in $\mathbb{R}^{15}$, with a total of $s = 150$ data points. With a kernel of degree 2, the lifted dimension is $N(15, 2) = 136$. The phase transition shows that some matrices are recovered when the guess of the rank is off by one.

**Robustness to measurement noise** In applications, it is common to assume some noise on the measurements. Assume that we receive $\mathcal{A}$ and $\tilde{b} := b + \xi$ such that $\mathcal{A}(M) = b$ where $\xi \in \mathbb{R}^m$ is some noise. In the following numerical test, we generate white Gaussian noise, i.e. $\xi_i \sim \mathcal{N}(0, \sigma^2)$ for some variance $\sigma^2$. In the noisy setting, the problem formulation is (4.2) with the penalty parameter $\lambda > 0$ that should be tuned based on the noise level.



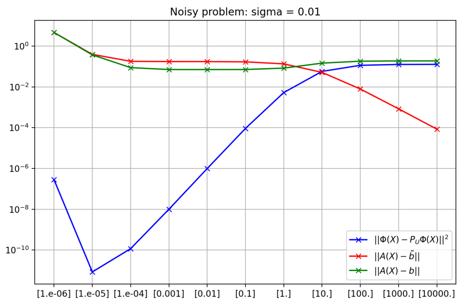
(a) Phase transition



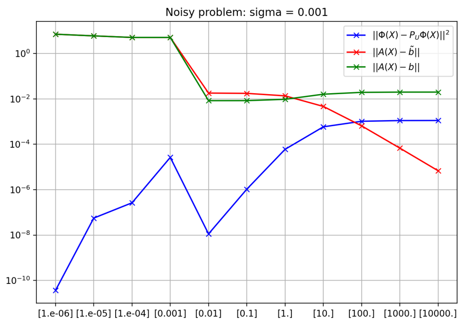
(b) Singular values of $\mathrm{K}_2(M, M)$

Figure 4.7: Impact of an incorrect estimate of the rank for the completion of a union of subspaces. Each square gives the proportion of problems solved over 50 randomly generated problems, white = 100% of instances recovered and black = 0%, using the polynomial kernel of degree $d = 2$ as lifting.

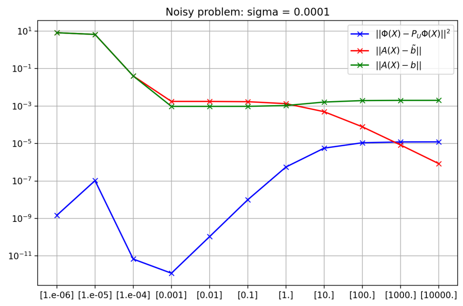
Estimating an appropriate value for $\lambda$ without knowledge of the noise variance $\sigma^2$ is an intricate task. The solution of (4.2) for $\lambda \in [0, \infty[$ represents the trade-off curve between minimization of the rank residual and minimization of the residual on the linear measurements. In practical settings, a user may be able to determine which trade-off is more meaningful for a particular application. To help determine a general strategy, we use a scheme which increases $\lambda$ over successive calls to the solver, while warm starting each solve with the previous solution to (4.2). We have found that starting with the value $\lambda = 10^{-6}$ is satisfactory and we multiply $\lambda$ by a factor 10 at each iteration. Figure 4.8 shows, for three different noise levels, the evolution of the solution of (4.2), labelled $X^*$, as the penalty parameter $\lambda$ increases. We see that where the blue and red lines cross, the green line is still near its lowest point, that is, the solution is still minimizing the true measurement residual as well as for any other value of $\lambda$. This allows to recommend the simple strategy of choosing the value of $\lambda$ where the value of the red and blue curves are the closest (which approximates the value for which they intersect). This choice gives equal weight to the rank minimization and satisfaction of the measurements. Table 4.1 shows the accuracy of the solution $X^*$ for that choice of $\lambda$. We see that both the infeasibility ($\|\mathcal{A}(X^*) - b\|_2$) and the distance to the solution ($\|X^* - M\|_{\mathrm{F}}$) are proportional to the noise level and decrease with the latter. This shows that the warm starting scheme used to tune the parameter $\lambda$ in conjunction with Problem (4.2) handles well the presence of noise in the measurements.



(a) $\sigma = 10^{-2}$



(b) $\sigma = 10^{-3}$



(c) $\sigma = 10^{-4}$

Figure 4.8: Solutions for noisy problems as a function of the parameter $\lambda$ on the horizontal axis.

| Standard deviation | $\left\|\mathcal{A}(X^*) - \tilde{b}\right\|_2$ | $\|\mathcal{A}(X^*) - b\|_2$ | $\|X^* - M\|_\mathrm{F}$ | $\|\mathcal{A}(M) - b\|_2$ |
|---|---|---|---|---|
| $\sigma = 10^{-2}$ | $2 \cdot 10^{-1}$ | $7 \cdot 10^{-2}$ | $8 \cdot 10^{-2}$ | 0.2 |
| $\sigma = 10^{-3}$ | $2 \cdot 10^{-2}$ | $8 \cdot 10^{-3}$ | $8 \cdot 10^{-3}$ | 0.02 |
| $\sigma = 10^{-4}$ | $2 \cdot 10^{-3}$ | $8 \cdot 10^{-4}$ | $9 \cdot 10^{-4}$ | 0.002 |

Table 4.1: Quality of the solution $X^*$ for different levels of noise $\sigma$ in the measurements.

## 4.7 Conclusions

In this chapter, we study the problem of nonlinear matrix recovery, where one tries to recover a high-rank matrix that exhibits a low-rank structure in some feature space. In therms of the use cases considered, in addition to the union of subspaces and algebraic varieties, we propose the use of the Gaussian kernel for clustering problems with missing data, which is a new application of nonlinear matrix completion.

We propose a novel formulation for the nonlinear matrix completion problem which is inspired from low-rank matrix completion techniques using the Grassmann manifold. We then show how Riemannian optimization and alternating minimization methods can be applied effectively to solve this optimization problem. The algorithms we propose, come with strong global convergence results to critical points and worst-case complexity guarantees. In addition, we show that the alternating minimization algorithm converges to a unique limit point using the Kurdyka-Lojasiewicz property.

We provide extensive numerical results that attest to the efficiency of the approach to recover high-rank matrices drawn from unions of subspaces and clustered data. We note that the second-order Riemannian trust-region method allows to recover with high accuracy. We expose the difficulty of using polynomials of high degree in the monomial kernel, as they require an exponentially increasing number of sample points to allow recovery. Our approach proves to be efficient at clustering a data set despite the presence of missing entries and our approach also shows great robustness against the presence of noise in the measurements.

Nonlinear matrix recovery is a relatively new technique, and directions of improvement remain to be investigated. Currently, nonlinear matrix recovery methods present in the literature, including this work, require to build the feature or kernel matrix in full. The scalability of the methods would be much improved if they only relied on a low-rank factorization of the feature matrix, but that is not straightforward to formulate. It might also be possible to use a low-dimensional approximation of the features, in the spirit of (Rahimi et al., 2007).

# Chapter 5

# Applications of the algebraic variety model

In this chapter, we consider two estimation problems which naturally build on the concepts developed in Chapter 4, namely the algebraic variety model (Case study (1), page 53) and the rank minimization of monomial features using Riemannian optimization.

Section 5.1 investigates the denoising of a point cloud that belongs to an algebraic variety. This problem is connected to the matrix completion task in Chapter 4. In this chapter, all the entries of the matrix are available but noisy, and the focus is on recovering the underlying algebraic variety structure of the uncorrupted matrix.

Section 5.2 introduces a new problem: finding the transformation between two given point clouds that have the same shape, but different positions in space. This is known as the point cloud registration problem, which we consider under the assumption that the point clouds to align are well approximated by an algebraic variety. Although the registration problem appears in a variety of settings, our work was inspired by medical imaging applications. Computed tomography (CT) scans are used as dental imaging tools, as they are able to measure a tooth and produce a point cloud that represents its surface. Comparing different scans of the same tooth taken months or years apart can be helpful to assess tooth decay. Therefore, it is desirable to have an automatic way to overlap two scans, which may have been taken from a different angle. The approach we develop leverages the algebraic variety structure of the point clouds to compute a transformation which overlaps two given point clouds.

Each of the two sections in this chapter begins with a review of related work, which puts our contribution into context and highlights the relevance of using the algebraic variety model for the denoising and registration problems.

## 5.1 Denoising an algebraic variety

This section is concerned with the problem of denoising a point cloud which can be approximated by an algebraic variety. Point cloud denoising has already received some attention in the literature, as surface estimation from point clouds plays a central role in numerous imaging applications. Despite the steady increase in accuracy, most available scanning techniques cause severe scanning artefacts such as noise and outliers (Weyrich et al., 2004). Therefore, to apply sophisticated data analysis on point clouds originating from scanners, substantial pre-processing is usually required.

### Related work

Existing approaches for surface approximation from a point cloud fall into two categories. They may divide the surface in patches and work on a local linearization of the surface, such as in (Gong et al., 2010; Wang and Tu, 2013; Deutsch et al., 2018; Hao et al., 2021), or build a global model of the surface, such as in (Weyrich et al., 2004; Bajaj et al., 1995; Subrahmonia et al., 1996; Zwicker et al., 2002; Adamson and Alexa, 2003). The latter approach usually builds a multivariate polynomial model of the surface, whose equations must be computed, resulting in a large-scale optimization problem.

Gong et al. (2010) develop an approximation method by local linearization. Each linear patch is locally denoised and a global denoising is obtained by aligning the local estimates. Amongst local approaches, we should also note filtering based smoothing methods, such as (Hein and Maier, 2007). While linearization approaches have shown to give satisfactory results in a wide range of applications, the approximation of the surface by a collection of small linear patches introduces issues of hyperparameter tuning, such as the number and width of the patches. We present an approach which conceptually circumvents these issues as it computes a global approximation of the point cloud by a single algebraic variety.

### Our contribution and outline of the section

We consider the problem of denoising a data set whose uncorrupted version belongs to an algebraic variety. Our approach uses a polynomial feature map and approximates the minimization of its rank using techniques from Riemannian optimization. Our advantage over local linearization methods is that our approximation of the point cloud by an algebraic variety is global, and does not require the tuning of hyperparameters stemming from the intricate partition of the manifold in patches. Additionally, by considering algebraic varieties, the point cloud may represent a nonsmooth surface, which cannot be handled by local linearization methods used in the denoising of smooth manifolds. Using polynomial features allows us to learn the equations that define the algebraic variety of the data set. This is useful for data analysis tasks, including the registration of point clouds (Section 5.2). In order to validate our approach, we compute the expected error of our denoising procedure using Stein's unbiased risk estimate (SURE). The relevance of our approach is illustrated on synthetic and real data sets, on which we show that we can extract the underlying algebraic variety structure.

### 5.1.1 Problem description

Consider a matrix $M \in \mathbb{R}^{n\times s}$ whose columns all belong to an algebraic variety, as described in Case study 1. We recall this definition for convenience.

**Definition 5.1** (Algebraic variety model). *Let $\mathbb{R}_d[x]$ be the set of real-valued polynomials of degree at most $d$ over $\mathbb{R}^n$. A real (affine)* algebraic variety *of degree $d$ is defined as the roots of a system of finitely many polynomials $P \subset \mathbb{R}_d[x]$:*

$$V(P) = \{x \in \mathbb{R}^n : p(x) = 0 \text{ for all } p \in P\}.$$

*We say that the matrix $X = \begin{bmatrix} x_1 & \dots & x_s \end{bmatrix} \in \mathbb{R}^{n\times s}$ follows an algebraic variety model if every column of $X$ belongs to the same algebraic variety, that is,*

$$x_i \in V(P) \text{ for all } i = 1, \dots, s.$$

Consider a noisy version

$$\hat{M} = M + \omega, \tag{5.1}$$

where $\omega \in \mathbb{R}^{n\times s}$ has small Frobenius norm compared to $M$, $\|\omega\|_{\mathrm{F}} \ll \|M\|_{\mathrm{F}}$. Given $\hat{M}$, the goal is to recover $M$ as accurately as possible, without having knowledge of the specific algebraic variety that the columns of $M$ lie on (Figure 5.1). The distribution of the noisy perturbation $\omega$ may or may not be known. For practical data sets, such as medical scans, the noise may be interpreted as the mismatch between the data and the algebraic variety model. In that case, our approach attempts to find the algebraic variety that best approximates the data.

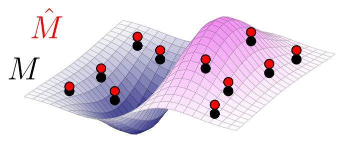


Figure 5.1: Denoising of an algebraic variety

### 5.1.2 The feature space and the monomial features

We briefly recall the connection between an algebraic variety and the monomial map, as outlined in Section 3.2.2. Let

$$N = \binom{n+d}{n},$$

the number of possible monomials of degree less than or equal to $d$ that can be formed with $n$ variables and let $\varphi_d \colon \mathbb{R}^n \to \mathbb{R}^N$ represent the monomial features for some degree $d$ (Equation (3.15)). Consider $X = \begin{bmatrix} x_1 & \dots & x_s \end{bmatrix} \in \mathbb{R}^{n \times s}$ and let

$$\Phi_d(X) = \begin{bmatrix} \varphi_d(x_1) & \dots & \varphi_d(x_s) \end{bmatrix}$$

be the matrix that applies $\varphi_d$ to each column of $X$. The key element of the approach is that the features matrix $\Phi_d(X)$ is rank-deficient when the columns $x_i \in \mathbb{R}^n$ belong to an algebraic variety. Suppose the variety $V(P) \subset \mathbb{R}^n$ is defined by the set of linearly independent polynomials $P = \{p_1, \dots, p_q\}$ where each polynomial $p_i$ is at most of degree $d$, then

$$x_i \in V(P) \text{ for all } i = 1, \dots, s \qquad \text{if and only if} \qquad \Phi_d(X)^\top C = 0, \tag{5.2}$$

where the columns of $C \in \mathbb{R}^{N \times q}$ define the coefficients of the polynomials $p_1, \dots, p_q$ in the monomial basis. Thus, $\operatorname{rank}(\Phi_d(X)) \leq \min(N - q, s)$. This ensures that $\Phi_d(X)$ is rank-deficient when $X$ follows an algebraic variety model, provided that $s > N - q$. In situations where $N \geq s$, the monomial kernel $\mathrm{K}_d$ can be used to represent the monomial features and $\operatorname{rank}(\Phi_d(X)) = \operatorname{rank}(\mathrm{K}_d(X, X))$ for all $X \in \mathbb{R}^{n \times s}$.

### 5.1.3 Denoising as an optimization problem

We formulate the denoising task as an optimization problem. As the previous section explained, one would aspire to minimize the rank of the monomial feature matrix to enforce an algebraic variety model. We do so under the constraint that the solution is not " too far away" from the noisy input $\hat{M}$. Given some $\eta > 0$, an estimate of the noise level, this translates to

$$\begin{cases} \min\limits_{X} & \operatorname{rank}(\Phi_d(X)) \\ & \left\| X - \hat{M} \right\|_{\mathrm{F}}^2 \leq \eta, \end{cases} \tag{5.3}$$

where $\hat{M} \in \mathbb{R}^{n \times s}$ is defined in (5.1) and $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. Solving (5.3) is impractical as the rank is discontinuous. In a fashion similar to Chapter 4, we choose a smooth approximation of the rank using the Grassmann manifold, which appears in (Balzano et al., 2010; Eftekhari et al., 2019). Recall that the Grassmann manifold, written Grass$(N, r)$ represents the set of all subspaces of dimension $r$ in $\mathbb{R}^N$. The Riemannian manifold structure of Grass$(N, r)$, which allows to define optimization algorithms on Grass$(N, r)$, is covered in Section 2.9.5. A point $\mathcal{U} \in \text{Grass}(N, r)$ is represented by a matrix $U \in \text{St}^{N\times r}$ such that range$(U) = \mathcal{U}$. Given an estimation of the rank of the lifted matrix, $r = \text{rank}(\Phi_d(M))$, we formulate the following problem

$$\begin{cases} \min\limits_{X,\mathcal{U}} & \|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}}^2 \\ & \mathcal{U} \in \text{Grass}(N, r) \\ & \left\|X - \hat{M}\right\|_{\mathrm{F}}^2 \leq \eta, \end{cases} \tag{5.4}$$

where $\mathrm{P}_{\mathcal{U}} = UU^\top \in \mathbb{R}^{N\times N}$ is the orthogonal projection on the subspace $\mathcal{U}$. In (5.4), the cost function is nonconvex but smooth. This cost function minimizes the components of $\Phi_d(X)$ contained outside the span of $\mathcal{U}$, so that when the cost is zero, all the columns of $\Phi_d(X)$ belong to the r-dimensional subspace $\mathcal{U}$ and $\Phi_d(X)$ has rank $r$.

An alternative formulation is the following, for some $\lambda \geq 0$:

$$\begin{cases} \min\limits_{X,\mathcal{U}} & f(X,\mathcal{U}; \hat{M}) := \|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}}^2 + \lambda \left\|X - \hat{M}\right\|_{\mathrm{F}}^2 \\ & \mathcal{U} \in \text{Grass}(N, r). \end{cases} \tag{5.5}$$

This formulation is easier to solve, as it is unconstrained in the variable $X$. Problem (5.5) is similar to (4.2), our noisy formulation for matrix recovery. Both receive noisy measurements, but all the entries are available in the former. There exists a $\lambda \geq 0$ such that (5.4) and (5.5) are equivalent. We focus on solving (5.5) in this section.

### Riemannian optimization and software

Through the lens of Riemannian optimization, which we covered in Chapter 2, Problem (5.5) can be viewed as the unconstrained minimization of a function defined on the manifold $\mathcal{M} = \mathbb{R}^{N\times s} \times \text{Grass}(N, r)$. This general framework allows us to use various off-the-shelf Riemannian optimization methods. We apply the second-order Riemannian trust-region (Algorithm 2) to formulation (5.5). The Riemannian trust-region method (RTR) is expected to converge (at least) superlinearly near a non-degenerate minimizer (Absil et al., 2007) and global rates of convergence to second-order stationary points were recently shown (Boumal et al., 2019).

Problem (5.5) is nonconvex but we have good initial guesses for the variables X and $\mathcal{U}$, which are respectively $\hat{M}$ and the first $r$ singular vectors of $\Phi_d(\hat{M})$. Therefore, for the examples considered in Section 5.1.5, we observe numerically that the denoising procedure is robust and appears to find the global minimum reliably despite the nonconvexity. Let us emphasize that in order to solve (5.5), we do not assume knowledge of the noise level

$\eta$. We have yet to detail how to choose a suitable value for the regularization parameter $\lambda$. We use a scheme which does not require an estimation of the noise level $\eta$. The problem is first solved with a small value of $\lambda = 10^{-6}$. The problem is then solved again with increasing values of $\lambda$, with the previous solution used as initial guess for the next solve. This process is stopped when a good trade-off between the values of $\|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}}$ and $\left\|X - \hat{M}\right\|_{\mathrm{F}}$ is reached.

Various other methods could be used to minimize (5.5), such as an alternating minimization over the variable $X$ and $\mathcal{U}$, as proposed in Section 4.3. In this section, we choose to use a Riemannian optimization method on $\mathbb{R}^{N\times s} \times \mathrm{Grass}(N, r)$, which allows to seamlessly employ second-order methods despite the Grassmann constraint. For surface estimation applications, where high accuracy is required, second-order methods have the significant advantage of having a superlinear convergence rate near limit points. In our experience, alternating methods do not seem to allow for local superlinear convergence.

### Learning the equations that define the algebraic variety

We can draw a connection with manifold learning problems which attempt to uncover the structure in high-dimensional data sets and perform a nonlinear dimensionality reduction (Cayton, 2005). If $\|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}} = 0$, then the range of $\Phi(X)$ has dimension at most $r$. The orthogonal complement to $\mathcal{U} = \mathrm{range}(\Phi(X))$, which is also the null space of $\Phi(X)^\top$, has dimension at least $N - r$. A basis for $\mathrm{null}(\Phi(X)^\top)$ gives coefficients of polynomial equations satisfied by every column in $X$, as shown by Equation (5.2). Concretely,

$$\begin{aligned}
\|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}} = 0 &\iff \mathcal{U}^\perp \perp \mathrm{range}(\Phi_d(X)) \\
&\iff \mathcal{U}^\perp \in (\mathrm{range}(\Phi_d(X)))^\perp \\
&\iff \mathcal{U}^\perp \in \mathrm{null}\left(\Phi_d(X)^\top\right) \\
&\iff \Phi_d(X)^\top U^\perp = 0 \qquad (5.6)
\end{aligned}$$

for any $U^\perp \in \mathrm{St}(N, N - r)$ such that $\mathrm{range}(U^\perp) = \mathcal{U}^\perp$. The columns of $U^\perp$, viewed as coefficients in the monomials basis $\varphi_d$, define $N - r$ linearly independent polynomials of $n$ variables. Consider the algebraic variety $V_{U^\perp}$, which is defined by these polynomials. Naturally, the variety $V_{U^\perp}$ does not depend on the choice of $U^\perp$ as a basis for the subspace $\mathcal{U}^\perp$. Using Equation (5.2) and (5.6), we deduce that

$$\|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}} = 0 \qquad \text{if and only if} \qquad x_i \in V_{U^\perp} \text{ for all } i = 1, \ldots, s \qquad (5.7)$$

where $x_1, \ldots, x_s$ denote the columns of $X \in \mathbb{R}^{n\times s}$.

For a given $\hat{M}$, solving (5.5) is expected to return a point $(X,\mathcal{U})$ such that the quantity $\|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}}$ is small, but not necessarily zero. In that case, the variable $\mathcal{U}^{\perp}$ defines an algebraic variety $V_{U^{\perp}}$, and the columns of $X$ approximately belong to that variety. Let $\{p_1, \dots, p_q\}$ be the polynomials that define the algebraic variety $V_{U^{\perp}}$, whose coefficients are given by $U_{\perp} = \begin{bmatrix} u_1^{\perp} & u_2^{\perp} & \cdots & u_q^{\perp} \end{bmatrix} \in \mathbb{R}^{N\times q}$ in the monomial basis, with $q = N - r$. The residual expresses the least square error of the polynomial system that defines the algebraic variety $V_{U^{\perp}}$, calculated for every column of $X$:

$$\|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}}^2 = \left\|U_{\perp}^{\top}\Phi_d(X)\right\|_{\mathrm{F}}^2 = \sum_{i=1}^{s}\sum_{j=1}^{q}((u_j^{\perp})^{\top}\varphi_d(x_i))^2 = \sum_{i=1}^{s}\sum_{j=1}^{q} p_j(x_i)^2. \quad (5.8)$$

Hence, despite the presence of noise, our approach allows recovery of the polynomial equations of an algebraic variety that the data approximately belongs to. Recovering the equations of the variety that the data belongs to is instrumental for the next section, where we estimate a transformation such that two algebraic varieties overlap (Section 5.2).

**Choosing an estimate of the rank**

In order to solve the denoising problem (5.5), an estimate of the rank of $\Phi_d(M)$ must be available. The rank of $\Phi_d(M)$ depends in a complex way on the intrinsic dimension of the variety and the degree $d$. This makes it difficult to know the rank a priori from $\hat{M}$. Upper bounds exist based on the dimensions of the algebraic variety (Cox et al., 1994). Specific information about the problem can often help in finding an estimate of the rank. If the algebraic variety is a surface of dimension $n-1$ in $\mathbb{R}^n$ that is described by one polynomial equation, then $r = N - 1$. The next section provides an estimation of the mean square error of the recovery in the case of Gaussian noise, which gives an estimation of the error at a solution which is obtained for a given value of the rank.

### 5.1.4 Statistical error estimation

In this section, we assume that the noise perturbation $\omega$ in (5.1) follows a normal distribution, namely, for some $\sigma > 0$, every entry $\omega_{ij} \sim \mathcal{N}(0, \sigma^2)$. Let $\left(X^*(\hat{M}), \mathcal{U}^*(\hat{M})\right)$ be an isolated local minimizer of (5.5) for a given matrix $\hat{M}$. We view $X^*(\hat{M})$ as an estimator of $M$ and use Stein's unbiased risk estimate (SURE) to estimate the error of our denoising procedure (Stein, 1981). The expected error is defined as $R = \mathbb{E}_{\omega}\left[\left\|M - X^*(\hat{M})\right\|_{\mathrm{F}}^2\right]$, where the expectation is taken over realizations of $\omega$. Based on (Stein, 1981), we have

$$\hat{R} = \left\|\hat{M} - X^*(\hat{M})\right\|_{\mathrm{F}}^2 - ns\sigma^2 + 2\sigma^2 \sum_{i=1}^{n}\sum_{j=1}^{s} \frac{\partial X_{ij}^*(\hat{M})}{\partial \hat{M}_{ij}}, \quad (5.9)$$

as an unbiased estimate for $R$, i.e. $\mathbb{E}(\hat{R}) = R$. The estimate $\hat{R}$ is called *Stein's unbiased risk estimate* (SURE). The quantity $\sum_{i=1}^{n}\sum_{j=1}^{s} \partial X_{ij}^*(\hat{M})/\partial \hat{M}_{ij}$ is known as the divergence of $X^*(\hat{M})$ and describes the sensitivity of the solution to the input data. It measures the *complexity* of the model, that is, it's tendency to *overfit* the data (Ghojogh and Crowley, 2019, Eq. 33). For the sake of example, if a point cloud is approximated by a line in the least square sense. Slightly modifying a point in the training set will have little effect on the regression, because the model is not complex and is underfitting. On the other hand, if a regression model were to pass through all the data points, it would change noticeably with a change in the training set, this model is very complex and overfitting. In this simple example, the first regression has a small divergence and the second model has a sensibly larger divergence.

We propose the use of SURE as an alternative to cross validation to evaluate the estimation error and detect the possibility of overfitting. For monomials of degree 2, which we use predominantly, the model is simple and the likelihood of overfitting is small. Estimating the generalization error with formula (5.9) is most useful to estimate the error for various model parameters such as the degree $d$ and the rank $r$, without having to resort to a cross validation procedure.

To compute the divergence in the context of Problem (5.5), we use a version of the implicit function theorem for functions defined on manifolds. For a smooth map $F\colon \mathcal{M}_1 \times \mathcal{M}_2 \to \mathcal{M}_3$, we write $\mathrm{D}_1F$ and $\mathrm{D}_2F$ for the differential of $F$, as defined in (2.2), with respect to its first and second argument, respectively.

**Theorem 5.1** ((Abraham et al., 2012), Prop. 3.3.13)**.** *Let $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ be manifolds. Let $F\colon \mathcal{M}_1 \times \mathcal{M}_2 \to \mathcal{M}_3$ be smooth and let $(x_0, y_0) \in \mathcal{M}_1 \times \mathcal{M}_2$ with $F(x_0, y_0) = 0$. If $\mathrm{D}_2F(x_0, y_0)$ is invertible, there exists open neighbourhoods $V_1$ of $x_0$ in $\mathcal{M}_1$ and $V_2$ of $y_0$ in $\mathcal{M}_2$, and a smooth function $g : V_1 \to V_2$ such that for all $x \in V_1$, $F(x, g(x)) = 0$. In addition,*

$$\mathrm{D}g(x_0) = -\left(\mathrm{D}_2F(x_0, y_0)\right)^{-1} \mathrm{D}_1F(x_0, y_0).$$

Recall that the root mean square error is defined by $\mathrm{RMSE} = \left\|M - X^*\right\|_{\mathrm{F}} / \sqrt{ns}$.

**Proposition 5.2.** *Let the noise $\omega \sim \mathcal{N}(0, \sigma^2)$ for some given $\sigma \geq 0$ and consider $\left(X^*(\hat{M}), \mathcal{U}^*(\hat{M})\right)$, an isolated local minimizer of the function $f$ defined in* (5.5) *for a given $\hat{M}$. Stein's Unbiased risk estimate of the RMSE is given by*

$$\mathrm{SURE} := \sqrt{\hat{R}/ns} \tag{5.10}$$

*where $\hat{R}$ is computed as in* (5.9) *with*

$$\frac{\partial X^*(\hat{M})}{\partial \hat{M}} = -\left[\nabla^2_{XX} f(X, \mathcal{U}; \hat{M})\right]^{-1} \frac{\partial}{\partial \hat{M}}\left(\nabla_X f(\hat{M}, g(\hat{M}); \hat{M})\right). \tag{5.11}$$

*where*

$$\frac{\partial}{\partial \hat{M}_{ij}}\Big(\nabla_X f(\hat{M}, g(\hat{M}); \hat{M})\Big) = -2\lambda\Delta_{ij}$$

*and* $\Delta_{ij} \in \mathbb{R}^{n\times s}$ *has entry* $(i,j)$ *equal to one and the other entries are zero.*

*Proof.* For readability, we write $(X^*,\mathcal{U}^*)$ for $\Big((X^*(\hat{M}),\mathcal{U}^*(\hat{M})\Big)$. Recalling that $\mathcal{M} = \mathbb{R}^{n\times s} \times \mathrm{Grass}(N,r)$, we define $F\colon \mathbb{R}^{n\times s} \times \mathcal{M} \to \mathrm{T}\mathcal{M}$ by

$$F(\hat{M}, X,\mathcal{U}) = \mathrm{grad}_{\mathcal{M}} f(X,\mathcal{U};\hat{M}) = \begin{bmatrix} \nabla_X f(X,\mathcal{U};\hat{M}) \\ \mathrm{grad}_{\mathcal{U}} f(X,\mathcal{U};\hat{M}) \end{bmatrix}. \tag{5.12}$$

where $\nabla_X f(X,\mathcal{U};\hat{M}) \in \mathbb{R}^{n\times s}$ is the Euclidean gradient of $f$ with respect to $X \in \mathbb{R}^{n\times s}$ and $\mathrm{grad}_{\mathcal{U}} f(X,\mathcal{U};\hat{M}) \in \mathrm{T}_{\mathcal{U}}\mathrm{Grass}(N,r)$ is the Riemannian gradient which belongs to the tangent space of $\mathrm{Grass}(N,r)$ at $\mathcal{U}$. First-order necessary optimality conditions for Problem (5.5) can be stated as

$$F(\hat{M}, X,\mathcal{U}) = 0.$$

We apply Theorem 5.1 to the function $F$, with the identification of the variables $\hat{M} \in \mathcal{M}_1 = \mathbb{R}^{n\times s}$ and $z = (X,\mathcal{U}) \in \mathcal{M}_2 = \mathcal{M} = \mathbb{R}^{n\times s} \times \mathrm{Grass}(N,r)$. The theorem applies to derivatives as abstract objects defined by tangent vectors. In particular, affine connections (Definition 2.14) — including the Riemannian connection — satisfy the axioms of derivatives. When applying this theorem to the function $F$ defined in Equation (5.12), we may view the derivative of a vector field as the covariant derivative associated to the Riemannian connection. Therefore, the derivative of the map $F$ with respect to its second variable $z \in \mathcal{M}$, is given by the Riemannian Hessian of $f$, that is, $\mathrm{D}_2 F(\hat{M}, X,\mathcal{U}) = \mathrm{Hess}_{\mathcal{M}} f(X,\mathcal{U};\hat{M})$. Therefore, if $\mathrm{Hess}_{\mathcal{M}} f(X^*,\mathcal{U}^*;\hat{M})$ is invertible, there exists an open set $V_1 \in \mathbb{R}^{n\times s}$ and a map $g\colon V_1 \to \mathcal{M}$ such that $g(\hat{M}) = (X^*,\mathcal{U}^*)$ and $F(\hat{M}, g(\hat{M})) = 0$. Also,

$$\frac{\partial g(\hat{M})}{\partial \hat{M}} = \frac{\partial}{\partial \hat{M}}\Big((X^*(\hat{M}),\mathcal{U}^*(\hat{M})\Big) = -\Big[\mathrm{Hess}_{\mathcal{M}} f(X^*,\mathcal{U}^*;\hat{M})\Big]^{-1}\left[\frac{\partial F(\hat{M}, g(\hat{M}))}{\partial \hat{M}}\right]. \tag{5.13}$$

If $(X^*,\mathcal{U}^*)$ is an isolated minimizer of (5.5), the matrix $\mathrm{Hess} f(X^*,\mathcal{U}^*)$ is positive definite and therefore invertible. In order to compute $\partial F(\hat{M}, g(\hat{M}))/\partial \hat{M}$, we see from the definition of $f(X,U;\hat{M})$ (5.5) that $\mathrm{grad}_{\mathcal{U}} f(X,U;\hat{M})$ does not depend on $\hat{M}$ and that $\nabla_X f(\hat{M}, g(\hat{M});\hat{M})$ depends on $\hat{M}$ through the term $2\lambda(X - \hat{M})$. Thus, we have

$$\frac{\partial}{\partial \hat{M}}\Big(\mathrm{grad}_{\mathcal{U}} f(\hat{M}, g(\hat{M});\hat{M})\Big) = 0 \tag{5.14}$$

and

$$\frac{\partial}{\partial \hat{M}_{ij}}\Big(\nabla_X f(\hat{M}, g(\hat{M}); \hat{M})\Big) = -2\lambda\Delta_{ij}$$

with where $\Delta_{ij} \in \mathbb{R}^{n\times s}$ has entry $ij$ equal to one and the other entries are zero. Using (5.14), the upper block of (5.13) reads

$$\frac{\partial X^*(\hat{M})}{\partial \hat{M}} = -\left[\nabla^2_{XX} f(X,\mathcal{U};\hat{M})\right]^{-1} \frac{\partial}{\partial \hat{M}}\Big(\nabla_X f(\hat{M}, g(\hat{M});\hat{M})\Big).$$

This gives (5.11), and (5.10) follows from the definition of the RMSE and (5.9). □

### 5.1.5 Numerical results

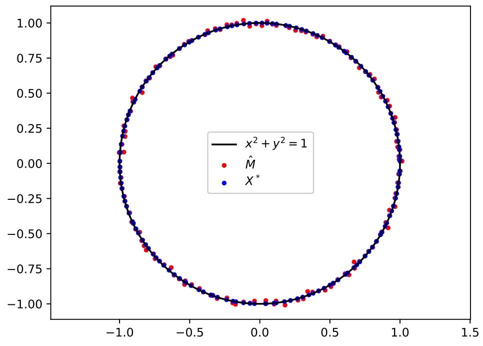
In this section, we numerically evaluate the performance of our denoising approach. Our code for the denoising of algebraic varieties can be found at `https://github.com/flgoyens/variety-denoising`, with implementations in both Matlab and Python. We use the Manopt (Boumal et al., 2014) and Pymanopt (Townsend et al., 2016) toolboxes for the Matlab and Python versions of the code, respectively. The RTR solver (Algorithm 2) in the toolbox is used with default parameters on problem (5.5). The numerical examples in this section were generated using the Matlab version of our code. The algorithm was set to stop when the norm of the Riemannian gradient in (5.5) was smaller than $10^{-6}$. For an output of the algorithm, labelled $(X^*,\mathcal{U}^*)$, we define

$$\text{RESIDUAL} = \|\Phi_d(X^*) - \mathrm{P}_{\mathcal{U}^*}\Phi_d(X^*)\|^2_{\mathrm{F}}\,,$$

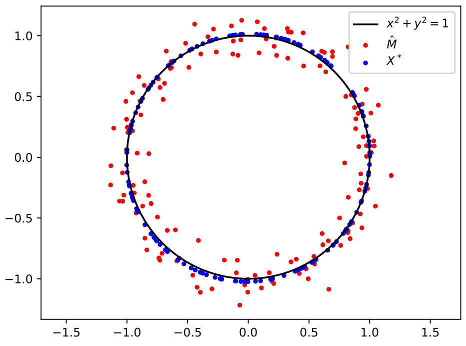
which is interpreted, according to (5.8), as the least squares infeasibility for the system of polynomial equations of the algebraic variety $V_{\mathcal{U}^*_\perp}$, calculated for every column of $X^*$. The RESIDUAL tells us how close the columns of $X^*$ are to some algebraic variety. The RMSE tells us how close $X^*$ is to the original matrix $M$. We begin with synthetic examples as a proof of concept.

#### Synthetic examples

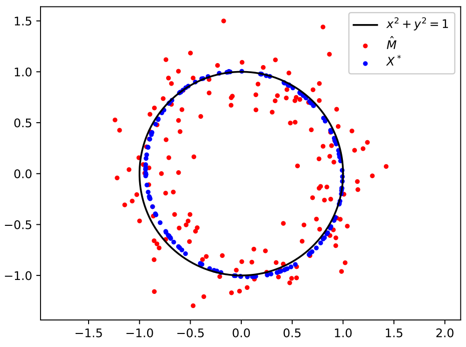
**Example 1: Denoising a circle** We generate a point cloud $\hat{M} \in \mathbb{R}^{2\times 150}$ as a noisy corruption of a set of points on the unit circle using $\sigma = \{10^{-3}, 10^{-2}, 10^{-1}, 2\cdot 10^{-1}\}$ for the standard deviation of the noise ($\hat{M}$ in red in Figure 5.2). We use a degree of 2 for the features and notice that the solution $X^*$ (blue) reached by the optimization algorithm is close to the original circle even for a visually large noise. For $\sigma = 10^{-2}$ and $\sigma = 2\cdot 10^{-1}$, the RESIDUAL reaches values of $10^{-9}$ and $10^{-5}$, respectively. The solver's output $X^*$ is therefore very well approximated an algebraic variety. This example also makes it clear that the approach is more general than a classical least-squares polynomial approximation, as the approximation does not need to be the image of a polynomial function. Table 5.1 reports the RMSE and Stein's Unbiased Risk Estimate (5.10). This shows that SURE accurately predicts the RMSE and that the RMSE increases proportionally with the standard deviation of the noise.



(a) $\sigma = 10^{-2}$



(b) $\sigma = 10^{-1}$



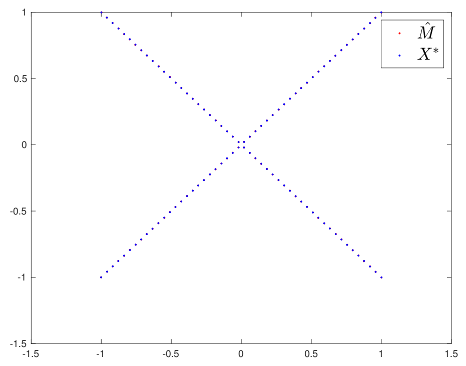
(c) $\sigma = 2 \cdot 10^{-1}$

Figure 5.2: Denoising a circle.

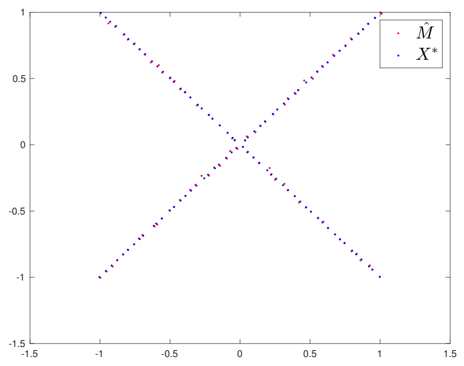
**Example 2: Denoising a union of two subspaces** In Figure 5.3, we denoise a point cloud which is near the union of two subspaces with $\sigma = \{10^{-3}, 10^{-2}, 10^{-1}\}$ for the standard deviation of the noise. This shows that the algebraic variety doesn't need to be a smooth set. It is an algebraic variety described by polynomial equations of degree 2 (since there are two subspaces), hence we use monomial features of degree 2. For $\sigma = 10^{-2}$, the output satisfies RESIDUAL $= 3 \cdot 10^{-10}$. Table 5.2 reports the RMSE and Stein's unbiased risk estimate.

| Noise $\sigma$ | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ | $2 \cdot 10^{-1}$ |
|---|---|---|---|---|
| SURE | $5.49 \cdot 10^{-4}$ | $6.76 \cdot 10^{-3}$ | $5.69 \cdot 10^{-2}$ | $1.22 \cdot 10^{-1}$ |
| RMSE | $6.77 \cdot 10^{-4}$ | $8.53 \cdot 10^{-3}$ | $7.52 \cdot 10^{-2}$ | $1.44 \cdot 10^{-1}$ |

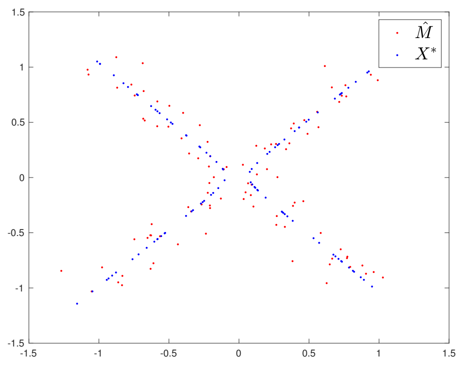
Table 5.1: SURE for denoising of a circle



(a) $\sigma = 10^{-3}$



(b) $\sigma = 10^{-2}$



(c) $\sigma = 10^{-1}$

Figure 5.3: Denoising a nonsmooth algebraic variety.

### Dental Tomography Scans

Through industrial partners, we received a data set of dental X-ray computed tomography scans (XCT). This was part of the original motivation for this research. These scans are composed of a very large number of points, on the order of several millions. It is desirable to remove the noise before they can be processed. Figure 5.4 shows an example of the dental scans available. The point cloud has been randomly subsampled to make it less dense and possible to visualize. The remaining point cloud has dimension $3 \times 2048$, so it is less large.

For complex data sets, such as dental scans, it may not be obvious which degree of the

| Noise $\sigma$ | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ |
|---|---|---|---|
| SURE | $7.05 \cdot 10^{-4}$ | $6.32 \cdot 10^{-3}$ | $7.04 \cdot 10^{-2}$ |
| RMSE | $7.10 \cdot 10^{-4}$ | $6.98 \cdot 10^{-3}$ | $7.37 \cdot 10^{-2}$ |

Table 5.2: SURE for denoising of a union of subspaces

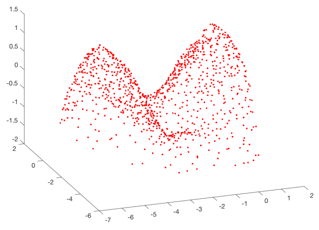


Figure 5.4: Tomographic dental scans (subsampled)

features will yield the best result. In theory, increasing the degree should give a model with more degrees of freedom and a better fit. The dimension of $N$ of feature space increases approximately as $n^d/d!$. As $N$ increases, the number of samples $s$ necessary for the approach to work also increases, as described on page 56. Indeed, if $s \leq N - q$ where $q$ is the number of equations that define the variety, the matrix $\Phi_d(M)$ will not be rank-deficient. In short, increasing the degree requires an exponentially increasing number of samples $s$, which directly increases the dimension of the optimization problem. Additionally, ill-conditioning may appear when the degree $d$ becomes large, because $\Phi_d(X)$ is a multivariate Vandermonde matrix. Therefore, we try values of $d$ going from 1 to 5 and choose the one that yields the best result, which is usually $d = 2$ in practice.

**Example 3: approximation of a dental scan by an algebraic variety** Figure 5.5 shows an XCT scan in red and the output of the algorithm in blue. The value RESIDUAL has been reduced from the order of 1 to the order of $10^{-3}$ by the algorithm, and $X^*$ is therefore closer to an algebraic variety than $\hat{M}$. Since no noise was artificially added, the value of $\sigma$ for the noise that represents the mismatch between the real dental scan and the algebraic variety model is unknown. For a value of $\sigma = 10^{-2}$, the output satisfies SURE = 0.21.

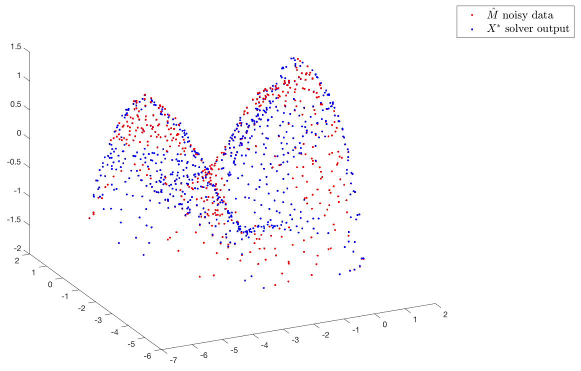


Figure 5.5: Original point cloud and denoised version

## 5.2 Registration

The registration problem consists in aligning point clouds using a specific set of transformations. The problem of point cloud registration has applications in a number of fields, including computer vision (Sharp et al., 2004; Tzeneva, 2011; Williams and Bennamoun, 2000), in distributed approaches to molecular conformation (Cucuringu et al., 2012b; Fang and Toh, 2013; Biswas et al., 2008), sensor network localization (Cucuringu et al., 2012a; Biswas et al., 2006) and aligning data from magnetic resonance imaging with computer-aided tomography scans (Hill et al., 1991; Studholme et al., 1995). It can be instrumental in merging multiple data sets into a globally consistent model or mapping a new measurement to a known data set. Surveys of this topic can be found in (Huang et al., 2021; Bellekens et al., 2014). We describe some of the main approaches.

### Related work

Several approaches have been devised for the various types of registration problems that appear in image and pattern analysis. Registration algorithms can be classified into rigid and non-rigid approaches. The rigid registration is concerned with finding affine transformations that preserve distances, typically rotations and translations. The non-rigid registration problem includes finding affine transformations (which may include scaling and a shear mapping) as well as non-linear transformations.

One of the earliest instances of registration in the literature is the (orthogonal) Procruste problem (Schönemann, 1966). Consider the orthogonal group

$$\mathrm{O}(n) = \left\{Q \in \mathbb{R}^{n\times n} : Q^\top Q = QQ^\top = \mathrm{I}_n\right\},$$

which is a Riemannian manifold, as described in Section 2.9.4. Given two matrices $A, B \in \mathbb{R}^{n\times s}$, one tries to find the orthogonal matrix $Q^* \in \mathrm{O}(n)$ that best matches $A$ to $B$ in Frobenius norm, that is,

$$Q^* = \operatorname*{arg\,min}_{Q\in\mathrm{O}(n)} \|QA - B\|_\mathrm{F}.$$

This problem is equivalent to finding the nearest orthogonal matrix to $BA^\top$,

$$Q^* = \operatorname*{arg\,min}_{Q\in\mathrm{O}(n)} \left\|Q - BA^\top\right\|_\mathrm{F},$$

which, given the singular value decomposition $BA^\top = U\Sigma V^\top$, is $Q^* = UV^\top$.

A natural extension is to consider a transformation which combines an orthogonal matrix with a translation. Consider two point clouds $M_1 = \{x_1, \dots, x_s\}$ and $M_2 =$

$\{y_1, \ldots, y_s\}$ in $\mathbb{R}^n$, respectively called the *source* and the *target* point clouds, as one is obtained through a *rigid transformation* of the other:

$$y_k = Qx_k + a \qquad\qquad k = 1, \ldots, s, \tag{5.15}$$

where $Q \in \mathrm{O}(n)$ is an orthogonal matrix and $a \in \mathbb{R}^n$ is a translation vector. In this setting, *exact point matching* is assumed: the two point clouds have the same number of samples and each sample matches with one and only one sample from the other point cloud. This leads to the following least squares problem

$$(Q^*, a^*) = \underset{Q \in \mathrm{O}(n), a \in \mathbb{R}^n}{\arg\min} \sum_{k=1}^{s} \|y_k - Qx_k - a\|_2^2 . \tag{5.16}$$

The optimization over $\mathrm{O}(n) \times \mathbb{R}^n$ seems difficult since the set $\mathrm{O}(n)$ is nonconvex. Remarkably, there exists a closed-form solution for the global minimizer of (5.16). Arun et al. (1987) show that the optimal orthogonal matrix is given by $Q^* = UV^\top$ where $U\Sigma V^\top$ is the SVD of

$$\sum_{k=1}^{s} (x_k - x_c)(y_k - y_c)^\top,$$

with $x_c = (x_1 + \cdots + x_s)/s$ and $y_c = (y_1 + \cdots + y_s)/s$, the centroids of the two point clouds. The optimal translation is $a^* = y_c - Q^* x_c$. In the case of three or more point clouds to align, no closed-form solution is known. Krishnan et al. (2005) consider such case of multiple point clouds using a rigid transformation and exact point matching. They propose to solve a single optimization problem with one variable on $\mathrm{SO}(n)$ for each point cloud. Chaudhury et al. (2015) also consider an arbitrary number of point clouds and show how the least square formulation, which generalizes (5.16) to several point clouds, can be relaxed into a convex program, with guarantees of tightness.

These approaches, as well as the Procruste problem, require knowledge of the *point correspondence*. That is, it is known at the outset which point from the target point cloud corresponds to a given point in the source point cloud. In a number of practical applications, the point correspondence is unknown. If exact point matching is assumed, the crux of the problem then resides in finding the correct point correspondence. Indeed, given the correspondence, the registration is solved using a single SVD to find the global solution of the least squares problem (5.16).

The iterative closest point (ICP) algorithm, introduced in (Besl and McKay, 1992), addresses the issue of unknown point correspondence. Assuming exact point matching, the ICP algorithm performs rigid registration in an iterative fashion. The method repeats the following two steps until the least squares error becomes smaller than some threshold:

1. find a point correspondence by using the nearest neighbour rule;

2. estimate a rotation and translation by solving (5.16) using a SVD.

In the original version of ICP, the matching in step 1 is determined by the nearest neighbour criteria. For a point in $M_1$, its corresponding point is the closest point from $M_2$ in Euclidean norm. Note that this may not produce a one-to-one matching. Because the point correspondence changes throughout the iterations, it is difficult to establish theoretical guarantees of convergence for the classical ICP algorithm. The use of the least squares residual (5.16) yields a method referred to as point-to-point ICP and is known to be sensitive to noise and outliers (Bellekens et al., 2014).

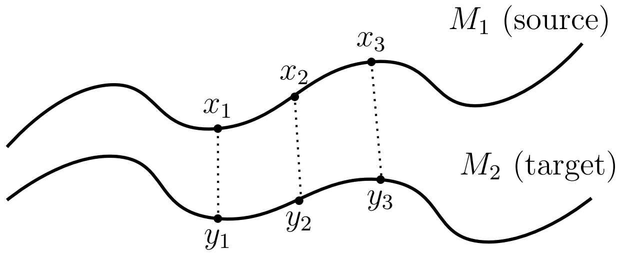


Figure 5.6: Point-to-point ICP.

Numerous adaptations of the original ICP algorithm have been proposed, impacting all stages of the algorithm, which notably include finding the point correspondence and the minimization approach to find the rigid transformation.

The point-to-plane ICP (also called point-to-surface) is one such adaptation (Chen and Medioni, 1992). This modification takes the local neighbourhood of a corresponding target point into account to modify the least squares residual in (5.16) and try to reduce the algorithm's sensitivity to noise. Assuming that the point clouds represent smooth surfaces, it minimizes the distance between a source point and the linear approximation of the point cloud at the corresponding target point. Concretely, in the point-to-plane

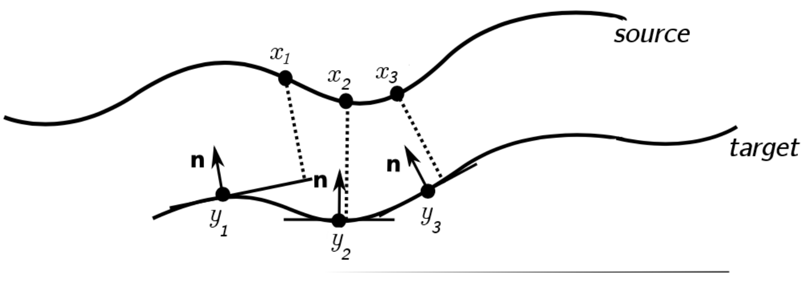


Figure 5.7: Point-to-plane ICP (Bellekens et al., 2014).

ICP the least square problem in step 2 is replaced by

$$(Q^*, a^*) = \underset{Q \in \mathrm{O}(n), a \in \mathbb{R}^n}{\arg\min} \sum_{k=1}^{s} |\langle y_k - Qx_k - a, \mathbf{n}_k \rangle|^2, \tag{5.17}$$

where $y_k$ is the corresponding point to $x_k$ and $\mathbf{n}_k$ is the normal vector to the linear approximation of the point cloud at $y_k$. This point-to-plane error metric does not have a closed-form solution and is minimized using nonlinear least squares methods. Although each iteration of the point-to-plane ICP is typically slower than an iteration of the point-to-point ICP, the former displays a faster convergence rate in practice (Rusinkiewicz and Levoy, 2001), as is predicted by theoretical analysis (Pottmann et al., 2004). An approximation as a linear least squares for the point-to-plane ICP was proposed in (Low, 2004). Plane-to-plane error metrics have also been proposed, which minimize the distance between the two tangent approximations around corresponding points. Generalized ICP allows to leverage the covariance matrix of the point clouds (Segal et al., 2009). Variants of ICP are summarized in (Rusinkiewicz and Levoy, 2001).

In the non-rigid registration paradigm, the transformation is no longer restricted to the combination of a rotation and a translation. There exists a wide range of approaches for the non-rigid registration which we do not cover in depth, as this chapter only considers applications of rigid registration. Some of the popular approaches for non-rigid registration include robust point matching (Gold et al., 1998); thin plate spline robust point matching (Chui and Rangarajan, 2003); kernel correlation approach (Tsin and Kanade, 2004) and coherent point drift (Myronenko and Song, 2010).

## Our contribution and outline of the section

We attempt to find a rigid transformation between two point clouds, akin to (5.15) and assume that the target and source point clouds belong to the same algebraic variety (up to a change in coordinates). Our method does not assume that a point correspondence is given, nor does it attempt to compute one. This avoids the convergence issues that stem from the iterative estimation of a point correspondence. The transformation we compute ensures that the image of the source samples by the rigid transformation belongs to the algebraic variety of the target point cloud. This, conceptually, aligns the algebraic varieties of the source and target point clouds.

In order to reduce the sensitivity to noise and outliers, variants of the ICP algorithm minimize a point-to-plane or plane-to-plane metric using local linear approximations of the point clouds. The approach described in this chapter can be considered as a *point-to-algebraic variety matching.* Our residual does not measure the Euclidean distance between a point and its corresponding point or the tangent plane at a corresponding point. Rather, our residual is computed in the feature space, and, intuitively, measures the distance between a source point and the algebraic variety that defines the target point cloud. This also allows us to consider two point clouds of different sizes without discarding samples, and seamlessly handle cases where the source point cloud only overlaps partially with the target point cloud. These cases are difficult to handle for conventional, point-to-point methods.

Additionally, we present the more realistic setting of noisy samples, which only satisfy the algebraic variety model approximately. The variety denoising technique developed in Section 5.1 is used as a preprocessing step, to identify an approximation of each point cloud by an algebraic variety, which is then used to compute the rigid transformation.

For the noisy and noiseless case, we show numerical results on synthetic examples which illustrate the efficiency and accuracy of our approach. We finally show that this registration technique is applicable to complex data such as medical three-dimensional scans.

### 5.2.1 Problem set up

Consider two algebraic varieties $V_1, V_2 \subset \mathbb{R}^n$ of the same degree $d$. We assume the existence of a rigid transformation $\mathcal{T} \colon \mathbb{R}^n \to \mathbb{R}^n$ that makes the varieties $V_1$ and $V_2$ overlap.

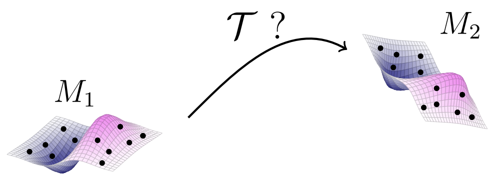
**A11.** *There exists $Q \in \mathrm{SO}(n)$ and $a \in \mathbb{R}^n$ such that for all $x_1 \in V_1$, we have*

$$\mathcal{T}(x_1) := Qx_1 + a \in V_2, \tag{5.18}$$

*where* $\mathrm{SO}(n) = \left\{Q \in \mathbb{R}^{n\times n} : Q^\top Q = \mathrm{I}_n, \det(Q) = 1\right\}$ *denotes the* special orthogonal group.

Intuitively, $V_1$ and $V_2$ represent the same shape but in different coordinate systems. To adhere to the terminology of registration, a matrix $X \in \mathbb{R}^{n\times s}$ in this section represents a *point cloud* in $\mathbb{R}^n$. Each column of $X$ is called a *sample* or *data point*. Let $M_1 \in \mathbb{R}^{n\times s_1}$ and $M_2 \in \mathbb{R}^{n\times s_2}$ be composed of respectively $s_1$ samples in $V_1$ and $s_2$ samples in $V_2$. Given $M_1$ and $M_2$, our task is to estimate $Q \in \mathrm{SO}(n)$ and $a \in \mathbb{R}^n$ which define the transformation $\mathcal{T}$. The special orthogonal group $\mathrm{SO}(n)$, which consists of the set of rotations in $\mathbb{R}^n$, can be equipped with a Riemannian manifold structure which is outlined in Section 2.9.4. If the transformation includes a reflection in addition to a rotation, the optimization should be done over the connected component of $\mathrm{O}(n)$ with matrices of determinant $-1$, instead of $\mathrm{SO}(n)$.

The samples of $M_1 \in \mathbb{R}^{n\times s_1}$ and $M_2 \in \mathbb{R}^{n\times s_2}$ are generated independently and it may be that the point clouds that we wish to overlap have a different number of samples, i.e., $s_1 \neq s_2$. The goal is therefore not to establish a point-to-point matching, but to ensure that the varieties $V_1$ and $V_2$ — observed through the available samples in $M_1$ and $M_2$ — overlap as best as possible.



### 5.2.2 Registration as an optimization problem

Assume that the algebraic varieties $V_1, V_2$ have degree at most $d$ and are described by $N-r$ linearly independent polynomial equations, for some value $r < N$. Also assume that $\min(s_1, s_2) \geq N(n,d)$, that is, each matrix contains more samples than the dimension of the feature space. According to (5.7), since every column of $M_2$ belongs to $V_2$, there exists a subspace $\mathcal{U}_{M_2} \in \text{Grass}(N,r)$ such that

$$\left\|\Phi_d(M_2) - \mathrm{P}_{\mathcal{U}_{M_2}}\Phi_d(M_2)\right\|_{\mathrm{F}}^2 = 0,$$

and, additionally, the variety $V_2$ is defined by polynomial equations of coefficients $U_{M_2}^{\perp} \in \text{St}(N, N-r)$ for any $U_{M_2}^{\perp}$ such that $\text{range}(U_{M_2}^{\perp}) = \mathcal{U}_{M_2}^{\perp}$. Thus, we can write $V_2 = V_{\mathcal{U}_{M_2}^{\perp}}$ in the manner of (5.7), where $V_{\mathcal{U}_{M_2}^{\perp}}$ is the variety defined by equations of coefficients $U_{M_2}^{\perp}$. Denote the columns of $M_1$ as $m_1^{(1)}, m_2^{(1)}, \cdots m_{s_1}^{(1)}$. We wish to find a rotation $Q \in \text{SO}(n)$ and a vector $a \in \mathbb{R}^n$ such that the points $\mathcal{T}\left(m_k^{(1)}\right) = Qm_k^{(1)} + a$ belong to $V_2$ for all $k$. This is satisfied whenever

$$\sum_{k=1}^{s_1} \left\|\varphi_d(Qm_k^{(1)} + a) - \mathrm{P}_{\mathcal{U}_{M_2}}\varphi_d(Qm_k^{(1)} + a)\right\|_2^2 = 0,$$

where $\varphi_d$ is the monomial features (Equation (3.15)). The distinction with the least squares minimization (5.16) appears clearly. In this case, we do not minimize a point-to-point distance; instead we want each sample of $M_1$ to belong to $V_2$ after application of the transformation. This is equivalently written in matrix notation with the monomial features applied column-wise:

$$\left\|\Phi_d(QM_1 + a\mathbf{1}_{1\times s}) - \mathrm{P}_{\mathcal{U}_{M_2}}\Phi_d(QM_1 + a\mathbf{1}_{1\times s_1})\right\|_{\mathrm{F}}^2 = 0,$$

where $\mathbf{1}_{1\times s_1}$ is a row vector of size $s_1$ that is full of ones. This leads to the following two-step strategy to identify the transformation between $V_1$ and $V_2$.

**Step 1: Algebraic variety identification** First, we identify the equations of the algebraic variety $V_2$ that the columns of $M_2$ lie on:

$$\mathcal{U}_{M_2} = \begin{cases} \min\limits_{\mathcal{U}} & \|\Phi_d(M_2) - \mathrm{P}_{\mathcal{U}}\Phi_d(M_2)\|_{\mathrm{F}}^2 \\ \text{s.t.} & \mathcal{U} \in \mathrm{Grass}(N, r), \end{cases} \tag{5.19}$$

where $\mathrm{Grass}(N, r)$ denotes the Grassmann manifold, defined in Section 2.9.5, and $\mathrm{P}_{\mathcal{U}}$ is the orthogonal projection on the subspace $\mathcal{U}$. Note that $\mathcal{U}_{M_2}$, the global minimizer of (5.19), is given by the $r$ leading singular vectors of $\Phi_d(M_2)$.

**Step 2: Registration** We are looking for a transformation defined by $Q$ and $a$ so that the points $Qm_k^{(1)} + a$ belongs to the algebraic variety $V_2$. This is obtained by,

$$(Q^*, a^*) = \begin{cases} \min\limits_{Q,a} & \left\|\Phi_d(QM_1 + a\mathbf{1}_{1\times s_1}) - \mathrm{P}_{\mathcal{U}_{M_2}}\Phi_d(QM_1 + a\mathbf{1}_{1\times s_1})\right\|_{\mathrm{F}}^2 \\ \text{s.t.} & Q \in \mathrm{SO}(n) \\ & a \in \mathbb{R}^n. \end{cases} \tag{5.20}$$

**Remark 5.1.** *Another possibility would be to compute both steps at once,*

$$\begin{cases} \min\limits_{\mathcal{U},Q,a} & \|\Phi_d([M_2, QM_1 + a\mathbf{1}_{1\times s_1}]) - \mathrm{P}_{\mathcal{U}}\Phi_d([M_2, QM_1 + a\mathbf{1}_{1\times s_1}])\|_{\mathrm{F}}^2 \\ s.t. & Q \in \mathrm{SO}(n) \\ & \mathcal{U} \in \mathrm{Grass}(N, r) \\ & a \in \mathbb{R}^n. \end{cases} \tag{5.21}$$

*where $[\cdot,\cdot]$ concatenates two matrices column-wise. We found that this optimization problem is harder to solve in practice. Performing steps 1 and 2 separately is suitable because we are able to estimate the subspace $\mathcal{U}$, which defines the underlying algebraic variety, from the samples of only one of the point clouds. Hence, problem* (5.21) *can be genuinely decomposed in two steps.*

**Remark 5.2.** *Interestingly, the two-step approach is asymmetrical, in that $M_1$ and $M_2$ play the roles of the source and target point clouds, respectively. In situations where one point cloud represents an object and the second point cloud only overlaps with a small part of that object, it is preferable to use the larger point cloud as the target (labelled $M_2$) and the smaller one as the source (labelled $M_1$).*

**Remark 5.3.** *Note that, for the registration problem* (5.20)*, it is not possible to use the monomial kernel to represent the monomial features. The monomial kernel is rotation invariant, i.e. $\mathrm{K}_d(QX, QX) = \mathrm{K}_d(X, X)$ for any $Q \in \mathrm{SO}(n)$.*

**Riemannian optimization and software**

The (noiseless) registration procedure we propose consists in solving (5.19) and (5.20) in succession. Both of these steps require minimizing a smooth, scalar-valued function on a Riemannian manifold, a subject that is discussed in Chapter 2. The Grassmann manifold $\mathrm{Grass}(N, r)$, the special orthogonal group $\mathrm{SO}(n)$ and the Euclidean space $\mathbb{R}^n$, which appear in Problems (5.19) and (5.20), can be equipped with a Riemannian manifold structure necessary to apply optimization methods. These are outlined in Section 2.9. We use a second-order Riemannian trust-region (Algorithm 2). In this noiseless setting, the point clouds belong exactly to an algebraic variety and Problem (5.19) is solved by a truncated singular value decomposition of the features matrix. For Problem (5.20), we use a random initial rotation $Q \in \mathrm{SO}(n)$ and translation $a \in \mathbb{R}^n$ with $\|a\|_2 = 1$. The stopping criterion is set to trigger when the norm of the Riemannian gradient is below $10^{-6}$.

## 5.2.3 Numerical results for noiseless registration

Our code for the registration problem is available in Python at `https://github.com/flgoyens/variety-registration`. In the numerical examples that follow, the default implementation of RTR from PyManopt was used where the full Riemannian Hessian is computed through automatic differentiation (Townsend et al., 2016). This section illustrates the efficiency of our approach to identify a rigid transformation between two quadratic curves or surfaces using the monomial features of degree $d = 2$. The given point clouds $M_1, M_2$ belong exactly to the varieties and no noise is added at this point. The rank is set to $r = N - 1$, since the points clouds are described by a single polynomial equation. The quality of the solution $(Q^*, a^*)$ returned by the solver is assessed using the following residual:

$$\mathrm{RESIDUAL} = \left\|\Phi_d\left(Q^* M_1 + a^* \mathbf{1}_{1\times s_1}\right) - \mathrm{P}_{\mathcal{U}_{M_2}} \Phi_d\left(Q^* M_1 + a^* \mathbf{1}_{1\times s_1}\right)\right\|_{\mathrm{F}}^2,$$

which can be interpreted using (5.8). The residual measures how accurately the image by the rigid transformation of the samples in $M_1$ satisfies the polynomial equations of the algebraic variety identified from the target point cloud $M_2$. We emphasize that the measure $\|Q^* M_1 + a^* \mathbf{1}_{1\times s_1} - M_2\|_{\mathrm{F}}^2$ is meaningless. We are not trying to match data points from different point clouds together, we aim for the underlying algebraic varieties to overlap. Additionally, this measure is undefined for point clouds of different sizes. In Figures 5.8 and 5.9, the upper image shows the two point clouds of input, and the lower image shoes the output of the algorithm.

**Example 4** In Figure 5.8, the initial point clouds have dimensions $M_1 \in \mathbb{R}^{2\times 20}$ and $M_2 \in \mathbb{R}^{2\times 30}$. Note that they have different sizes. The samples in $M_2$ are randomly generated as points that satisfy the quadratic $x \mapsto x^2$. The samples in $M_1$ originate from a rotated and translated version of that curve, for some randomly generated rotation and translation which we attempt to recover. The algorithm's output has RESIDUAL $\approx 10^{-7}$.

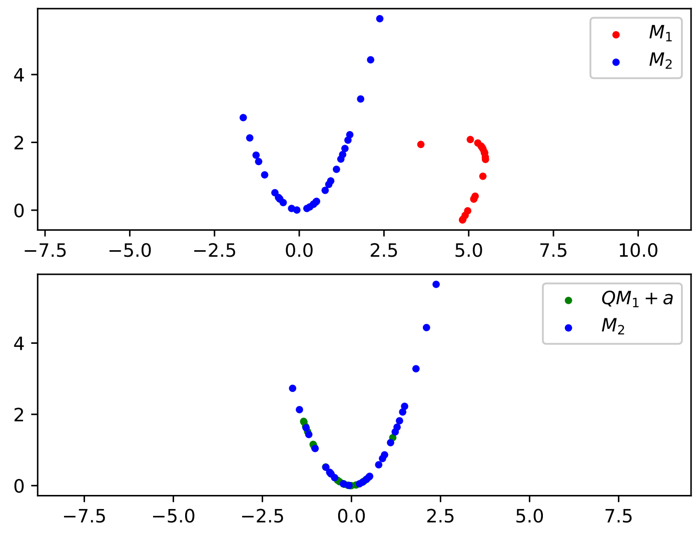


Figure 5.8: Registration for a quadratic curve in $\mathbb{R}^2$.

**Example 5** In Figure 5.9, the data is on a quadratic surface in $\mathbb{R}^3$. The input data has dimensions $M_1 \in \mathbb{R}^{3\times 200}$, $M_2 \in \mathbb{R}^{3\times 200}$. The result satisfies RESIDUAL $\approx 10^{-7}$.

These numerical results illustrate that our approach successfully finds an accurate rigid transformation in these simple cases. Due to the nonconvexity of $\mathrm{SO}(n)$ and the random initialization of the algorithm, the solver sometimes fails to find the correct transformation and appears to find a local minimum. We then restart the registration problem (5.20) with a new random initial guess, and report that this usually finds the global minimum under five attempts. This is likely helped by the fact that registration problems are most often considered in $\mathbb{R}^2$ or $\mathbb{R}^3$, and the dimension of the search space is small.

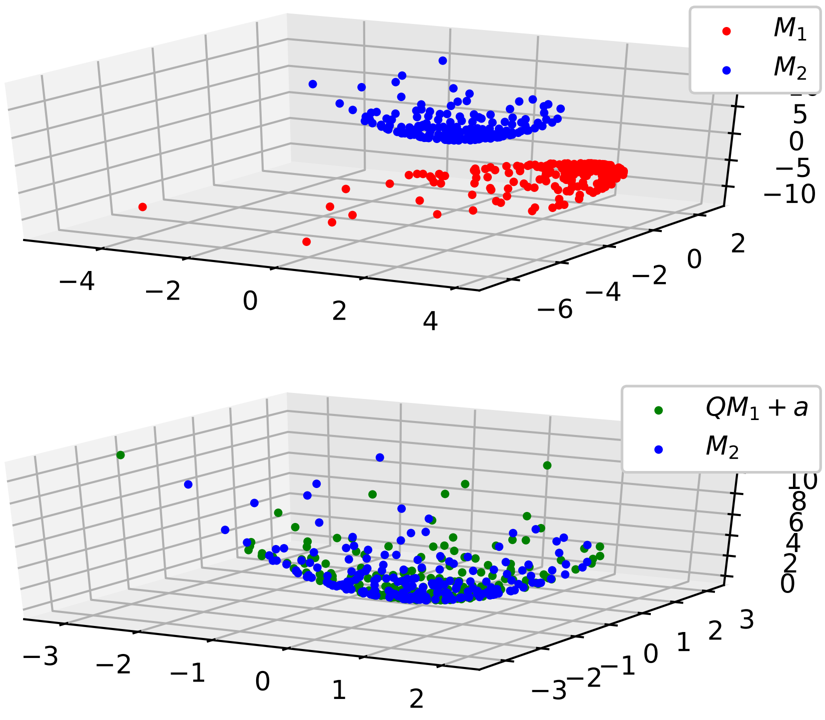


Figure 5.9: Registration for a quadratic surface in $\mathbb{R}^3$.

### 5.2.4 Noisy registration

We consider the more practical setting where the samples available are corrupted by noise, that is, they are close to an algebraic variety, but may not belong to one exactly. As in the denoising problem, the noise may be interpreted as the mismatch between a practical data set and the algebraic variety model. Let $M_1$, $M_2$ belong to the same algebraic variety (up to a change of coordinates) as in the previous section. We are now given two point clouds $\hat{M}_1$ and $\hat{M}_2$, which are noisy versions of $M_1$ and $M_2$:

$$\hat{M}_1 = M_1 + \omega_1$$
$$\hat{M}_2 = M_2 + \omega_2,$$

where $\omega_1, \omega_2$ are noisy perturbations following the same distribution, with $\|\omega_1\|_{\mathrm{F}} \ll \|M_1\|_{\mathrm{F}}$ and $\|\omega_2\|_{\mathrm{F}} \ll \|M_2\|_{\mathrm{F}}$. Given $\hat{M}_1$ and $\hat{M}_2$, the goal is to find a rigid transformation $\mathcal{T}\colon \mathbb{R}^n \to \mathbb{R}^n$ such that $\mathcal{T}(M_1)$ and $M_2$ overlap. To achieve this, we first attempt to recover $M_1$ and $M_2$ from $\hat{M}_1$ and $\hat{M}_2$, by solving the denoising problem (5.5), in the manner of

Section 5.1. This yields estimates $X_1$ and $X_2$, along with subspaces $\mathcal{U}_1$ and $\mathcal{U}_2$ which define the algebraic varieties that $X_1$ and $X_2$ (approximately) belong to. The second step estimates a transformation $\mathcal{T}$ such that $\mathcal{T}(X_1)$ belongs to the algebraic variety identified for $X_2$.

**Step 1: Algebraic variety identification and denoising**

$$(X_1,\mathcal{U}_1) := \begin{cases} \underset{X,\mathcal{U}}{\arg\min} & \|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}}^2 + \lambda \left\|X - \hat{M}_1\right\|_{\mathrm{F}}^2 \\ & \mathcal{U} \in \mathrm{Grass}(N, r). \end{cases} \tag{5.22}$$

$$(X_2,\mathcal{U}_2) := \begin{cases} \underset{X,\mathcal{U}}{\arg\min} & \|\Phi_d(X) - \mathrm{P}_{\mathcal{U}}\Phi_d(X)\|_{\mathrm{F}}^2 + \lambda \left\|X - \hat{M}_2\right\|_{\mathrm{F}}^2 \\ & \mathcal{U} \in \mathrm{Grass}(N, r). \end{cases} \tag{5.23}$$

**Step 2: Registration**

$$(Q^*, a^*) := \begin{cases} \underset{Q,a}{\min} & \|\Phi_d(QX_1 + a\mathbf{1}_{1\times s_1}) - \mathrm{P}_{\mathcal{U}_2}\Phi_d(QX_1 + a\mathbf{1}_{1\times s_1})\|_{\mathrm{F}}^2 \\ & Q \in \mathrm{SO}(n) \\ & a \in \mathbb{R}^n. \end{cases} \tag{5.24}$$

Steps 1 and 2 are solved with the Riemannian trust-region method (Algorithm 2), with the same settings as in the noiseless case. The notable difference with the noiseless case is that calculating the SVD of $\Phi_d(\hat{M}_i)$ does not give the global minimizer of Step 1. The noisy case requires to solve Problem (5.5), an algebraic variety denoising problem. Recall from Section 5.1 that the nonconvexity of (5.22) and (5.23) does not seem to cause problems in practice for the denoising. Indeed, very good initial guesses are available for the variables $X$ and $\mathcal{U}$, in $\hat{M}_i$ and the $r$ first singular vectors of $\Phi_d(\hat{M}_i)$ respectively. For the registration step (5.24), we do not assume any prior information on the transformation $\mathcal{T}$ and initialize with a random rotation and random vector $a$ of unit norm.

### 5.2.5 Numerical results for noisy registration

We demonstrate the efficiency of our registration approach on a dental scan and synthetic test problems which do not satisfy the algebraic variety model exactly. We consider examples on the quadratic curve $x \mapsto x^2$ in $\mathbb{R}^2$ as these examples are easier to visualize than in $\mathbb{R}^3$. Matrices $M_1$ and $M_2$ are generated as in the noiseless case, as point clouds that belong to the quadratic curve. We then add random matrices of Gaussian noise $\omega_1, \omega_2 \sim \mathcal{N}(0, \sigma^2)$ which yields matrices $\hat{M}_1$ and $\hat{M}_2$. In the noisy setting, the residual is now measured using the denoised point clouds, labelled $X_1$ and $X_2$:

$$\mathrm{RESIDUAL} = \|\Phi_d(Q^*X_1 + a^*\mathbf{1}_{1\times s_1}) - \mathrm{P}_{\mathcal{U}_2}\Phi_d(Q^*X_1 + a^*\mathbf{1}_{1\times s_1})\|_{\mathrm{F}}^2 .$$

**Example 6** In Figure 5.10, the standard deviation of the noise is $\sigma = 5 \cdot 10^{-2}$ and the algorithm's output satisfies RESIDUAL $\approx 10^{-4}$. As usual, the top figure shows the initial noisy data and the bottom figure shows the algorithm's output, after denoising and computation of the registration. We see that despite the presence of noise in the original data, the transformation is well estimated to create an overlap.

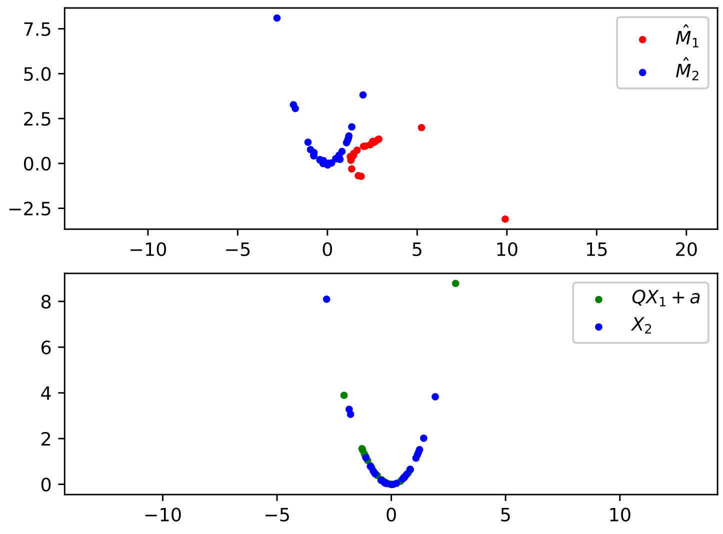


Figure 5.10: Noisy registration with $\sigma = 5 \cdot 10^{-2}$

**Example 7** In Figure 5.11, the magnitude of the noise is increased to $\sigma = 10^{-1}$ and the output satisfies RESIDUAL $\approx 10^{-2}$. The transformation is again estimated correctly, with the residual increasing proportionally to the magnitude of the noise. We now show examples where there is a partial overlap between the source and target point clouds.

**Example 8: partial overlap** In 5.12 and 5.13, the point cloud $\hat{M}_1$ only overlaps with a small part of $\hat{M}_2$. The noise levels are of $\sigma = 10^{-2}$ and $\sigma = 10^{-1}$ and give RESIDUAL values of $10^{-4}$ and $10^{-2}$, respectively.

**Example 9: no overlap** In Figure 5.14, we push things even further and show that the registration may be possible even in cases where there is no overlap between the point clouds. The registration is possible provided that the samples belong to a common algebraic variety. The noise level is set to $\sigma = 10^{-2}$ and RESIDUAL $\approx 10^{-4}$.

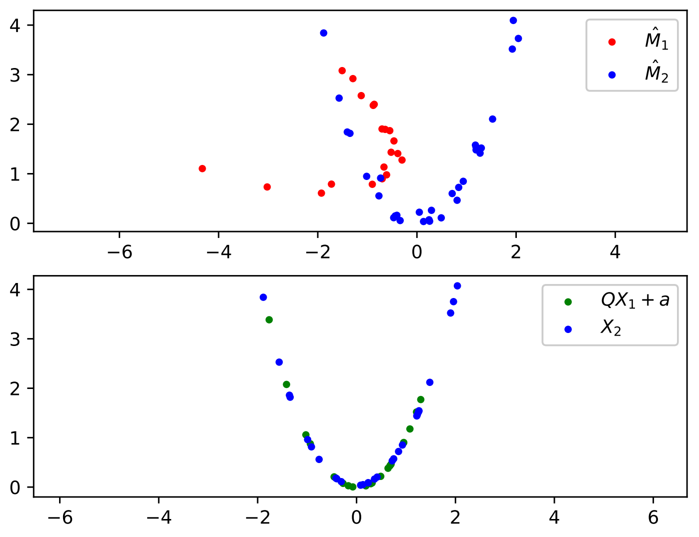


Figure 5.11: Noisy registration with $\sigma = 10^{-1}$

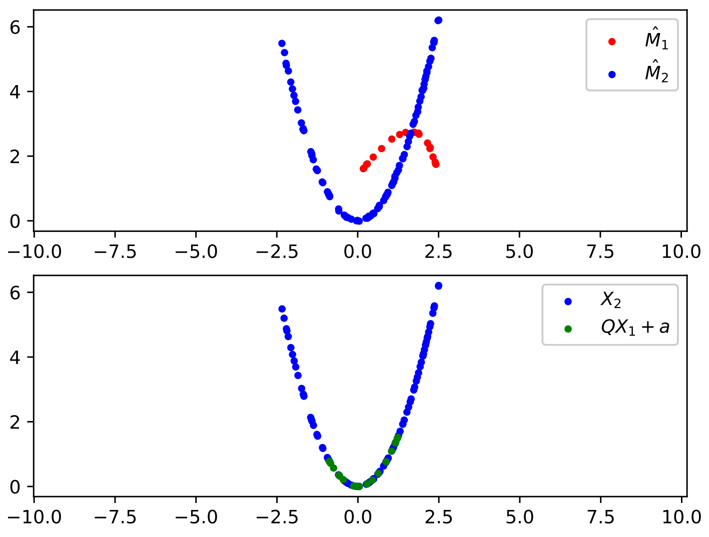


Figure 5.12: Noisy registration with $\sigma = 10^{-2}$ and partial overlap

**Example 10: registration of dental scan** Let us now consider a practical data set, a computed tomography (CT) dental scan of dimension $3 \times 2048$. The data is naturally noisy, as the scan is not precisely defined by an algebraic variety. Nevertheless, we see that the approximation by a variety allows us to estimate with some level of accuracy the rigid transformation that was generated between two identical versions of the scan. We use a degree $d = 2$ for the monomial features which yields the best result. The output of

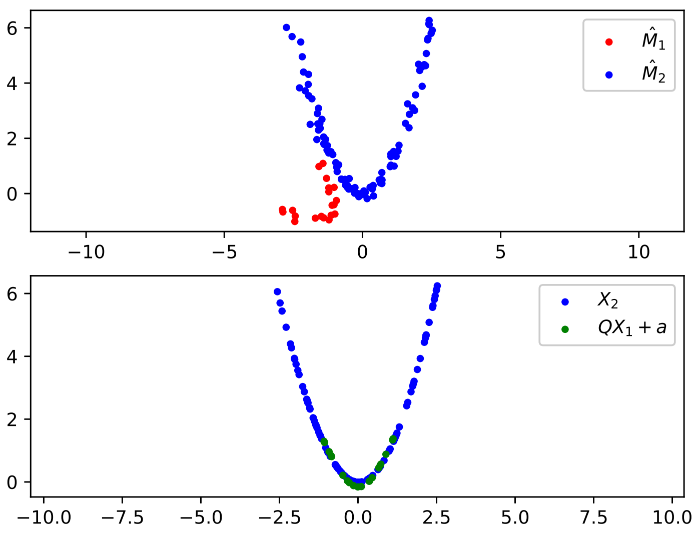


Figure 5.13: Noisy registration with $\sigma = 10^{-1}$ and partial overlap

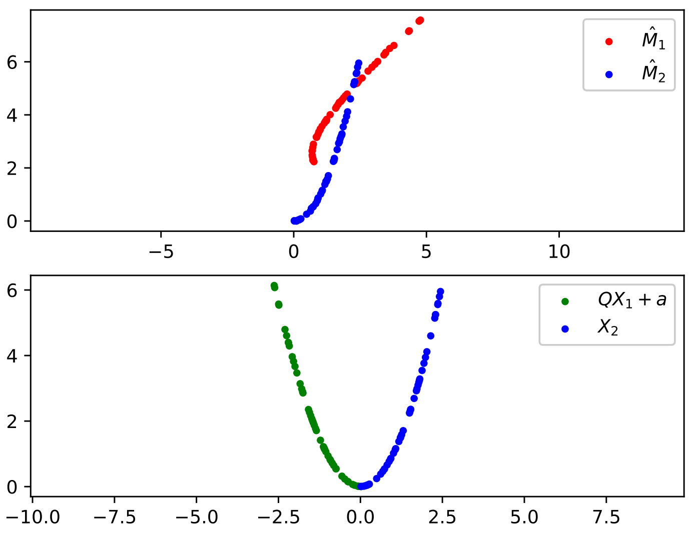


Figure 5.14: Noisy registration with $\sigma = 10^{-2}$ and no overlap

the solver gives RESIDUAL $\approx 10^{-1}$.

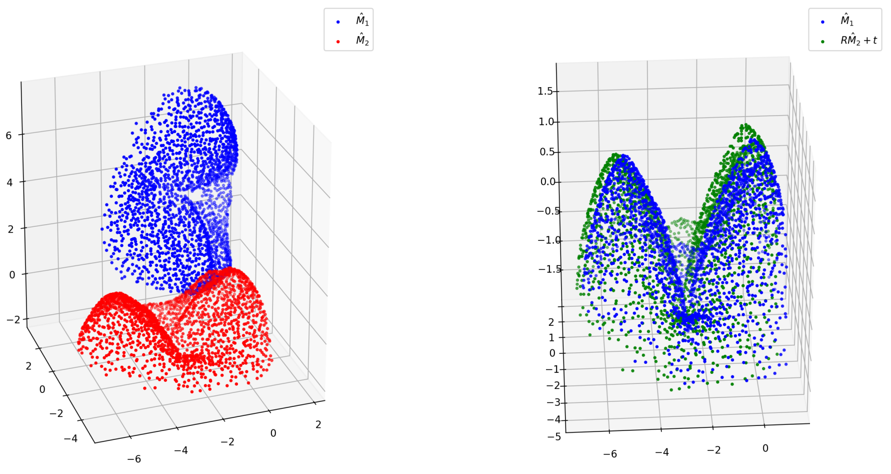


Figure 5.15: Registration for dental scan.

## 5.3 Conclusions

In this chapter, we present a framework which leverages the relationship between algebraic varieties and the monomial map for the approximation and denoising of point clouds. The features approach offers a conceptually appealing way to perform denoising on algebraic varieties and numerical results show great accuracy in the recovery for various noise levels, with a theoretical estimate of the accuracy using Stein's unbiased risk estimate. The main characteristics of the approach are that the approximation is global and it is able to identify the polynomial equations that define the algebraic variety from a noisy set of points on the algebraic variety. The use of second-order Riemannian optimization methods allows to achieve high accuracy.

The algebraic variety model is then further exploited for the registration problem, where we show numerical evidence that we are able to overlap point clouds which are approximated by algebraic varieties. We observe some robustness to noise and the method performs even on complex data such as dental scans. One advantage of our approach over common algorithms for rigid registration is that we do not assume exact point matching and our method achieves good results even in cases of partial overlap between the two point clouds. The main limitation is that our method may behave poorly on point clouds which cannot be approximated by an algebraic variety.

For both problems, future research directions include the improvement of the scalability of the algorithm with respect to the number of data points $s$, as well as the use of polynomial features of higher degree. A current limitation of our implementation is that it works best when using polynomial features of degree 2, even for complex data sets such as dental scans. As explained on page 56, the dimension of the feature space $N(n, d)$ as well as the number of data points necessary increases exponentially with the degree $d$ of the polynomial features. The exponential increase in the dimension with $d$ makes the problems much harder to solve for increasing degrees as large as 3 or 4. Despite this fact, it would be worthwhile to develop tools to tackle problems of larger scale and reach better results for a degree higher than two, which likely also requires to use a different polynomial basis than the monomials. This should improve the conditioning of the features matrix and avoid the notorious ill-conditioning of Vandermonde matrices. The dimension of the feature space could also potentially be reduced using low-dimensional representations of the features or the kernel, in the spirit of (Rahimi et al., 2007).

The scalability of the algorithm with respect to the number of data points $s$ may also be improved. The cost functions in (5.5) and (5.24) can be written as a finite sum over the columns of the matrices involved. This allows to use stochastic variants of classical optimization methods, where the derivatives are approximated by subsampling a batch of terms in the sum at each iteration, in the spirit of (Kohler and Lucchi, 2017; Xu et al., 2020). This would allow to process a larger number of data points.

# Chapter 6

# Equality constrained optimization

Working over a Euclidean space $\mathcal{E}$ with inner product $\langle \cdot, \cdot \rangle$ and associated norm $\|\cdot\|$, we consider the constrained optimization problem

$$\min_x f(x) \text{ subject to } h(x) = 0, \tag{P}$$

where $f \colon \mathcal{E} \to \mathbb{R}$ and $h \colon \mathcal{E} \to \mathbb{R}^m$ are smooth. The feasible set is denoted by

$$\mathcal{M} = \{x \in \mathcal{E} : h(x) = 0\}.$$

The analysis of global complexity for equality constrained problems has attracted a lot of attention in recent years. Indeed, it is desirable to establish the worst-case number of iterations that an algorithm requires to achieve approximate criticality. Usually, approximate first- or second-order critical points are sought. In the presence of equality constraints, various definitions of approximate second-order critical points are present in the literature, which makes the comparison of different methods difficult. In this chapter, we propose a definition of approximate second-order criticality for (P), which attempts to unify these different definitions. Our condition is based on optimality conditions for optimization on Riemannian manifolds, defined in Section 2.5. It has a natural geometric interpretation using the Riemannian Hessian of the cost function on a manifold near the feasible set $\mathcal{M}$. Our main contribution, which is mostly of a theoretical nature, is an algorithm which computes such approximate second-order critical points with optimal worst-case iteration complexity for an arbitrary initialization. This rate is sharp, meaning that no second-order method can achieve a better performance in the worst case. This is achieved by minimizing a smooth penalty function known as Fletcher's augmented Lagrangian, introduced in (Fletcher, 1970).

In Section 6.2, we review recent works with complexity analysis for constrained problems and discuss how our results and assumptions differ from the existing literature. We devote particular attention to various notions of approximate second-order criticality that have been used in recent works for constrained problems. Section 6.3 establishes properties of Fletcher's augmented Lagrangian, whose relevance goes beyond the study of worst-case complexities. Most notably, we establish a correspondence between approximate minimizers of Fletcher's augmented Lagrangian and approximate minimizers of (P). We present our algorithm in Section 6.4, which features first- and second-order versions. Section 6.5 shows that a key assumption made early in the chapter can be lifted. Namely, it is possible to estimate a suitable value for the penalty parameter while retaining the global convergence rates of the original algorithm (up to a logarithmic factor). Section 6.1 outlines the main components of our approach and the assumptions we make on problem (P). We explain that, under regularity assumptions, our approach builds on Riemannian optimization concepts. Most notably, the Riemannian viewpoint gives meaning to our approximate criticality conditions.

## 6.1 Introduction

### 6.1.1 Assumptions

We introduce three central assumptions about the set $\mathcal{M}$. Define the set

$$\mathcal{D} = \{x \in \mathcal{E} \colon \operatorname{rank}(\mathrm{D}h(x)) = m\}. \tag{6.1}$$

It is known that if $\mathcal{M}$ is included in $\mathcal{D}$ then $\mathcal{M}$ is a (smooth) embedded submanifold of $\mathcal{E}$ (Absil et al., 2008). We further assume that there is a region around $\mathcal{M}$ where the differential of the constraints is nonsingular. Let $\|\cdot\|$ denote the usual norm on $\mathbb{R}^m$.

**A12.** *There exist constants $R, \underline{\sigma} > 0$ such that for all $x$ in the set*

$$\mathcal{C} = \{x \in \mathcal{E} : \|h(x)\| \leq R\} \tag{6.2}$$

*we have $\sigma_{\min}(\mathrm{D}h(x)) = \sigma_m(\mathrm{D}h(x)) \geq \underline{\sigma} > 0$ where $\sigma_k(A)$ and $\sigma_{\min}(A)$ denote the kth and the smallest singular value of a linear map $A$, respectively. In particular, we have $\operatorname{rank}(\mathrm{D}h(x)) = m$.*

**A13.** *The sets $\mathcal{M} = \{x \in \mathcal{E} : h(x) = 0\}$ and $\mathcal{C} = \{x \in \mathcal{E} : \|h(x)\| \leq R\}$ are compact.*

**A14.** *There exists a constant $C_h > 0$ such that, for all $x \in \mathcal{C}$ and $v \in \mathcal{E}$,*

$$h(x+v) = h(x) + \mathrm{D}h(x)[v] + E(x,v)$$

*with $\|E(x,v)\| \leq C_h\|v\|^2$.*

The set $\mathcal{C}$ is our region of interest. If Algorithm 7 enters that region, then it stays in it. Accordingly, we make the following assumption.

**A15.** *The initial iterate $x_0$ belongs to $\mathcal{C}$.*

Note that affine constraints do not satisfy A13 and A14, but these constraints are usually not problematic as there are various effective approaches to handle them, including feasible methods. The two following examples show how to compute the constants $R$ and $\underline{\sigma}$, which define the region of interest $\mathcal{C}$, for the Stiefel manifold (see Section 2.9.3) and for a convex quadratic constraint. The derivations can be found in Appendix A.2.

**Example 6.1** (The Stiefel manifold). *Let $\mathcal{E} = \mathbb{R}^{n\times p}$, the Stiefel manifold is defined as*

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

*The manifold corresponds to the defining function $h\colon \mathbb{R}^{n\times p} \to \mathrm{Sym}(p)\colon X \mapsto h(X) = X^\top X - \mathrm{I}_p$, where $\mathrm{Sym}(p)$ is the set of symmetric matrices of size $p$. For any $R < 1$, all $X \in \mathbb{R}^{n\times p}$ such that $\|h(X)\| \leq R$, satisfy $\sigma_{\min}(\mathrm{D}h(X)) \geq 2\sigma_{\min}(X) \geq 2\sqrt{1-R}$. Therefore, A12 is satisfied for any $R < 1$ and $\underline{\sigma} \leq 2\sqrt{1-R}$.*

**Example 6.2** (Convex quadratic constraint). *Let $\mathcal{E} = \mathbb{R}^n$ and consider the set $\mathcal{M}$ for $h(x) = x^\top Ax + b^\top x + c$, where $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. For any $R < |h(x^*)|$, where $x^*$ is the minimizer of the quadratic, all $x \in \mathbb{R}^n$ satisfy $\sigma_{\min}(\mathrm{D}h(x)) > 0$. Additionally, $\underline{\sigma} = \min_{x\in\mathcal{C}} \sigma_{\min}(\mathrm{D}h(x)) > 0$ by compactness of $\mathcal{C}$.*

### 6.1.2 Layered manifolds

Assumption A12 allows us to characterize any point in $\mathcal{C}$ as belonging to some Riemannian submanifold. This manifold is defined by a level set of the function $h$, while the feasible set $\mathcal{M}$ is the 0-set of $h$. This observation partitions the region of interest $\mathcal{C}$ into Riemannian submanifolds which we call *layered manifolds.* These layered manifolds help to derive meaningful criticality conditions for points which are nearly feasible.

**Proposition 6.1** (Layered manifolds). *Under A12, for any $x \in \mathcal{C}$, the set $\mathcal{M}_x = \{y \in \mathcal{E} : h(y) = h(x)\}$ is a submanifold of $\mathcal{E}$ contained in $\mathcal{C}$. The tangent space and the normal space of $\mathcal{M}_x$ at $y \in \mathcal{M}_x$ are given respectively by:*

$$\mathrm{T}_y\mathcal{M}_x = \{v \in \mathcal{E} : \mathrm{D}h(y)[v] = 0\} \qquad \textit{and} \qquad \mathrm{N}_y\mathcal{M}_x = \mathrm{span}(\mathrm{D}h(y)^*)\,,$$

*where a star indicates an adjoint.*

*Proof.* The set $\mathcal{M} = \{y \in \mathcal{E} : h(y) = 0\}$ is an embedded submanifold of $\mathcal{E}$ if $\mathrm{rank}(\mathrm{D}h(y)) = m$ for all $y \in \mathcal{M}$ (Absil et al., 2008, Prop. 3.3.3). For some $x \in \mathcal{C}$, one readily checks that the set $\mathcal{M}_x = \{y \in \mathcal{E} : \tilde{h}(y) := h(y) - h(x) = 0\}$ is also an embedded submanifold of $\mathcal{E}$ owing to A12. For all $y \in \mathcal{M}_x$, since $y \in \mathcal{C}$, $\mathrm{D}h(y)$ has full rank $m$ and so does $\mathrm{D}(\tilde{h}(y))$.

This implies that $\mathcal{M}_x$ is an embedded submanifold of $\mathcal{E}$ and the tangent spaces at $y \in \mathcal{M}_x$ is given by $\mathrm{T}_y\mathcal{M}_x = \{v \in \mathcal{E} : \mathrm{D}h(y)[v] = 0\}$. At any $y \in \mathcal{M}_x$, the normal space is the orthogonal complement of the tangent space with respect to the inner product on $\mathcal{E}$, that is, $\mathrm{N}_y\mathcal{M}_x = (\ker \mathrm{D}h(y))^\perp = \mathrm{span}(\mathrm{D}h(y)^*)$. $\square$

The embedded submanifold $\mathcal{M}_x$ for some $x \in \mathcal{C}$ is turned into a Riemannian submanifold using the Euclidean inner product of $\mathcal{E}$ restricted to the tangent spaces of $\mathcal{M}_x$. We proceed to compute the Riemannian gradient and Riemannian Hessian of $f$ on a layer manifold. First, we define the function $\lambda\colon \mathcal{E} \to \mathbb{R}^m$ as follows:

$$\lambda(x) = (\mathrm{D}h(x)^*)^\dagger[\nabla f(x)], \tag{6.3}$$

where a dagger indicates a Moore–Penrose pseudo-inverse. This function is particularly relevant at points $x$ such that $\mathrm{D}h(x)$ has (full) rank $m$. If $x \in \mathcal{E}$ satisfies $\operatorname{rank} \mathrm{D}h(x) = m$, then the orthogonal projector from $\mathcal{E}$ to the tangent space $\mathrm{T}_x\mathcal{M}_x = \ker \mathrm{D}h(x)$ is given in explicit form by

$$\mathrm{Proj}_x(v) = v - \mathrm{D}h(x)^*[z] \qquad \text{with} \qquad z = (\mathrm{D}h(x)^*)^\dagger[v].$$

Based on the derivations in Section 2.9.6, we find that the Riemannian gradient of $f$ on $\mathcal{M}_x$ is given by

$$\mathrm{grad}_{\mathcal{M}_x} f(x) = \mathrm{Proj}_x(\nabla f(x)) = \nabla f(x) - \mathrm{D}h(x)^*[\lambda(x)], \tag{6.4}$$

the orthogonal projection of the Euclidean gradient of $f$ to the tangent space $\mathrm{T}_x\mathcal{M}_x$ and the Riemannian Hessian of $f$ on $\mathcal{M}_x$ is given by

$$\mathrm{Hess}_{\mathcal{M}_x} f(x) = \mathrm{Proj}_x \circ \left( \nabla^2 f(x) - \sum_{i=1}^m \lambda_i(x) \nabla^2 h_i(x) \right) \circ \mathrm{Proj}_x, \tag{6.5}$$

a self-adjoint linear operator on $\mathrm{T}_x\mathcal{M}_x$.

For problem (P), the Lagrangian $\mathcal{L}(x, \lambda)\colon \mathcal{E} \times \mathbb{R}^m \to \mathbb{R}$ is defined as

$$\mathcal{L}(x, \lambda) = f(x) - \langle \lambda, h(x) \rangle,$$

where $\lambda \in \mathbb{R}^m$ is called the vector of multipliers. We also recall the augmented Lagrangian $\mathcal{L}_\beta\colon \mathcal{E} \times \mathbb{R}^m \to \mathbb{R}$ for some penalty parameter $\beta \geq 0$,

$$\mathcal{L}_\beta(x, \lambda) = f(x) - \langle \lambda, h(x) \rangle + \beta \|h(x)\|^2.$$

The augmented Lagrangian is a penalty function which has given rise to a number of popular methods for constrained optimization (Bertsekas, 1982). This allows us to introduce Fletcher's augmented Lagrangian, a penalty function which first appeared in (Fletcher, 1970).

**Definition 6.1** (Fletcher's augmented Lagrangian). *Fletcher's augmented Lagrangian, which we write $g$, is defined as*

$$g(x) = \mathcal{L}_\beta(x, \lambda(x)). \tag{6.6}$$

*where $\lambda(x)$ is defined in* (6.3).

We note that the set $\mathcal{D}$ (Equation (6.1)) is open, and it is easy to verify that $\lambda(\cdot)$ is $C^\infty$ on that set. We also note that, under A12, the set $\mathcal{C}$ is included in $\mathcal{D}$. Therefore, $g$ is also smooth on $\mathcal{C}$. Fletcher's augmented Lagrangian is a smooth penalty which depends only on $x$, the primal variable. The multipliers are computed as a function of $x$. In Section 6.3, we derive meaningful relations between approximate minimizers of $g$ and minimizers of (P). Our main complexity result follows from the minimization of the function $g$ inside the set $\mathcal{C}$.

### 6.1.3 Optimality conditions

We now go over exact and approximate criticality conditions for problem (P). First-order critical points of (P) are defined by

$$h(x) = 0 \qquad \text{and} \qquad \mathrm{grad}_{\mathcal{M}} f(x) = 0, \tag{6.7}$$

whereas second-order critical points satisfy

$$h(x) = 0, \qquad \mathrm{grad}_{\mathcal{M}} f(x) = 0, \qquad \text{and} \qquad \mathrm{Hess}_{\mathcal{M}} f(x) \succeq 0. \tag{6.8}$$

**Proposition 6.2.** *Any local minimizer $x \in \mathcal{E}$ of* (P)*, where* $\mathrm{D}h(x)$ *has rank $m$, is a second-order critical point.*

This leads to our approximate optimality conditions for smooth equality constrained problems.

**Definition 6.2.** *The point $x \in \mathcal{D}$ is an $(\varepsilon_0, \varepsilon_1)$-approximate first-order critical point of* (P) *if*

$$\|h(x)\| \le \varepsilon_0 \qquad \textit{and} \qquad \left\|\mathrm{grad}_{\mathcal{M}_x} f(x)\right\| \le \varepsilon_1. \tag{$\varepsilon$-FOCP}$$

**Definition 6.3.** *The point $x \in \mathcal{D}$ is an $(\varepsilon_0, \varepsilon_1, \varepsilon_2)$-approximate second-order critical point of* (P) *if*

$$\|h(x)\| \le \varepsilon_0, \qquad \left\|\mathrm{grad}_{\mathcal{M}_x} f(x)\right\| \le \varepsilon_1 \qquad \textit{and} \qquad \mathrm{Hess}_{\mathcal{M}_x} f(x) \succeq -\varepsilon_2 \,\mathrm{Id}\,. \tag{$\varepsilon$-SOCP}$$

The notions of ($\varepsilon$-FOCP) and ($\varepsilon$-SOCP) have a natural geometric interpretation. For a point $x \in \mathcal{C}$ which is nearly feasible, the criticality is assessed with respect to the manifold layer to which $x$ belongs. In essence, $x$ satisfies the usual approximate criticality conditions for a Riemannian optimization problem, i.e., small Riemannian gradient and almost positive semidefinite Riemannian Hessian. However, these conditions are satisfied on the tangent space of a layer manifold $\mathcal{M}_x$ rather than on the target manifold $\mathcal{M}$. In Section 6.3, we show that approximate first- and second-order critical points of $g$ are related to ($\varepsilon$-FOCP) and ($\varepsilon$-SOCP) of (P), provided that the penalty parameter $\beta$ is large enough. Section 6.4 presents an algorithm that minimizes $g$ (Algorithm 7), and in so doing finds points which satisfy ($\varepsilon$-FOCP) and ($\varepsilon$-SOCP) for (P). We present an informal version of the main complexity result of this chapter. For the complete statement and proof, see Theorem 6.17.

**Theorem 6.3** (Informal statement)**.** *Under A12, A13, A14, A15, given $\beta > 0$ large enough, Algorithm 7 produces an* $(\varepsilon_1, 2\varepsilon_1)$-FOCP *of* (P) *in at most $\mathcal{O}\left(\varepsilon_1^{-2}\right)$ iterations. Algorithm 7 also produces an* $(\varepsilon_1, 2\varepsilon_1, \varepsilon_2 + C\varepsilon_1)$-SOCP *of* (P) *in at most $\mathcal{O}\left(\max\{\varepsilon_1^{-2}, \varepsilon_2^{-3}\}\right)$ iterations, where $C \geq 0$ is a constant depending on the function $h$.*

These definitions of criticality using the Riemannian gradient and Hessian are naturally related to optimality conditions based on the Lagrangian function, which are commonly used in the optimization literature. Indeed, take $\varepsilon_0 \geq 0, \varepsilon_1 \geq 0, \varepsilon_2 \geq 0$. If $x \in \mathcal{E}$ is an $(\varepsilon_0, \varepsilon_1, \varepsilon_2)$-(SOCP) of (P), there exists $\lambda \in \mathbb{R}^m$ such that

$$\|h(x)\| \leq \varepsilon_0, \qquad\qquad \|\nabla_x \mathcal{L}(x, \lambda)\| \leq \varepsilon_1 \tag{6.9}$$

and

$$\left\langle \nabla^2_{xx} \mathcal{L}(x, \lambda)[v], v \right\rangle \geq -\varepsilon_2 \|v\|^2 \text{ for all } v \in \mathcal{E} \text{ such that } \mathrm{D}h(x)[v] = 0. \tag{6.10}$$

Conditions (6.9) and (6.10) which are used in (Birgin et al., 2018; Xie and Wright, 2021) are weaker than ($\varepsilon$-SOCP). Therefore, finding points which satisfy ($\varepsilon$-SOCP) is more demanding than finding points which satisfy (6.9) and (6.10). Moreover, recent works dealing with global complexity for constrained optimization have used a number of different definitions of approximate second-order criticality, which are not equivalent to each other. In Section 6.2 we compare and discuss these various conditions.

## 6.2 Related work

The study of complexity in optimization has been very active in recent years, both for constrained and unconstrained problems. The field focuses on giving guarantees on the worst-case number of iterations an algorithm requires to achieve a predetermined termination criterion. The first results dealt with the unconstrained case, where $\mathcal{M} = \mathcal{E}$. Among others, Nesterov (2004) shows that for Lipschitz differentiable $f$, gradient descent with an appropriate step size requires $\mathcal{O}(\varepsilon^{-2})$ iterations in the worst-case to find a point which satisfies $\|\nabla f(x)\| \leq \varepsilon$. This is sharp (Cartis et al., 2010), meaning that no algorithm using an oracle of the same order can have a better worst-case performance. Note that if Hessian information is available and $f$ is twice Lipschitz continuously differentiable, cubic regularization has a $\mathcal{O}(\varepsilon^{-3/2})$ complexity for approximate first-order critical points (Cartis et al., 2010). Using derivatives of higher order can further improve the rate of regularization methods (Birgin et al., 2017). Cartis et al. (2012) further show that a point which satisfies both $\|\nabla f(x)\| \leq \varepsilon$ and $\lambda_{\min}(\nabla^2 f(x)) \geq -\varepsilon$ can be found in $\mathcal{O}(\varepsilon^{-3})$ iterations using a cubic regularization method. This bound is also sharp.

The study of the constrained case adds some difficulties. We focus on problems with equality constraints and leave aside inequality constraints. For complexity bounds of constrained problems, the sharpness of the unconstrained bounds carries over to the constrained case. That is, the best worst-case bounds achievable for the constrained case are also $\mathcal{O}(\varepsilon^{-2})$ and $\mathcal{O}(\varepsilon^{-3})$ for first- and second-order points using first- and second-order methods respectively.

Under A12, problem (P) is defined over a smooth manifold. For some known manifolds, such as those described in (Absil et al., 2008), Riemannian optimization offers an elegant and efficient way to solve constrained optimization problems. Boumal et al. (2019) showed that Riemannian optimization algorithms have the same worst-case bounds as their unconstrained counterparts. That is, under a Lipschitz smoothness assumption, Riemannian gradient descent with an appropriate step size finds a point which satisfies $\|\mathrm{grad}_{\mathcal{M}} f(x)\| \leq \varepsilon$ in $\mathcal{O}(\varepsilon^{-2})$ iterations. Similarly, a Riemannian trust-region algorithm finds a point which satisfies $\|\mathrm{grad}_{\mathcal{M}} f(x)\| \leq \varepsilon$ and $\lambda_{\min}(\mathrm{Hess}_{\mathcal{M}} f(x)) \geq -\varepsilon \, \mathrm{Id}$ in $\mathcal{O}(\varepsilon^{-3})$ iterations. Riemannian optimization methods are applicable to manifolds $\mathcal{M}$ provided that one is able to compute retractions and generate a feasible sequence of iterates. This is sometimes impossible or too expensive computationally. This prompts the use of infeasible methods to solve (P).

Several different notions of approximate criticality for (P) are considered in the literature. We review and compare them now, together with existing algorithmic guarantees to find such points. Among those that cover approximate second-order critical points, the rates are either not optimal (worse than $\mathcal{O}(\varepsilon^{-3})$), or they rely on an unusual notion of criticality.

For a non-empty, closed convex set $\mathcal{F}$, Cartis et al. (2019) consider the problem $\min_{x \in \mathcal{F}} f(x)$ such that $h(x) = 0$, which is a problem class more general than (P). They propose a two-phase algorithm which finds approximate first-, second- and even third-order critical points. The first phase of their algorithm attempts to find an approximately feasible point. The second phase minimizes the cost function while tracking infeasibility and staying close to the feasible set. This approach is unlikely to perform well in practice but yields optimal complexity rates for finding approximately critical points, which for first- and second-order are respectively $\mathcal{O}(\varepsilon^{-2})$ and $\mathcal{O}(\varepsilon^{-3})$ iterations. Their notion of criticality is unusual. Considering the merit function

$$\mu(x,t) := \frac{1}{2}\left\|r(x,t)\right\|^2 := \frac{1}{2}\left\|\begin{pmatrix} h(x) \\ f(x)-t \end{pmatrix}\right\|^2,$$

an approximate second-order critical point $x \in \mathcal{E}$ is defined as satisfying

$$\phi^{\Delta}_{\mu,j}(x,t) \le \varepsilon\Delta^j \left\|r(x,t)\right\| \text{ for } j = 1,2 \tag{6.11}$$

where

$$\phi^{\Delta}_{\mu,j} := \mu(x,t) - \min_{\substack{d\in\mathcal{E} \\ \|d\|\le\Delta}} T_{\mu,j}(x,d),$$

is the largest feasible decrease of the $j$th order Taylor model $T_{\mu,j}(x,d)$ achievable at distance at most $\Delta$ from $x$.

Cifuentes and Moitra (2019) tackle the problem of minimizing semidefinite programs using the Burer-Monteiro factorization. They adapt the two-phase algorithm from (Cartis et al., 2019) so that the target points satisfy the following notion of criticality. For $\gamma > 0$ and $\boldsymbol{\varepsilon} = (\varepsilon_0, \varepsilon_1, \varepsilon_2)$, they define a point $x \in \mathcal{E}$ as $(\boldsymbol{\varepsilon}, \gamma)$-approximately feasible approximately 2-critical (AFAC) if there exists $\lambda \in \mathbb{R}^m$ such that:

$$\begin{aligned} \|h(x)\| \le \varepsilon_0, \ \ \|\nabla_x \mathcal{L}(x,\lambda)\| \le \varepsilon_1 \\ u^\top \nabla^2_{xx} L(x,\lambda) u \ge -\varepsilon_2, \quad \forall u \text{ of unit norm such that } \|\mathrm{D}h(x)^*[u]\| \le \gamma. \end{aligned} \tag{AFAC}$$

The set of directions $u \in \mathcal{E}$ that satisfy $\|\mathrm{D}h(x)^*[u]\| \le \gamma$ for some $\gamma > 0$ includes the tangent space defined by $\mathrm{D}h(x)^*[u] = 0$. Therefore, the condition (AFAC) implies (6.9) and (6.10) but the converse is not true. Under A12 and A15, along with uniform boundedness and Lipschitz continuity of $f$, $h$ and their derivatives on $\mathcal{C}$, Cifuentes and Moitra (2019) show that an (AFAC) point can be found in $\mathcal{O}\left(\max\left\{\varepsilon_0^{-2}\varepsilon_1^{-2}, \varepsilon_0^{-3}\varepsilon_2^{-3}\right\}\right)$ iterations. The smoothness and initialization assumptions made are mostly equivalent to the ones we make. Their complexity estimate is not optimal, but it is the first complexity estimate which guarantees an approximate solution to a semi-definite program with high probability using the Burer-Monteiro formulation. They point out that in adapting the two-phase algorithm from (Cartis et al., 2019) to guarantee (AFAC) points, a factor $\varepsilon_0^{-1}$ is lost in the complexity.

A recent point of interest in the literature has been the study of complexity for algorithms that belong to the family of augmented Lagrangian methods (ALM). These methods have always been popular, with good practical results, but worst-case complexity results were lacking. Augmented Lagrangian methods minimize $\mathcal{L}_\beta(x, \lambda)$ by updating the variables $x \in \mathcal{E}$ and $\lambda \in \mathbb{R}^m$ alternatively. At iteration $k$, the subproblem to update $x$ usually attempts to find approximate minimizers of $\mathcal{L}_\beta(\cdot, \lambda_k)$, while the multipliers are updated using the first-order step $\lambda_{k+1} = \lambda_k - \beta h(x_k)$. The penalty parameter $\beta$ is also typically increased throughout the iterations.

Xie and Wright (2021) analyse a proximal ALM, and suggest to solve the subproblems using a Newton-conjugate gradient algorithm from (Royer et al., 2020). For this second-order algorithm, they show a total iteration complexity to reach approximate first- and second-order critical points of $\mathcal{O}(\varepsilon^{-5.5})$ and $\mathcal{O}(\varepsilon^{-7})$. When $h$ is linear, their guarantees match the best-known result of $\mathcal{O}(\varepsilon^{-3})$ total iterations for second-order points. These results require the initial iterate to satisfy $\|h(x_0)\|^2 \leq \min(C_0/\rho, 1)$ for some constant $C_0 > 0$ and $\rho$ that increases as $\varepsilon$ and $\sigma$ decrease. This condition is difficult to verify in practice, when there is no simple way to generate a feasible point. Admittedly, it can also be difficult to satisfy A15 in general. The advantage of A15 is that generating an initial iterate in $\mathcal{C}$ does not depend on $\varepsilon$ but only on the function $h$. They require that for some $\rho_0 \leq 0$, the function $f(x) + \frac{\rho_0}{2}\|h(x)\|^2$ has compact level sets, and also that $f$ be upper-bounded on the set $\{x \in \mathcal{E} : \|h(x)\| \leq 1\}$. Their target points satisfy the conditions (6.9) and (6.10) for first- and second-order target points respectively. Note that these conditions are weaker than ($\varepsilon$-SOCP). If $x \in \mathcal{E}$ is an ($\varepsilon$-SOCP), then it satisfies (6.9) and (6.10) with multipliers $\lambda(x) \in \mathbb{R}^m$. However, Equations (6.9) and (6.10) do not imply ($\varepsilon$-SOCP). As a counter-example, in $\mathcal{E} = \mathbb{R}^n$ take $\varepsilon_1 = \varepsilon_2 = 1/2$ with the functions $h(x) = \|x\|^2 - 1$ and $f(x) = \langle x, w\rangle$ for some $w \in \mathcal{E}$, $\|w\| = 1$. The maximizer of this problem is the point $w \in \mathcal{E}$: it satisfies (6.9) and (6.10) with multiplier $\lambda = 1/4$. However, $\lambda_{\min}(\mathrm{Hess}_\mathcal{M} f(w)) = -1$ hence $w$ is not an ($\varepsilon$-SOCP).

Grapiglia and Yuan (2021) provide a worst-case complexity analysis for an augmented Lagrangian method that can be applied to both equality and inequality constraints. For $h(x_0) = 0$, the number of outer iterations to reach a first-order critical point is $\mathcal{O}\left(\varepsilon^{-2/(\alpha-1)}\right)$ where $\alpha > 1$ defines the rate of increase of the penalty parameter $\beta$, i.e., $\beta_{k+1} = \max\{(k+1)^\alpha, \beta_k\}$. In this way, increasing $\alpha$ worsens the conditioning of the subproblems and increases the inner iteration count, which is not included in the bound above. The definition of first-order critical point simplifies to (6.9) for problems with equality constraints only. Under the additional assumption that the penalty parameters $\beta_k$ stay bounded as $k \to \infty$, the outer complexity is improved to $\mathcal{O}(\log(\varepsilon^{-1}))$. It is debatable whether the assumption that the penalty parameters remain bounded is reasonable in practice, as increasing the penalty parameters is critical to ensure convergence.

Birgin and Martnez (2019) analyse the outer iteration complexity of the popular optimization software Algencan introduced in (Andreani et al., 2008). This software was designed with practical efficiency in mind. It handles both equalities and inequalities. It is also safeguarded, meaning that upper and lower bounds are imposed on the multipliers. That work also shows an outer iteration complexity of $\mathcal{O}(\log(\varepsilon^{-1}))$ in the case of bounded penalty parameters.

In (Conn et al., 1991), the authors present a classical augmented Lagragian algorithm, where the usual first-order update for the multipliers $\lambda_{k+1} = \lambda_k - \beta h(x)$ can be replaced by the least-squares update (6.3), namely, $\lambda_{k+1} = \lambda(x_{k+1})$. They show that if a limit point $x^*$ of the algorithm is feasible, then it is first-order critical for (P) with multipliers $\lambda(x^*)$ (Equation (6.7)).

Sahin et al. (2019) study the problem $\min_{x\in\mathbb{R}^n} f_1(x) + f_2(x)$ such that $F(x) = 0$ where $f_1$ is nonconvex and smooth, $f_2$ is proximal friendly and convex and $F\colon \mathbb{R}^n \to \mathbb{R}^m$ is a nonlinear operator. The potential nonsmoothness of $g$ extends the range of applications compared to (P). For an augmented Lagrangian algorithm, they report bounds of $\mathcal{O}(\varepsilon^{-3})$ and $\mathcal{O}(\varepsilon^{-5})$ outer iterations to find approximate first- and second-order critical points of the augmented Lagrangian, where the subproblems are solved with a first- or second-order solver respectively.

Optimization under orthogonality constraints appears in a number of applications and is an active area of research. Riemannian optimization methods can be used on the Stiefel manifold $\mathrm{St}(n, p) = \left\{X \in \mathbb{R}^{n\times p} : X^\top X = \mathrm{I}_n\right\}$ and the orthogonal group $\mathrm{O}(p) = \left\{X \in \mathbb{R}^{p\times p} : X^\top X = \mathrm{I}_p\right\}$. These algorithms require to perform an orthogonalization procedure, known as a retraction, at every step throughout the optimization process. When $p$ is small compared to $n$, fast retractions are available for the Stiefel manifold. However, when $p$ is large, computing these retractions is often the computational bottleneck (Gao et al., 2019). This has prompted the search for retraction-free algorithms to deal with orthogonality constraints. Our use of Fletcher's augmented Lagrangian is partially inspired from (Gao et al., 2019). Those authors propose an algorithm specific to the Stiefel manifold. The algorithm is a primal-dual scheme, which updates alternatively the variable $x$ and the multipliers $\lambda$. The primal update is obtained from approximately minimizing $\mathcal{L}_\beta(x, \lambda)$ over $x$, while $\lambda$ is updated using a simplified version of formula (6.3), which we call $\hat{\lambda}(\cdot)$. Consider $\hat{g}(x) = \mathcal{L}_\beta(x, \hat{\lambda}(x))$ the penalty where the least-squares multipliers $\lambda(\cdot)$ are replaced by the approximation $\hat{\lambda}(\cdot)$. In (Xiao et al., 2020), the authors further study the penalty function $\hat{g}(x)$ for the Stiefel manifold. Recognizing that this function is unbounded below on $\mathbb{R}^{n\times p}$, they add an artificial box constraint around Stiefel to prevent divergence and develop a second-order method to minimize $\hat{g}(x)$ with asymptotic convergence results. Ablin and Peyré (2021) also presents a retraction-free algorithm on the orthogonal group. Their algorithm is an infeasible method which converges to an orthogonal matrix through the minimization of a purposefully constructed potential energy function. They report a speed up over classical retraction-based methods on some large-scale problems.

In the recent works (Liu and Boumal, 2020; Jia et al., 2021), the authors consider a framework where, conceptually, part of the constraints are easy to project onto, while the other constraints are more difficult to handle, as can be the case for the constraints of (P). Mixed approaches are proposed where the easy constraints are dealt with in a Riemannian-like fashion, while the other constraints are penalized with an augmented Lagrangian function.

Table 6.1 presents a list of works that have significant theoretical results for a problem definition similar to (P). The table shows whether they consider second-order critical points, how the target points are defined and the results in terms of global complexity and local rate of convergence.

Around the year that augmented Lagrangian methods came about, the penalty function we use in this chapter, Fletcher's augmented Lagrangian, was introduced in (Fletcher, 1970). In this original work, some fundamental properties of the function were established, most notably, connecting critical points of $f$ on $\mathcal{M}$ and critical points of $g$. Rapcsák (1997) studies the modified Lagrangian function $\mathcal{L}(x, \lambda(x))$ and considers the properties of its critical points on a subset of $\mathcal{M}$ that is geodesically convex. Bertsekas (1982, section 4.3.2) also covers properties of $g$, as an exact penalty function that depends only on $x$ and does not use a variable for the Lagrange multipliers. We go over these properties in the next section, and extend them substantially to situations with approximate critical points of first- and second-order.

## 6.3 Properties of Fletcher's augmented Lagrangian

We now cover properties of the function $g$, Fletcher's augmented Lagrangian (Equation 6.6). In this section, we recall an original result from (Bertsekas, 1982) which establishes conditions under which the critical points and minimizers of $g$ and (P) are equivalent. The core of this section then establishes extensions of this result to the case of approximate critical points. That is, we show that approximate first- and second-order critical points of $g$ are also approximately critical for (P) in the sense of ($\varepsilon$-FOCP) and ($\varepsilon$-SOCP).

| Paper | local rate | complexity | target points | problem class | 2nd order |
|---|---|---|---|---|---|
| Our work | ✗* | $\mathcal{O}(\varepsilon^{-2})$ and $\mathcal{O}(\varepsilon^{-3})$ | ($\varepsilon$-SOCP) | $\min f(x)$ s.t. $h(x)=0$ | ✓ |
| (Polyak, 2009) | quadratic | ✗ | ✗ | $\min f(x)$ s.t. $h(x)=0$ | ✓ |
| (Cartis et al., 2019) | ✗ | $\mathcal{O}(\varepsilon^{-2})$ and $\mathcal{O}(\varepsilon^{-3})$ | (6.11) | $\min f(x)$ s.t. $h(x)=0, x\in C$ cvx | ✓ |
| (Xie and Wright, 2021) | ✗ | $\mathcal{O}(\varepsilon^{-7})$ | ($\varepsilon$-SOCP) | $\min f(x)$ s.t. $h(x)=0$ | ✓ |
| (Cifuentes and Moitra, 2019) | ✗ | $\mathcal{O}(\varepsilon^{-6})$ | (AFAC) | BM for SDP | ✓ |
| (Andreani et al., 2007) | ✗ | ✗ | ($\varepsilon$-SOCP) + ineq. | $\min f(x)$ s.t. $h(x)=0, h_2(x)\leq 0$ | ✓ |
| (Xiao et al., 2020) | quadratic | ✗ | (6.8) | $\min f(X)$ s.t. $X\in \mathrm{St}(n,p)$ | ✓ |
| (Grapiglia and xiang Yuan, 2019) | ✗ | $\mathcal{O}(\varepsilon^{-2/(\alpha-1)})$, $\alpha>1^{\dagger}$ | ($\varepsilon$-FOCP) | $\min f(x)$ s.t. $h(x)=0, h_2(x)\leq 0$ | ✗ |
| (Gao et al., 2019) | linear | $\mathcal{O}(\varepsilon^{-2})$ | ($\varepsilon$-FOCP) | $\min f(X)$ s.t. $X\in \mathrm{St}(n,p)$ | ✗ |
| (Bai and Mei, 2018) | linear | $\mathcal{O}(\varepsilon^{-4})$ | ($\varepsilon$-FOCP) | $\min f(x)$ s.t. $h(x)=0$ | ✗ |
| (Bai et al., 2019) | linear | ✗ | (6.7) | $\min f(h(x))$ s.t. $\mathcal{A}(h(x))=b$, $f$ cvx | ✗ |
| (Birgin and Martnez, 2019) | ✗ | $\mathcal{O}(\log(1/\varepsilon))^{\ddagger}$ | ($\varepsilon$-FOCP) + ineq. | $\min f(x)$ s.t. $h(x)=0, h_2(x)\leq 0$ | ✗ |

Table 6.1: Summary of related works on complexity for constrained optimization. The complexity column gives the total iteration complexity to reach first-order target points and second-order when available. The last column indicates whether second-order critical points are considered. $*$) The algorithm that we present does not come with a guarantee of local quadratic convergence. However, it is possible to modify it to ensure local quadratic convergence, see Remark 6.1. †) The bound $\mathcal{O}(\varepsilon^{-2/(\alpha-1)})$ in (Grapiglia and Yuan, 2021) is an outer iteration complexity. ‡) The bound $\mathcal{O}(\log(1/\varepsilon))$ in (Birgin and Martnez, 2019) assumes that the penalty parameters $\beta_k$ remain bounded as $k\to\infty$.

We define $C_\lambda(x)$ as the operator norm of the differential of $\lambda(\cdot)$. Since $\mathcal{C}$ is assumed compact and $\lambda(\cdot)$ is smooth, this quantity is bounded.

**Definition 6.4.** *Under A12, for any $x \in \mathcal{C}$, we define the quantity*

$$C_\lambda(x) := \|\mathrm{D}\lambda(x)\|_{\mathrm{op}} = \sigma_1\left(\mathrm{D}\lambda(x)\right).$$

*Additionally, under A13, we define the constant*

$$\overline{C_\lambda} := \max_{x\in\mathcal{C}} \|\mathrm{D}\lambda(x)\|_{\mathrm{op}} < \infty.$$

**Definition 6.5.** *Under A12, for $x \in \mathcal{C}$, we define the following quantities*

$$\begin{aligned}
\beta_1(x) &= \frac{\sigma_1(\mathrm{D}h(x))C_\lambda(x)}{2\sigma_{\min}^2(\mathrm{D}h(x))}\\
\beta_2(x) &= \frac{C_\lambda(x)}{\sigma_{\min}(\mathrm{D}h(x))}\\
\beta_3(x) &= \frac{1}{\sigma_{\min}(\mathrm{D}h(x))}.
\end{aligned}$$

*Additionally, under A13, we define the constants $\bar{\beta}_i = \max_{x\in\mathcal{C}} \beta_i(x)$ for $i = 1, 2, 3$.*

The following result connects first-order critical points and minimizers of $g$ and $f$ on $\mathcal{M}$. The original proof is adapted to our notations in Appendix A.2.

**Proposition 6.4** ((Bertsekas, 1982), Prop. 4.22)**.** *Let $g(x) = \mathcal{L}_\beta(x, \lambda(x))$ be Fletcher's augmented Lagrangian and assume $\mathcal{M} \subset \mathcal{D}$, where $\mathcal{D} = \{x \in \mathcal{E} : \mathrm{rank}(\mathrm{D}h(x)) = m\}$ and $\mathcal{M} = \{x \in \mathcal{E} : h(x) = 0\}$.*

1. *For any $\beta$, if $x$ is a first-order critical point of* (P)*, then $x$ is a first-order critical point of $g$.*

2. *Let $x \in \mathcal{D}$ and $\beta > \beta_1(x)$. If $x$ is a first-order critical point of $g$, then $x$ is a first-order critical point of* (P)*.*

3. *Let $x$ be a first-order critical point of* (P) *and let $K$ be a compact subset of $\mathcal{D}$. Assume $x$ is the unique global minimum of $f$ over $\mathcal{M} \cap K$ and that $x$ is in the interior of $K$. Then, there exists $\beta$ large enough such that $x$ is the unique global minimum of $g$ over $K$.*

4. *Let $x \in \mathcal{D}$ and $\beta > \beta_1(x)$. If $x$ is a local minimum of $g$, then $x$ is a local minimum of* (P)*.*

The previous result is encouraging. It shows that minimizing the function $g$ inside $\mathcal{D}$ provides a way to find minimizers of (P). However, in practice, algorithms can only find approximate first- and second-order critical points in finite time. With the above proposition, one is left wondering whether such approximate points for $g$ correspond to similarly approximate critical points for (P). The remainder of this section provides such guarantees.

The gradient of the augmented Lagrangian $\mathcal{L}_\beta$ with respect to $x$ is given by

$$\begin{aligned}\nabla_x \mathcal{L}_\beta(x, \lambda) &= \nabla f(x) - \mathrm{D}h(x)^*[\lambda] + 2\beta \mathrm{D}h(x)^*[h(x)] \\ &= \nabla f(x) - \mathrm{D}h(x)^*[\lambda - 2\beta h(x)].\end{aligned}$$

Owing to (6.4), we make the following central observation: the gradient of $\mathcal{L}_\beta$ with respect to its first argument, when evaluated at $(x, \lambda(x))$, splits into orthogonal components; one component in the tangent space $\mathrm{T}_x\mathcal{M}_x$, and one component in the normal space to $\mathcal{M}_x$ at $x$,

$$\nabla_x \mathcal{L}_\beta(x, \lambda(x)) = \mathrm{grad}_{\mathcal{M}_x} f(x) + 2\beta \mathrm{D}h(x)^*[h(x)]. \tag{6.12}$$

Owing to orthogonality, $\nabla_x \mathcal{L}_\beta(x, \lambda(x))$ is small if and only if the two terms on the right are small. It takes an easy computation to check that for all $x \in \mathcal{D}$ we have

$$\begin{aligned}\mathrm{D}g(x)[v] &= \mathrm{D}f(x)[v] - \langle \mathrm{D}\lambda(x)[v], h(x)\rangle - \langle \lambda(x), \mathrm{D}h(x)[v]\rangle + 2\beta \langle h(x), \mathrm{D}h(x)[v]\rangle \\ &= \langle \nabla f(x), v\rangle - \langle \mathrm{D}h(x)^*[\lambda(x) - 2\beta h(x)], v\rangle - \langle \mathrm{D}\lambda(x)^*[h(x)], v\rangle \\ &= \langle \nabla_x \mathcal{L}_\beta(x, \lambda(x)), v\rangle - \langle \mathrm{D}\lambda(x)^*[h(x)], v\rangle .\end{aligned}$$

Thus, for all $x \in \mathcal{D}$,

$$\begin{aligned}\nabla g(x) &= \nabla_x \mathcal{L}_\beta(x, \lambda(x)) - \mathrm{D}\lambda(x)^*[h(x)] \\ &= \mathrm{grad}_{\mathcal{M}_x} f(x) + 2\beta \mathrm{D}h(x)^*[h(x)] - \mathrm{D}\lambda(x)^*[h(x)] .\end{aligned} \tag{6.13}$$

Therefore, for $x \in \mathcal{M}$, $\nabla g(x) = \mathrm{grad}_{\mathcal{M}} f(x)$. Consequently, for any value of $\beta$, if $x$ satisfies the constraints $h(x) = 0$, that is, if $x$ is on the manifold $\mathcal{M}$, then $x$ is first-order critical for $f$ on $\mathcal{M}$ (Equation(6.7)) if and only if $\nabla g(x) = 0$.

## 6.3.1 Approximate first-order criticality

In this section, we show that if $\nabla g(x)$ is small at some $x \in \mathcal{C}$, the point $x$ is approximately first-order critical for (P) in the sense of ($\varepsilon$-FOCP). We first recall a lemma about singular values.

**Lemma 6.5** ((Horn and Johnson, 1991), Theorem 3.3.16). *Let $A, B \in \mathbb{R}^{m\times n}$ and let $q = \min(m,n)$,*

$$\sigma_q(A - B) \geq \sigma_q(A) - \sigma_1(B)$$

**Proposition 6.6.** *Under A12, take $\varepsilon_1 \geq 0$ and $x \in \mathcal{C}$ with $\beta > \max\{\beta_2(x), \beta_3(x)\}$. If $\|\nabla g(x)\| \leq \varepsilon_1$, then*

$$\|h(x)\| \leq \frac{\varepsilon_1}{\beta\sigma_{\min}(\mathrm{D}h(x))} \leq \varepsilon_1 \quad \textit{and} \quad \left\|\mathrm{grad}_{\mathcal{M}_x} f(x)\right\| \leq \left(1 + \frac{C_\lambda(x)}{\beta\sigma_{\min}(\mathrm{D}h(x))}\right)\varepsilon_1 \leq 2\varepsilon_1.$$

*Proof.* We remember from (6.13) that

$$\begin{aligned}\nabla g(x) &= \mathrm{grad}_{\mathcal{M}_x} f(x) + 2\beta \mathrm{D}h(x)^*[h(x)] - \mathrm{D}\lambda(x)^*[h(x)] \\ &= \mathrm{grad}_{\mathcal{M}_x} f(x) - \mathrm{Proj}_x\left(\mathrm{D}\lambda(x)^*[h(x)]\right) + 2\beta\mathrm{D}h(x)^*[h(x)] - \mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*[h(x)]\right)\end{aligned}$$

where $\mathrm{Proj}_x^\perp = \mathrm{Id} - \mathrm{Proj}_x$, is the orthogonal projection on $\mathrm{N}_x\mathcal{M}_x = (\mathrm{T}_x\mathcal{M}_x)^\perp$, the normal space to $\mathcal{M}_x$ at $x$. We have decomposed the right-hand side in two tangent and two normal terms with respect to the manifold $\mathcal{M}_x$. By orthogonality, $\|\nabla g(x)\| \leq \varepsilon_1$ implies that both the tangent and normal components have norm smaller than $\varepsilon_1$. For the normal terms this yields,

$$\left\|2\beta\mathrm{D}h(x)^*[h(x)] - \mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*[h(x)]\right)\right\| = \left\|\left(2\beta\mathrm{D}h(x)^* - \mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*\right)\right)[h(x)]\right\| \leq \varepsilon_1. \tag{6.14}$$

Note that $\mathrm{D}h(x)^*$ is nonsingular since $x \in \mathcal{C}$. We show that $\beta$ is large enough so that the operator $\left(2\beta\mathrm{D}h(x)^* - \mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*\right)\right)$ is nonsingular. We use Lemma 6.5 to write

$$\sigma_{\min}\left(2\beta\mathrm{D}h(x)^* - \mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*\right)\right) \geq \sigma_{\min}\left(2\beta\mathrm{D}h(x)^*\right) - \sigma_{\max}\left(\mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*\right)\right).$$

The assumption on $\beta$ then provides

$$\begin{aligned}\sigma_{\min}\left(2\beta\mathrm{D}h(x)^*\right) - \sigma_{\max}\left(\mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*\right)\right) &\geq 2\beta\sigma_{\min}\left(\mathrm{D}h(x)\right) - C_\lambda(x) \\ &> \beta\sigma_{\min}\left(\mathrm{D}h(x)\right) > 1.\end{aligned}$$

We inject this into (6.14) to find:

$$\begin{aligned}\|h(x)\| &\leq \frac{\varepsilon_1}{\sigma_{\min}\left(2\beta\mathrm{D}h(x)^* - \mathrm{Proj}_x^\perp\left(\mathrm{D}\lambda(x)^*\right)\right)} \\ &\leq \frac{\varepsilon_1}{\beta\sigma_{\min}(\mathrm{D}h(x))} \leq \varepsilon_1.\end{aligned}$$

Now we use the tangent terms:

$$\begin{aligned}\varepsilon_1 &\geq \left\|\operatorname{grad}_{\mathcal{M}_x} f(x) - \operatorname{Proj}_x\left(\mathrm{D}\lambda(x)^*[h(x)]\right)\right\| \\ &\geq \left\|\operatorname{grad}_{\mathcal{M}_x} f(x)\right\| - \left\|\operatorname{Proj}_x\left(\mathrm{D}\lambda(x)^*[h(x)]\right)\right\| \\ &\geq \left\|\operatorname{grad}_{\mathcal{M}_x} f(x)\right\| - \left\|\mathrm{D}\lambda(x)^*[h(x)]\right\| \\ &\geq \left\|\operatorname{grad}_{\mathcal{M}_x} f(x)\right\| - C_\lambda(x)\left\|h(x)\right\| \\ &\geq \left\|\operatorname{grad}_{\mathcal{M}_x} f(x)\right\| - C_\lambda(x)\frac{\varepsilon_1}{\beta\sigma_{\min}(\mathrm{D}h(x))}.\end{aligned}$$

This allows to conclude $\left\|\operatorname{grad}_{\mathcal{M}_x} f(x)\right\| \leq \varepsilon_1 + \frac{C_\lambda(x)}{\beta\sigma_{\min}(\mathrm{D}h(x))}\varepsilon_1 \leq 2\varepsilon_1$. □

**Corollary 6.7.** *Under A12 and A13, take $\varepsilon_1 \geq 0$ and $x \in \mathcal{C}$. If $\beta$ satisfies the global bounds*

$$\beta > \bar{\beta}_2 \qquad \text{and} \qquad \beta > \bar{\beta}_3, \tag{6.15}$$

*and additionally $\|\nabla g(x)\| \leq \varepsilon_1$, then*

$$\|h(x)\| \leq \frac{\varepsilon_1}{\beta\underline{\sigma}} \leq \varepsilon_1 \qquad \text{and} \qquad \left\|\operatorname{grad}_{\mathcal{M}_x} f(x)\right\| \leq \left(1 + \frac{\overline{C_\lambda}}{\beta\underline{\sigma}}\right)\varepsilon_1 \leq 2\varepsilon_1,$$

*where $\underline{\sigma} \leq \min_{x\in\mathcal{C}} \sigma_{\min}\left(\mathrm{D}h(x)\right)$ and $\bar{\beta}_2, \bar{\beta}_3$ are introduced in Definition 6.5.*

### 6.3.2 Approximate second-order criticality

We now turn our attention to approximate second-order critical points of Fletcher's augmented Lagrangian. Similarly to first-order criticality, we investigate connections with ($\varepsilon$-SOCP) points for (P). As this section shows, some second-order critical points of (P) are not second-order critical points of $g$. However, strict second-order critical points of $g$ and (P) match, provided that $\beta$ is large enough. The Hessian of $g$ is obtained by taking a directional derivative of (6.13). For any $\dot{x} \in \mathcal{E}$,

$$\begin{aligned}\nabla^2 g(x)[\dot{x}] = \nabla^2 f(x)[\dot{x}] & \\ &- \left(\mathrm{D}(x \mapsto \mathrm{D}\lambda(x)^*)(x)[\dot{x}]\right)[h(x)] \\ &- \mathrm{D}\lambda(x)^*[\mathrm{D}h(x)[\dot{x}]] \\ &- \left(\mathrm{D}(x \mapsto \mathrm{D}h(x)^*)(x)[\dot{x}]\right)[\lambda(x) - 2\beta h(x)] \\ &- \mathrm{D}h(x)^*[\mathrm{D}\lambda(x)[\dot{x}] - 2\beta\mathrm{D}h(x)[\dot{x}]].\end{aligned} \tag{6.16}$$

We begin with a statement about feasible points which connects the Hessian of $g$ and the Riemannian Hessian of $f$ on $\mathcal{M}$.

**Proposition 6.8.** *For all $x \in \mathcal{M}$ we have*

$$\mathrm{Hess}_{\mathcal{M}} f(x) = \mathrm{Proj}_x \circ \nabla^2 g(x) \circ \mathrm{Proj}_x.$$

*Therefore, if $\nabla^2 g(x) \succeq -\varepsilon_2 \,\mathrm{Id}$ for some $\varepsilon_2 \geq 0$, then $\mathrm{Hess}_{\mathcal{M}} f(x) \succeq -\varepsilon_2 \,\mathrm{Id}$. If $\nabla^2 g(x) \succ 0$, then $\mathrm{Hess}_{\mathcal{M}} f(x) \succ 0$.*

*Proof.* We show that if $h(x) = 0$, $\mathrm{Proj}_x \circ \nabla^2 g(x) \circ \mathrm{Proj}_x = \mathrm{Hess}_{\mathcal{M}} f(x)$. Take $\dot{x} \in \mathcal{E}$ and plug $h(x) = 0$ into Equation (6.16). This gives

$$\begin{aligned}\nabla^2 g(x)[\dot{x}] = \nabla^2 f(x)[\dot{x}] &- \sum_{i=1}^m \lambda_i(x) \nabla^2 h_i(x)[\dot{x}] \\ &+ 2\beta \mathrm{D}h(x)^*[\mathrm{D}h(x)[\dot{x}]] \\ &- \mathrm{D}\lambda(x)^*[\mathrm{D}h(x)[\dot{x}]] - \mathrm{D}h(x)^*[\mathrm{D}\lambda(x)[\dot{x}]]\,.\end{aligned}$$

If, in addition, $\dot{x} \in \ker \mathrm{D}h(x)$, then

$$\left\langle \dot{x}, \nabla^2 g(x)[\dot{x}] \right\rangle = \left\langle \dot{x}, \nabla^2 f(x)[\dot{x}] \right\rangle - \sum_{i=1}^m \lambda_i(x) \left\langle \dot{x}, \nabla^2 h_i(x)[\dot{x}] \right\rangle.$$

Since $\ker \mathrm{D}h(x) = \mathrm{T}_x\mathcal{M}$, we conclude from Equation (6.5) that

$$\mathrm{Proj}_x \circ \nabla^2 g(x) \circ \mathrm{Proj}_x = \mathrm{Proj}_x \circ \left( \nabla^2 f(x) - \sum_{i=1}^m \lambda_i(x) \nabla^2 h_i(x) \right) \circ \mathrm{Proj}_x = \mathrm{Hess}_{\mathcal{M}} f(x). \square$$

In particular, for $\varepsilon_2 = 0$, the above result tells us that, irrespective of $\beta \geq 0$, if $x \in \mathcal{M}$ satisfies $\nabla g(x) = 0$ and $\nabla^2 g(x) \succeq 0$, the point $x$ is second-order critical for $f$ on $\mathcal{M}$ (Equation (6.8)). Unfortunately, the converse is not true (Fletcher, 1970). For $x \in \mathcal{M}$ such that $\mathrm{grad}_{\mathcal{M}} f(x) = 0$ and $\mathrm{Hess}_{\mathcal{M}} f(x) \succeq 0$, it is not possible in general to guarantee the existence of a finite $\beta$ such that $\nabla^2 g(x) \succeq 0$. Fletcher (1970) showed that the converse holds for positive definite Hessians.

**Proposition 6.9** ((Fletcher, 1970))**.** *If $x \in \mathcal{M}$ is a local minimizer of* (P) *with $\mathrm{Hess}_{\mathcal{M}} f(x) \succ 0$, there exists $\beta$ large enough such that $\nabla^2 g(x) \succ 0$.*

**Remark 6.1** (Local quadratic convergence)**.** *Assume that A12, A13 and A15 hold. For $\beta$ large enough, it is possible to use Fletcher's augmented Lagrangian to converge at a local quadratic rate towards isolated minimizers of* (P)*. Let $x^* \in \mathcal{M}$, a strict second-order critical point for* (P) *satisfying $\mathrm{grad} f(x^*) = 0$ and $\mathrm{Hess} f(x^*) \succ 0$. Provided $\beta$ is large enough, $x^*$ satisfies $\nabla g(x^*) = 0$ and $\nabla^2 g(x^*) \succ 0$ (Propositions 6.4 and 6.9). Take $x_0 \in \mathcal{C}$ close enough to $x^*$, the classical Newton method applied to the function $g$ produces a sequence which converges towards $x^*$ at a quadratic rate.*

Proposition 6.8 can be generalized to any point in $\mathcal{C}$ using an upper bound on the gradient norm of $g$.

**Proposition 6.10.** *Under A12, take $x \in \mathcal{C}$ with $\beta > \max\{\beta_2(x), \beta_3(x)\}$. Assume $\|\nabla g(x)\| \leq \varepsilon_1$ so that Proposition 6.6 applies at $x$. If $\nabla^2 g(x) \succeq -\varepsilon_2 \operatorname{Id}$, then*

$$\operatorname{Hess}_{\mathcal{M}_x} f(x) \succeq -(\varepsilon_2 + C(x)\varepsilon_1) \operatorname{Id}_{\mathrm{T}_x\mathcal{M}_x},$$

*where $C(x) = 2\left\|(\mathrm{D}(x \mapsto \mathrm{D}h(x)^*)(x))\right\|_{\mathrm{op}} / \sigma_{\min}(\mathrm{D}h(x)) + \left\|(\mathrm{D}(x \mapsto \mathrm{D}\lambda(x)^*)(x))\right\|_{\mathrm{op}}$.*

*Proof.* Since $x \in \mathcal{C}$, for any $\dot{x} \in \mathrm{T}_x\mathcal{M}_x$, Equation (6.5) gives the Riemannian Hessian of $f$ at $x$ and yields

$$\langle \dot{x}, \operatorname{Hess}_{\mathcal{M}_x} f(x)[\dot{x}]\rangle = \left\langle \dot{x}, \nabla^2 f(x)[\dot{x}] - \sum_{i=1}^{m} \lambda_i(x)\nabla^2 h_i(x)[\dot{x}] \right\rangle.$$

By assumption, for any $\dot{x} \in \mathcal{E}$,

$$\langle \dot{x}, \nabla^2 g(x)[\dot{x}]\rangle \geq -\varepsilon_2 \|\dot{x}\|^2.$$

In Equation (6.16), take $\dot{x} \in \mathrm{T}_x\mathcal{M}_x = \ker(\mathrm{D}h(x))$ and remember that the span of $\mathrm{D}h(x)^*$ is orthogonal to $\mathrm{T}_x\mathcal{M}_x$. This gives

$$\begin{aligned}\left(\operatorname{Proj}_x \circ \nabla^2 g(x) \circ \operatorname{Proj}_x\right)[\dot{x}] = \operatorname{Proj}_x \circ \Big(&\nabla^2 f(x)[\dot{x}] \\ &- (\mathrm{D}(x \mapsto \mathrm{D}h(x)^*)(x)[\dot{x}])\,[\lambda(x) - 2\beta h(x)] \\ &- (\mathrm{D}(x \mapsto \mathrm{D}\lambda(x)^*)(x)[\dot{x}])\,[h(x)]\Big).\end{aligned}$$

For clarity, we write $F_h(x) = \mathrm{D}(x \mapsto \mathrm{D}h(x)^*)(x)$ and $F_\lambda(x) = \mathrm{D}(x \mapsto \mathrm{D}\lambda(x)^*)(x)$. We compute the derivative

$$-\left(F_h(x)[\dot{x}]\right)[\lambda(x) - 2\beta h(x)] = -\sum_{i=1}^{m}(\lambda_i(x) - 2\beta h_i(x))\nabla^2 h_i(x)[\dot{x}],$$

which gives

$$\begin{aligned}\left\langle \dot{x}, \operatorname{Proj}_x \circ \nabla^2 g(x) \circ \operatorname{Proj}_x[\dot{x}]\right\rangle = &\left\langle \dot{x}, \nabla^2 f(x)[\dot{x}] - \sum_{i=1}^{m} \lambda_i(x)\nabla^2 h_i(x)[\dot{x}] \right\rangle \\ &+ \langle \dot{x}, 2\beta\left(F_h(x)[\dot{x}]\right)[h(x)]\rangle \\ &- \langle \dot{x}, \left(F_\lambda(x)[\dot{x}]\right)[h(x)]\rangle \\ \geq &-\varepsilon_2 \|\dot{x}\|^2.\end{aligned}$$

The formula for $\mathrm{Hess}_{\mathcal{M}_x} f(x)$ has appeared on the right-hand side. Using $\|h(x)\| \leq \frac{\varepsilon_1}{\beta \sigma_{\min}(\mathrm{D}h(x))} \leq \varepsilon_1$ from Proposition 6.6, we find

$$\begin{aligned}
\left\langle \dot{x}, \nabla^2 f(x)[\dot{x}] - \sum_{i=1}^{m} \lambda_i(x) \nabla^2 h_i(x)[\dot{x}] \right\rangle &\geq -2\beta \left\langle \dot{x}, \left(F_h(x)[\dot{x}]\right)[h(x)] \right\rangle \\
&\quad + \left\langle \dot{x}, \left(F_\lambda(x)[\dot{x}]\right)[h(x)] \right\rangle - \varepsilon_2 \|\dot{x}\|^2 \\
&\geq -2\beta \|F_h(x)\|_{\mathrm{op}} \|\dot{x}\|^2 \|h(x)\| \\
&\quad - \|F_\lambda(x)\|_{\mathrm{op}} \|\dot{x}\|^2 \|h(x)\| - \varepsilon_2 \|\dot{x}\|^2 \\
&\geq -2\beta \|F_h(x)\|_{\mathrm{op}} \|\dot{x}\|^2 \frac{\varepsilon_1}{\beta \sigma_{\min}(\mathrm{D}h(x))} \\
&\quad - \|F_\lambda(x)\|_{\mathrm{op}} \|\dot{x}\|^2 \varepsilon_1 - \varepsilon_2 \|\dot{x}\|^2 \\
&\geq -\varepsilon_2 \|\dot{x}\|^2 - \Big(2 \|F_h(x)\|_{\mathrm{op}} / \sigma_{\min}(\mathrm{D}h(x)) \\
&\quad + \|F_\lambda(x)\|_{\mathrm{op}}\Big) \varepsilon_1 \|\dot{x}\|^2 . \qquad \square
\end{aligned}$$

**Corollary 6.11.** *Under A12 and A13, take $x \in \mathcal{C}$ with $\beta > \max\left\{\bar{\beta}_2, \bar{\beta}_3\right\}$. Assume $\|\nabla g(x)\| \leq \varepsilon_1$ so that Corollary 6.7 applies. If $\nabla^2 g(x) \succeq -\varepsilon_2 Id$, then*

$$\mathrm{Hess}_{\mathcal{M}_x} f(x) \succeq -(\varepsilon_2 + C\varepsilon_1)\, \mathrm{Id}_{\mathrm{T}_x \mathcal{M}_x},$$

*where $C = \max_{x \in \mathcal{C}} 2 \|(\mathrm{D}(x \mapsto \mathrm{D}h(x)^*)(x))\|_{\mathrm{op}} / \underline{\sigma} + \|(\mathrm{D}(x \mapsto \mathrm{D}\lambda(x)^*)(x))\|_{\mathrm{op}}$.*

### 6.3.3 Property of the region $\mathcal{C}$

The algorithms we design and analyze in later sections produce iterates which are initialized in a given connected component of $\mathcal{C} = \{x \in \mathcal{E} : \|h(x)\| = R\}$, with $R$ as in A12. Since $\mathcal{C}$ may in general have more than one such component, and since we hope in particular that our iterates converge to a feasible point, that is, to a point in $\mathcal{M} = \{x \in \mathcal{E} : h(x) = 0\}$, it is natural to wonder whether each connected component of $\mathcal{C}$ intersects with $\mathcal{M}$. That is indeed the case, as we now show.

**Proposition 6.12.** *Under A12, every connected component of $\mathcal{C}$ contains a point $\bar{z} \in \mathcal{E}$ such that $h(\bar{z}) = 0$.*

*Proof.* Define $\varphi(x) = \frac{1}{2} \|h(x)\|^2$ and take any $x_0 \in \mathcal{C} = \{x \in \mathcal{E} : \varphi(x) \leq R^2/2\}$. Consider the following differential system:

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t} x(t) = -\nabla \varphi(x(t)) \\ x(0) = x_0. \end{cases} \tag{6.17}$$

The fundamental theorem of flows (Lee, 2018, Theorem A.42) guarantees the existence of a unique maximal integral curve starting at $x_0$ for (6.17). Let $z(\cdot)\colon I \to \mathcal{E}$ denote this maximal integral curve and $T > 0$ be the supremum of the interval $I$ on which $z(\cdot)$ is defined. We rely on the Escape Lemma (Lee, 2018, Lemma A.43) to show that $z(t)$ is defined for all times $t \geq 0$. For $t < T$, we write $\ell = \varphi \circ z$ and find

$$\begin{aligned}\ell'(t) &= \mathrm{D}\varphi(z(t))\left[\frac{\mathrm{d}}{\mathrm{d}t}z(t)\right] = \left\langle \nabla\varphi(z(t)), \frac{\mathrm{d}}{\mathrm{d}t}z(t)\right\rangle\\ &= -\left\|\nabla\varphi(z(t))\right\|^2\\ &= -\left\|\mathrm{D}h(z(t))^*[h(z(t))]\right\|^2 \leq 0.\end{aligned}$$

This implies that $z(t) \in \mathcal{C}$ for all $0 \leq t < T$. We show that the trajectory $z(t)$ has finite length. To that end, we note that $\frac{1}{2}\left\|\nabla\varphi(x)\right\|^2 = \frac{1}{2}\left\|\mathrm{D}h(x)[h(x)]\right\|^2 \geq \underline{\sigma}^2\frac{1}{2}\left\|h(x)\right\|^2 = \underline{\sigma}^2\varphi(x)$ for all $x \in \mathcal{C}$. The length of the trajectory from time $t = 0$ to $t = T$ is given by

$$\begin{aligned}\int_0^T \left\|\frac{\mathrm{d}}{\mathrm{d}t}z(t)\right\|\mathrm{d}t &= \int_0^T \left\|-\nabla\varphi(z(t))\right\|\mathrm{d}t\\ &= \int_0^T \frac{\left\|\nabla\varphi(z(t))\right\|^2}{\left\|\nabla\varphi(z(t))\right\|}\mathrm{d}t\\ &= \int_0^T \frac{\left\langle -\nabla\varphi(z(t)), \frac{\mathrm{d}}{\mathrm{d}t}z(t)\right\rangle}{\left\|\nabla\varphi(z(t))\right\|}\mathrm{d}t\\ &= \int_0^T \frac{-(\varphi\circ z)'(t)}{\left\|\nabla\varphi(z(t))\right\|}\mathrm{d}t\\ &\leq \int_0^T \frac{-(\varphi\circ z)'(t)}{\underline{\sigma}\sqrt{2(\varphi\circ z)(t)}}\mathrm{d}t\\ &= \frac{-\sqrt{2}}{\underline{\sigma}}\left[\sqrt{\varphi(z(T))} - \sqrt{\varphi(z(0))}\right]\\ &\leq \frac{\sqrt{2\varphi(z(0))}}{\underline{\sigma}}.\end{aligned}$$

The length is bounded independently of $T$ and therefore the flow has finite length. The Escape Lemma states that for a maximum integral curve $z(\cdot)\colon I \to \mathcal{E}$, if $I$ has a finite upper bound, then the curve $z(\cdot)$ must be unbounded. The converse ensures that, since $z(\cdot)$ is contained in a compact set, the interval $I$ does not have a finite upper bound and therefore, $I = \mathbb{R}_+$. Since the trajectory $z(t)$ is bounded for $t \geq 0$, it must have an accumulation point $\bar{z}$. From A12, we have $\sigma_{\min}(\mathrm{D}h(z(t)) \geq \underline{\sigma} > 0$ for all $t \geq 0$. This gives the bound $\ell'(t) \leq -\underline{\sigma}^2\left\|h(z(t))\right\|^2 = -2\underline{\sigma}^2\ell(t)$. Gronwall's inequality then yields

$$\ell(t) \leq \varphi(x_0)e^{-2\underline{\sigma}^2 t}.$$

Therefore $\ell(t) \to 0$ as $t \to \infty$, which implies $h(z(t)) \to 0$ as $t \to \infty$. We conclude that the accumulation point satisfies $h(\bar{z}) = 0$. Since $\mathcal{C}$ is closed, the point $\bar{z}$ is in $\mathcal{C}$. Therefore, $\bar{z}$ is both in $\mathcal{M}$ and in the connected component of $\mathcal{C}$ that contains $z(0) = x_0$. $\square$

## 6.4 Gradient-Eigenstep algorithm

In this section we present an optimization algorithm to minimize $g$. It is designed to remain in $\mathcal{C}$, the region of interest where $\lambda(x)$ is well defined. The algorithm alternates between gradient steps (first-order) and eigensteps (second-order) to reach approximate second-order critical points of $g$. If the gradient of $g$ is large, a gradient step on $g$ is used. If the gradient of $g$ is below a tolerance, the algorithm follows a direction of negative curvature of the Hessian of $g$. Gradient and eigensteps must fulfil two purposes: they must guarantee a sufficient decrease of the penalty $g$ and also ensure that the next iterate remains inside $\mathcal{C}$. This is detailed in Algorithm 7. Given values $\varepsilon_1 > 0, \varepsilon_2 > 0$, the algorithm returns a point which satisfies $\|\nabla g(x)\| \leq \varepsilon_1$ and $\lambda_{\min}(\nabla^2 g(x)) \geq -\varepsilon_2$. This ensures that $x$ is an $(\varepsilon_1, \varepsilon_2 + C(x)\varepsilon_1)$-SOCP of (P) according to Proposition 6.10.

Whenever $\|\nabla g(x)\| > \varepsilon_1$, a gradient step is used and we require that the step-length $\alpha$ satisfies a classical Armijo sufficient decrease condition:

$$g(x) - g(x - \alpha \nabla g(x)) \geq c_1 \alpha \|\nabla g(x)\|^2, \tag{6.18}$$

for some $c_1 < 1$. The backtracking procedure for gradient steps is presented in Algorithm 8. This is a classical backtracking modified to additionally ensure that the iterates stay in $\mathcal{C}$, which is always possible for small enough steps, as we show in Proposition 6.13.

Given $x \in \mathcal{C}$ with $\|\nabla g(x)\| \leq \varepsilon_1$ and $\lambda_{\min}(\nabla^2 g(x)) < -\varepsilon_2$, a second-order step must be applied. We compute a unit-norm vector $d \in \mathcal{E}$ such that $\langle d, \nabla^2 g(x)[d] \rangle < -\varepsilon_2 \|d\|^2$. To ensure sufficient decrease, we wish to find $\alpha > 0$ such that

$$g(x) - g(x + \alpha d) \geq -c_2 \alpha^2 \langle d, \nabla^2 g(x)[d] \rangle, \tag{6.19}$$

for some $0 < c_2 < 1/2$. In Algorithm 9, we detail the backtracking used for second-order steps. It is designed to ensure (6.19) and additionally that the steps are small enough to remain in $\mathcal{C}$, which is possible as we show in Proposition 6.15.

We define some bounds on the derivatives of $g$, which are finite due to the smoothness of $g$ in $\mathcal{C}$ and compactness of $\mathcal{C}$ (A13).

**Definition 6.6.** *Under A12 and A13, define the constants*

$$L_g = \max_{x \in \mathcal{C}} \left\|\nabla^2 g(x)\right\|_{\mathrm{op}} \qquad \textit{and} \qquad M_g = \max_{x \in \mathcal{C}} \left\|\nabla^3 g(x)\right\|_{\mathrm{op}}. \tag{6.20}$$

### 6.4.1 Algorithm

We define Algorithm 7, a procedure which combines first- and second-order steps to minimize $g$ up to approximate second-order criticality if $\varepsilon_2 < \infty$. Setting $\varepsilon_2 = \infty$ gives a first-order version of the algorithm. To run Algorithm 7, we assume that the value of the penalty parameter $\beta$ does not change and is large enough in the following sense.

**A16.** *Under A12 and A13, $\beta$ is chosen such that $\beta > \overline{\beta}$ with*

$$\overline{\beta} := \max \left\{ \overline{\beta}_1, \overline{\beta}_2, \overline{\beta}_3 \right\}, \tag{6.21}$$

*where $\overline{\beta}_i$ for $i = 1, 2, 3$ are defined in Definition 6.5.*

In Section 6.5, we show how this assumption can be removed, using an adaptive scheme for $\beta$.

**Algorithm 7** Gradient-Eigenstep

```
1: Given: Function $f$ and $h$, $x_0 \in \mathcal{C}$, $\beta > 0$, $0 \leq \varepsilon_1 \leq R/2$ and $\varepsilon_2 \geq 0$.
2: Set $k = 0$
3: while no optional stopping criterion triggers do
4:     if $\|\nabla g(x_k)\| > \varepsilon_1$ then
5:         $x_{k+1} = x_k - t\nabla g(x_k)$ with $t$ given by Algorithm 8
6:     else if $\varepsilon_2 < \infty$ then
7:         if $\lambda_{\min}(\nabla^2 g(x_k)) < -\varepsilon_2$ then
8:             Find $d \in \mathcal{E}$ such that $\langle d, \nabla^2 g(x_k)[d] \rangle < -\varepsilon_2 \|d\|^2$, $\langle d, \nabla g(x_k) \rangle \leq 0$ and $\|d\| = 1$.
9:             $x_{k+1} = x_k + td$ where $t$ is given by Algorithm 9.
10:        else
11:            return $x_k$        ▷ $\|\nabla g(x_k)\| \leq \varepsilon_1$ and $\nabla^2 g(x_k) \succeq -\varepsilon_2 \operatorname{Id}$
12:        end if
13:    else
14:        return $x_k$        ▷ $\|\nabla g(x_k)\| \leq \varepsilon_1$
15:    end if
16:    $k = k + 1$
17: end while
```

**Algorithm 8** Gradient step backtracking, modified to stay in $\mathcal{C}$

```
1: Given: $x \in \mathcal{C}$, $\alpha_{01} > 0$, $0 < c_1 < 1$, $0 < \tau_1 < 1$.
2: Set $\alpha = \alpha_{01}$
3: while true do
4:     if $g(x) - g(x - \alpha\nabla g(x)) \geq c_1\alpha \|\nabla g(x)\|^2$ and $x - \alpha\nabla g(x) \in \mathcal{C}$ then
5:         return $\alpha$
6:     else
7:         $\alpha = \tau_1\alpha$
8:     end if
9: end while
```

**Algorithm 9** Eigenstep backtracking, modified to stay in $\mathcal{C}$

```
1: Given: $x \in \mathcal{C}$, unit-norm $d \in \mathcal{E}$, $\alpha_{02} > 0$, $0 < c_2 < 1/2$, $0 < \tau_2 < 1$.
2: Set $\alpha = \alpha_{02}$
3: while true do
4:     if $g(x) - g(x + \alpha d) \geq -c_2\alpha^2\langle d, \nabla^2 g(x)[d]\rangle$ and $x + \alpha d \in \mathcal{C}$ then
5:         return $\alpha$
6:     else
7:         $\alpha = \tau_2\alpha$
8:     end if
9: end while
```

## 6.4.2 First-order steps

We show that small enough gradient steps remain in $\mathcal{C}$.

**Proposition 6.13.** *Assume A12 holds with constant $R$ and A14 holds with constant $C_h$. Then, for all $x \in \mathcal{C}$, if $\beta > \beta_1(x)$, it holds that $x - t\nabla g(x)$ is in $\mathcal{C}$ for all $t$ in the interval $[0, t_1(x)]$ where $t_1(x)$ is defined by*

$$t_1(x) := \min\left(\sqrt{\frac{R}{2C_h}}\frac{1}{\|\nabla g(x)\|}, \frac{(2\beta\sigma_{\min}(\mathrm{D}h(x))^2 - \sigma_1(\mathrm{D}h(x))C_\lambda(x))R}{2C_h\|\nabla g(x)\|^2}, \frac{1}{2\beta\|\mathrm{D}h(x)\|_{\mathrm{op}}^2}\right), \tag{6.22}$$

*where $C_\lambda(x) = \|\mathrm{D}\lambda(x)\|_{\mathrm{op}}$ (Definition 6.4).*

*Proof.* Given $x \in \mathcal{C}$, consider the gradient step $x_t = x - t\nabla g(x)$ for some $t \geq 0$. We wish to find $t_{\max} > 0$ such that $x_t \in \mathcal{C}$ for all $t \in [0, t_{\max}]$. Using A14, we have

$$\begin{aligned} h(x_t) &= h(x - t\nabla g(x)) \\ &= h(x) - t\mathrm{D}h(x)[\nabla g(x)] + E\left(x, -t\nabla g(x)\right) \end{aligned}$$

where $\|E(x, -t\nabla g(x))\| \leq C_h\|t\nabla g(x)\|^2$. Using Equation (6.13) gives

$$h(x_t) = h(x) - t\mathrm{D}h(x)\big[\mathrm{grad}_{\mathcal{M}_x}f(x) + 2\beta\mathrm{D}h(x)^*[h(x)] - \mathrm{D}\lambda(x)^*[h(x)]\big] + E(x, -t\nabla g(x)).$$

Since $\mathrm{grad}_{\mathcal{M}_x}f(x)$ belongs to $\ker(\mathrm{D}h(x))$ by construction, one term cancels:

$$\begin{aligned} h(x_t) &= h(x) - 2\beta t\mathrm{D}h(x)[\mathrm{D}h(x)^*[h(x)]] + t\mathrm{D}h(x)[\mathrm{D}\lambda(x)^*[h(x)]] + E(x, -t\nabla g(x)) \\ &= (\mathrm{I}_m - 2\beta t\mathrm{D}h(x) \circ \mathrm{D}h(x)^*)[h(x)] + (t\mathrm{D}h(x) \circ \mathrm{D}\lambda(x)^*)[h(x)] + E(x, -t\nabla g(x)). \end{aligned}$$

Let $\sigma_1 \geq \cdots > \sigma_m > 0$ denote the singular values of $\mathrm{D}h(x)$. The eigenvalues of the symmetric operator $(\mathrm{I}_m - 2\beta t\mathrm{D}h(x) \circ \mathrm{D}h(x)^*)$ are $1 - 2\beta t\sigma_1^2 \leq \cdots \leq 1 - 2\beta t\sigma_m^2$. All these eigenvalues are smaller than one and are nonnegative provided $0 \leq 1 - 2\beta t\sigma_1^2$ and $t \geq 0$, or equivalently:

$$0 \leq t \leq \frac{1}{2\beta\|\mathrm{D}h(x)\|_{\mathrm{op}}^2}. \tag{6.23}$$

Under that assumption, we further find:

$$\|h(x_t)\| \le (1 - 2\beta t\sigma_m^2)\,\|h(x)\| + t\sigma_1\,\|\mathrm{D}\lambda(x)^*\|_{\mathrm{op}}\,\|h(x)\| + C_h t^2\,\|\nabla g(x)\|^2 .$$

We want to show $\|h(x_t)\| \le R$, which is indeed the case if

$$(1 - 2\beta t\sigma_m^2)\,\|h(x)\| + t\sigma_1 C_\lambda(x)\,\|h(x)\| + C_h\,\|\nabla g(x)\|^2\, t^2 \le R.$$

Thus, we seek conditions on $t$ to ensure that the following quadratic inequality in $t$ holds:

$$C_h\,\|\nabla g(x)\|^2\, t^2 + \left(\sigma_1 C_\lambda(x) - 2\beta\sigma_m^2\right)\|h(x)\|\, t + \|h(x)\| - R \le 0. \tag{6.24}$$

We branch into two cases. Firstly, consider $\|h(x)\| \in [R/2, R]$. In this case, (6.24) holds a fortiori if we remove the independent term $\|h(x)\| - R$ since the latter is nonpositive. By assumption, $\beta > \beta_1(x) = \sigma_1 C_\lambda(x)/2\sigma_m^2$, so the linear term is nonpositive. Therefore, we can upper bound the quadratic by setting $\|h(x)\| = R/2$. This shows that

$$\begin{aligned} &C_h\,\|\nabla g(x)\|^2\, t^2 + \left(\sigma_1 C_\lambda(x) - 2\beta\sigma_m^2\right)\|h(x)\|\, t + \|h(x)\| - R \\ &\quad\le C_h\,\|\nabla g(x)\|^2\, t^2 + \left(\sigma_1 C_\lambda(x) - 2\beta\sigma_m^2\right)\frac{R}{2} t. \end{aligned}$$

The above is a convex quadratic with two real roots. It is nonpositive — and (6.24) is satisfied — if:

$$0 \le t \le \frac{\left(2\beta\sigma_m^2 - \sigma_1 C_\lambda(x)\right) R}{2C_h\,\|\nabla g(x)\|^2}. \tag{6.25}$$

For $\|h(x)\| \in [0, R/2]$, the linear term in (6.24) is still nonpositive. Additionally, the constant term of the quadratic is upper bounded by $-R/2$. This establishes

$$C_h\,\|\nabla g(x)\|^2\, t^2 + \left(\sigma_1 C_\lambda(x)\,\|h(x)\| - 2\beta\sigma_m^2\,\|h(x)\|\right) t + \|h(x)\| - R \le C_h\,\|\nabla g(x)\|^2\, t^2 - \frac{R}{2}.$$

We infer that, for $\|h(x)\| \in [0, R/2]$, condition (6.24) is satisfied for

$$0 \le t \le \sqrt{\frac{R}{2C_h}}\frac{1}{\|\nabla g(x)\|}. \tag{6.26}$$

The main claim now follows by collecting the conditions in equations (6.23), (6.25) and (6.26). □

We now show that the backtracking in Algorithm 8 terminates in a finite number of steps and guarantees a sufficient decrease.

**Lemma 6.14** (Gradient step decrease)**.** *Take $x \in \mathcal{C}$ and $\beta > \beta_1(x)$. The backtracking procedure in Algorithm 8 terminates with a step-size $t \geq \tau_1 \min(\underline{\alpha_1}, t_1(x)) > 0$ where*

$$\underline{\alpha_1} = \min\left(\alpha_{01}, \frac{2(1-c_1)}{L_g}\right)$$

*and $t_1(x)$ is defined in Equation* (6.22)*. This guarantees the following decrease:*

$$g(x) - g(x - t\nabla g(x)) \geq c_1 \tau_1 \min(\underline{\alpha_1}, t_1(x)) \left\|\nabla g(x)\right\|^2. \tag{6.27}$$

*Proof.* From Proposition 6.13, we know that $x - \alpha\nabla g(x)$ is in $\mathcal{C}$ for every $0 \leq \alpha \leq t_1(x)$. We proceed to show that the Armijo decrease condition (6.18) is satisfied for any $0 \leq \alpha \leq \min(t_1(x), \underline{\alpha_1})$. For every $0 \leq \alpha \leq t_1(x)$, the norm of the Hessian of $g$ is bounded by the constant $L_g$ (Equation (6.20)), which implies that $\nabla g$ is $L_g$-Lipschitz continuous on the segment that connects $x$ and $x - t_1(x)\nabla g(x)$. Thus, for all $0 \leq \alpha \leq t_1(x)$, we have

$$\begin{aligned} g(x - \alpha\nabla g(x)) &\leq g(x) + \langle -\alpha\nabla g(x), \nabla g(x)\rangle + \frac{L_g}{2}\left\|\alpha\nabla g(x)\right\|^2 \\ &= g(x) + \left(\frac{\alpha L_g}{2} - 1\right)\alpha\left\|\nabla g(x)\right\|^2. \end{aligned}$$

This is equivalent to $g(x) - g(x - \alpha\nabla g(x)) \geq (1 - \alpha L_g/2)\alpha\left\|\nabla g(x)\right\|^2$. For $0 \leq \alpha \leq 2(1-c_1)/L_g$, we have $(1 - \alpha L_g/2) \geq c_1$. Hence, for $0 \leq \alpha \leq \min(t_1(x), \underline{\alpha_1})$, condition (6.18), $g(x) - g(x - \alpha\nabla g(x)) \geq c_1\alpha\left\|\nabla g(x)\right\|^2$ is satisfied. Given $\beta > \beta_1(x)$, one readily checks that $t_1(x)$ is positive. Since $\underline{\alpha_1}$ is also positive, there exists a nonempty interval, $]0, \min(\underline{\alpha_1}, t_1(x))]$, where the step size satisfies the Armijo condition and defines a next iterate inside $\mathcal{C}$. Therefore, Algorithm 8 returns a step $t$ satisfying $t \geq \tau_1 \min(\underline{\alpha_1}, t_1(x))$. In addition, the Armijo condition gives

$$\begin{aligned} g(x) - g(x - t\nabla g(x)) &\geq c_1 t\left\|\nabla g(x)\right\|^2 \\ &\geq c_1\tau_1 \min(\underline{\alpha_1}, t_1(x))\left\|\nabla g(x)\right\|^2. \end{aligned}$$ □

## 6.4.3 Second-order steps

We begin with a result which guarantees small enough steps stay in $\mathcal{C}$ when $\nabla g(x)$ is small.

**Proposition 6.15.** *Suppose A12 and A14 hold. Take $x \in \mathcal{C}$ with $\beta > \max\{\beta_1(x), \beta_2(x), \beta_3(x)\}$. Assume that $\|\nabla g(x)\| \leq \varepsilon_1$ for some $\varepsilon_1 \leq R/2$ so that Proposition 6.6 applies. For any $d \in \mathcal{E}$ with $\|d\| = 1$, the point $x + td$ is in $\mathcal{C}$ for all $t$ in the interval $[0, t_2(x)]$ with $t_2(x)$ defined by*

$$t_2(x) := \left(-\sigma_1(\mathrm{D}h(x)) + \sqrt{\sigma_1(\mathrm{D}h(x))^2 + 2C_hR}\right)/2C_h. \tag{6.28}$$

*Proof.* Since $\|\nabla g(x)\| \le \varepsilon_1$, Proposition 6.6 ensures $\|h(x)\| \le \varepsilon_1$. For $t > 0$, A14 yields

$$\begin{aligned} h(x+td) &= h(x) + t\mathrm{D}h(x)[d] + E(x,td) \\ \|h(x+td)\| &\le \|h(x)\| + t\sigma_1 \|d\| + C_h t^2 \|d\|^2 \\ &\le \varepsilon_1 + t\sigma_1 + C_h t^2, \end{aligned}$$

where $\sigma_1$ is the largest singular value of $\mathrm{D}h(x)$. We want to find the values of $t \ge 0$ for which $\varepsilon_1 + t\sigma_1 + C_h t^2 \le R$. The convex quadratic $t \mapsto C_h t^2 + \sigma_1 t + \varepsilon_1 - R$ has roots $\left(-\sigma_1 \pm \sqrt{\sigma_1^2 - 4(\varepsilon_1 - R)C_h}\right)/2C_h$, which for $\varepsilon_1 < R$ are real and of opposite signs. Hence, the quadratic is nonpositive for all $t$ such that

$$0 \le t \le \left(-\sigma_1 + \sqrt{\sigma_1^2 + 4|R - \varepsilon_1|C_h}\right)/2C_h.$$

By assumption, $\varepsilon_1 \le R/2$ and therefore $x + td$ belongs to $\mathcal{C}$ for all $t$ such that

$$0 \le t \le \left(-\sigma_1 + \sqrt{\sigma_1^2 + 4C_h R/2}\right)/2C_h. \qquad \square$$

We now show that the backtracking of Algorithm 9 terminates in a finite number of steps and guarantees a sufficient decrease.

**Lemma 6.16** (Eigenstep decrease). *Take $x \in \mathcal{C}$ and $\beta > \max\{\beta_1(x), \beta_2(x), \beta_3(x)\}$ with $\|\nabla g(x)\| \le \varepsilon_1$ for some $\varepsilon_1 \le R/2$. Assume there exists a direction $d \in \mathcal{E}$ such that $\|d\| = 1$, $\langle d, \nabla^2 g(x)[d]\rangle < -\varepsilon_2$ for some $\varepsilon_2 > 0$ and $\langle d, \nabla g(x)\rangle \le 0$. The backtracking procedure in Algorithm 9 terminates with a step size $t \ge \tau_2 \min(\alpha_2(x), t_2(x)) > 0$ where*

$$\alpha_2(x) = \min\left(\alpha_{02}, \frac{3|2c_2 - 1||\langle d, \nabla^2 g(x)[d]\rangle|}{M_g}\right)$$

*and $t_2(x)$ is defined in Equation (6.28). This ensures the following decrease:*

$$g(x) - g(x+td) \ge -c_2\tau_2^2 \min(\alpha_2(x), t_2(x))^2 \langle d, \nabla^2 g(x)[d]\rangle. \tag{6.29}$$

*Proof.* From Proposition 6.15, the point $x + \alpha d$ is in $\mathcal{C}$ for all $0 \le \alpha \le t_2(x)$. We show that for all $0 \le \alpha \le \min(\alpha_2(x), t_2(x))$, the decrease condition (6.19) is satisfied. For every $0 \le \alpha \le t_2(x)$, the norm of the third derivative of $g$ is bounded by the constant $M_g$ (Equation (6.20)), which implies that $\nabla^2 g$ is $M_g$-Lipschitz continuous on the segment that connects $x$ and $x + t_2(x)d$. Thus, for all $0 \le \alpha \le t_2(x)$, we have

$$\begin{aligned} g(x+\alpha d) &\le g(x) + \alpha\langle d, \nabla g(x)\rangle + \frac{\alpha^2}{2}\langle d, \nabla^2 g(x)[d]\rangle + \frac{M_g}{6}\alpha^3 \|d\|^3 \\ &\le g(x) + \frac{\alpha^2}{2}\langle d, \nabla^2 g(x)[d]\rangle + \frac{M_g}{6}\alpha^3 \\ &\le g(x) + \frac{\alpha^2}{2}\left(\langle d, \nabla^2 g(x)[d]\rangle + \frac{M_g \alpha}{3}\right). \end{aligned}$$

The sufficient decrease condition (6.19), $g(x) - g(x + \alpha d) \geq -c_2\alpha^2\langle d, \nabla^2 g(x)[d]\rangle$, is satisfied if

$$-\frac{\alpha^2}{2}\left(\langle d, \nabla^2 g(x)[d]\rangle + \frac{M_g\alpha}{3}\right) \geq -c_2\alpha^2\langle d, \nabla^2 g(x)[d]\rangle.$$

This is equivalent to

$$\begin{aligned}\langle d, \nabla^2 g(x)[d]\rangle + \frac{M_g\alpha}{3} &\leq 2c_2\langle d, \nabla^2 g(x)[d]\rangle \\ \alpha &\leq \frac{3(2c_2 - 1)\langle d, \nabla^2 g(x)[d]\rangle}{M_g} \\ \alpha &\leq \frac{3|2c_2 - 1||\langle d, \nabla^2 g(x)[d]\rangle|}{M_g},\end{aligned}$$

since $c_2 < 1/2$. Therefore, (6.19) is satisfied for all $\alpha \leq \min(\alpha_2(x), t_2(x))$. One readily checks that $t_2(x)$ and $\alpha_2(x)$ are positive. Therefore, there exists a nonempty interval $]0, \min(\alpha_2(x), t_2(x))]$ where the step-size satisfies the decrease condition (6.19) and defines a next iterate inside $\mathcal{C}$. Therefore the backtracking in Algorithm 9 returns a step-size $t$ satisfying $t \geq \tau_2 \min(\alpha_2(x), t_2(x))$ in a finite number of iterations. In addition, the decrease condition (6.19) gives

$$\begin{aligned}g(x) - g(x + td) &\geq -c_2 t^2\langle d, \nabla^2 g(x)[d]\rangle \\ &\geq -c_2\tau_2^2 \min(\alpha_2(x), t_2(x))^2\langle d, \nabla^2 g(x)[d]\rangle.\end{aligned}$$

□

**Remark 6.2.** *It may seem surprising that* $\underline{\alpha_1}$ *is a constant and* $\alpha_2(x)$ *depends on* $x$ *through the quadratic term* $|\langle d, \nabla^2 g(x)[d]\rangle|$. *This is a consequence of the way first-and second-order directions are defined. The step-size for a first-order step multiplies the gradient which can vary in norm whereas the step-size in second-order steps always multiplies a unit-norm direction.*

### 6.4.4 Worst-case global complexity

We are now in a position to give a formal version of our main result, the worst-case complexity of the Gradient-Eigenstep algorithm for problem (P).

**Theorem 6.17** (Complexity of Algorithm 7)**.** *Consider Problem* (P) *under A12, A13, A14, A15 and A16. Let* $0 < \varepsilon_1 \leq R/2$ *and let* $\underline{g}$ *be the lower bound of* $g$ *over the compact set* $\mathcal{C}$. *Algorithm 7 produces an iterate* $x_{N_1} \in \mathcal{C}$ *satisfying* $\|\nabla g(x_{N_1})\| \leq \varepsilon_1$ *with*

$$N_1 \leq \frac{g(x_0) - \underline{g}}{c_1\tau_1 \min(\underline{\alpha_1}, \underline{t_1})\varepsilon_1^2},$$

*where* $\underline{t_1} = \min_{x\in\mathcal{C}} t_1(x) > 0$.
*Furthermore if* $0 < \varepsilon_2 < \infty$*, Algorithm 7 also produces an iterate* $x_{N_2}$ *satisfying* $\|\nabla g(x_{N_2})\| \leq \varepsilon_1$ *and* $\lambda_{\min}(\nabla^2 g(x_{N_2})) \geq -\varepsilon_2$ *with*

$$N_2 \leq (g(x_0) - \underline{g}) \left[\min\left(c_1\tau_1 \min(\underline{\alpha_1}, \underline{t_1})\varepsilon_1^2, c_2\tau_2^2 \min\left(\min\left(\alpha_{02}, \frac{3|2c_2-1|\varepsilon_2}{M_g}\right), \underline{t_2}\right)^2 \varepsilon_2\right)\right]^{-1}, \tag{6.30}$$

*where* $\underline{t_2} = \min_{x\in\mathcal{C}} t_2(x) > 0$*. The iterate* $x_{N_1}$ *is an* $(\varepsilon_1, 2\varepsilon_1)$*-FOCP of* (P) *and* $x_{N_2}$ *is an* $(\varepsilon_1, 2\varepsilon_1, \varepsilon_2 + C\varepsilon_1)$*-SOCP of* (P)*, where* $C$ *is defined in Corollary 6.11.*

*Proof.* We first show that the constants $\underline{t_1} = \min_{x\in\mathcal{C}} t_1(x)$ and $\underline{t_2} = \min_{x\in\mathcal{C}} t_2(x)$ are positive. Recall from Equation (6.22) that

$$t_1(x) = \min\left(\sqrt{\frac{R}{2C_h}}\frac{1}{\|\nabla g(x)\|}, \frac{(2\beta\sigma_{\min}(\mathrm{D}h(x))^2 - \sigma_1(\mathrm{D}h(x))C_\lambda(x))R}{2C_h\|\nabla g(x)\|^2}, \frac{1}{2\beta\|\mathrm{D}h(x)\|_{\mathrm{op}}^2}\right)$$

One readily checks that $t_1(x) > 0$ for all $x \in \mathcal{C}$. The first term, $\sqrt{R/2C_h}/\|\nabla g(x)\|$ is positive since $\nabla g$ is continuous over $\mathcal{C}$ and $\mathcal{C}$ is compact. Using that $\beta > \bar{\beta}_1$(A16), the numerator of the second term is positive and bounded away from zero for all $x \in \mathcal{C}$. Using compactness of $\mathcal{C}$ and smoothness of $h$, the quantity $\|\mathrm{D}h(x)\|_{\mathrm{op}}$ is upper bounded over $\mathcal{C}$ and therefore $1/2\beta\|\mathrm{D}h(x)\|_{\mathrm{op}}^2$ is bounded away from zero over $\mathcal{C}$. We note that $t_1$ is a continuous function of $x$ which is positive for all $x$ in the compact set $\mathcal{C}$. Therefore, $\min_{x\in\mathcal{C}} t_1(x)$ is attained at a point in $\mathcal{C}$ and $\underline{t_1} > 0$. A similar process shows that $\underline{t_2} > 0$. The function $t_2(x) = \left(-\sigma_1(\mathrm{D}h(x)) + \sqrt{\sigma_1(\mathrm{D}h(x))^2 + 2C_hR}\right)/2C_h$ is continuous over $\mathcal{C}$. We also note that $t_2(x) > 0$ for all $x \in \mathcal{C}$ since the constants $R$ and $C_h$ are positive as a consequence of A12 and A14 respectively.

For every iteration $k$ where a first-order step is performed, one has $\|\nabla g(x_k)\| > \varepsilon_1$, while for second-order steps $\langle d, \nabla^2 g(x_k)[d]\rangle < -\varepsilon_2$. Therefore, Equation (6.27) gives the following decrease for first-order steps

$$\begin{aligned} g(x_k) - g(x_{k+1}) &\geq c_1\tau_1 \min(\underline{\alpha_1}, t_1(x_k))\|\nabla g(x_k)\|^2 \\ &\geq c_1\tau_1\min(\underline{\alpha_1}, \underline{t_1})\varepsilon_1^2, \end{aligned}$$

where $\underline{t_1} = \min_{x\in\mathcal{C}} t_1(x) > 0$, as shown above. The decrease for second-order steps follows from Equation (6.29), that is,

$$\begin{aligned} g(x_k) - g(x_{k+1}) &\geq -c_2\tau_2^2 \min(\alpha_2(x_k), t_2(x_k))^2\langle d, \nabla^2 g(x)[d]\rangle \\ &\geq c_2\tau_2^2 \min\left(\min\left(\alpha_{02}, \frac{3|2c_2-1||\langle d, \nabla^2 g(x_k)[d]\rangle|}{M_g}\right), t_2(x_k)\right)^2 \varepsilon_2 \\ &\geq c_2\tau_2^2 \min\left(\min\left(\alpha_{02}, \frac{3|2c_2-1|\varepsilon_2}{M_g}\right), \underline{t_2}\right)^2 \varepsilon_2, \end{aligned}$$

where $\underline{t_2} = \min_{x \in \mathcal{C}} t_2(x) > 0$, as shown above. Since $\mathcal{C}$ is compact (A13) and $g$ is continuous on $\mathcal{C}$, let $\underline{g} := \min_{x \in \mathcal{C}} g(x) > -\infty$. Consider the case $\varepsilon_2 < \infty$. For any $K \geq 0$, we have

$$\begin{aligned} g(x_0) - \underline{g} &\geq \sum_{k=0}^{K} g(x_k) - g(x_{k+1}) \qquad (6.31) \\ &\geq K \min\left( c_1 \tau_1 \min(\underline{\alpha_1}, \underline{t_1}) \varepsilon_1^2, c_2 \tau_2^2 \min\left( \min\left( \alpha_{02}, \frac{3|2c_2 - 1|\varepsilon_2}{M_g} \right), \underline{t_2} \right)^2 \varepsilon_2 \right). \end{aligned}$$

Given the definition of $N_2$, Equation (6.31) tells us that $K \leq N_2$. Hence, if more than $N_2$ iterations are performed, it must be that a point where $\|\nabla g(x)\| \leq \varepsilon_1$ and $\lambda_{\min}(\nabla^2 g(x)) \geq -\varepsilon_2$ has been encountered. In the case $\varepsilon_2 = \infty$, no second-order step is performed, which simplifies as follows

$$\begin{aligned} g(x_0) - \underline{g} &\geq \sum_{k=0}^{K} g(x_k) - g(x_{k+1}) \\ &\geq K c_1 \tau_1 \min(\underline{\alpha_1}, t_1(x_k)) \|\nabla g(x_k)\|^2 \\ &\geq K c_1 \tau_1 \min(\underline{\alpha_1}, \underline{t_1}) \varepsilon_1^2. \end{aligned}$$

The fact that $x_{N_1}$ and $x_{N_2}$ are respectively $(\varepsilon_1, 2\varepsilon_1)$-FOCP and $(\varepsilon_1, 2\varepsilon_1, \varepsilon_2 + C\varepsilon_1)$-SOCP of (P) follows from Proposition 6.6 and Proposition 6.10. $\square$

## 6.5 Estimating the penalty parameter

The previous section establishes convergence results under the assumption that the penalty parameter $\beta$ is fixed and large enough to satisfy A16. In practice, it is rarely possible to know whether this assumption is satisfied. Therefore, this section outlines a scheme which estimates a suitable value for $\beta$ without requiring A16. Let us define the value

$$B(x) := \max\{\beta_1(x), \beta_2(x), \beta_3(x)\}, \qquad (6.32)$$

where $\beta_i(x)$ for $i = 1, 2, 3$ are defined in Definition 6.5. Ensuring that $\beta > B(x_k)$ for every iterate $x_k$ of Algorithm 7 is sufficient for the algorithm to run smoothly and converge. Practically, it is possible to compute $B(x_k)$ at the current iterate while running the algorithm to increase $\beta$ if needed and ensure $\beta > B(x_k)$. However, changing $\beta$ throughout the algorithm would change the penalty function $g$, which invalidates the convergence analysis.

To address this issue, we propose the plateau scheme in Algorithm 10. It calls Algorithm 7 several times, each time for a fixed number of iterations and using a constant value of $\beta$. Each call with a constant $\beta$ is called a *plateau*. On each plateau, our analysis from previous sections is informative since $\beta$ is fixed, though possibly too small. With each new call, the value of $\beta$ and the length of the plateau (LP) are increased. The increase is designed to ensure that, after sufficiently many calls, $\beta$ and LP are large enough for Algorithm 7 to converge and return. This stops the plateau scheme.

**Algorithm 10** Plateau scheme

```
 1: Given: Functions $f$ and $h$, $0 \le \varepsilon_1 \le R/2$, $\varepsilon_2 \ge 0$, $\gamma > 1$, $x \in \mathcal{C}$, $\beta_0$ and $\mathrm{LP}_0$.
 2: for $\ell = 0, 1, 2, \dots$ do
 3:     $x$ = Gradient-Eigenstep($x, \beta_\ell$) with optional stopping criterion set to $B(x) \ge \beta_\ell$
        and $k > \mathrm{LP}_\ell$                                  ▷ This is a call to Algorithm 7
 4:     if $\|\nabla g(x)\| \le \varepsilon_1$ and $\lambda_{\min}(\nabla^2 g(x)) \ge -\varepsilon_2$ then
 5:         return
 6:     end if
 7:     if Algorithm 7 stopped because $B(x) \ge \beta_\ell$ then
 8:         Set $B = B(x)$
 9:         Set $\mathrm{LP}_{\ell+1} = (\gamma B/\beta_\ell)^4 \mathrm{LP}_\ell$ then $\beta_{\ell+1} = \gamma B$
10:     else
11:         $\beta_{\ell+1} = \gamma \beta_\ell$ and $\mathrm{LP}_{\ell+1} = \gamma^4 \mathrm{LP}_\ell$
12:     end if
13: end for
```

We proceed to show that the plateau scheme (Algorithm 10) converges in a finite number of iterations. Moreover, the complexity with respect to $\varepsilon_1, \varepsilon_2$ is of the same order as Algorithm 7 up to logarithmic factors. Let $K(\beta)$ be the right-hand side of Equation (6.30) as a function of $\beta$,

$$K(\beta) = (g(x_0) - \underline{g}) \left[ \min\left( c_1 \tau_1 \min(\underline{\alpha_1}, \underline{t_1}) \varepsilon_1^2, c_2 \tau_2^2 \min\left( \min\left( \alpha_{02}, \frac{3|2c_2 - 1|\varepsilon_2}{M_g} \right), \underline{t_2} \right)^2 \varepsilon_2 \right) \right]^{-1}.$$

This gives the worst-case number of iterations required for Algorithm 7 to achieve our target tolerances using a constant value of $\beta$, according to Theorem 6.17. We show that $K(\beta)$ is upper bounded by a cubic function of $\beta$ for $\beta \ge \bar{\beta}$.

**Theorem 6.18.** *For $\varepsilon_2 < 1$, there exists $C_3 > 0$, independent of $\varepsilon_1$, $\varepsilon_2$ and $\beta$ such that for all $\beta > \overline{\beta}$, we have*

$$K(\beta) \le C_3 \max\left(\varepsilon_1^{-2}, \varepsilon_2^{-3}\right) \beta^3,$$

*where $\overline{\beta}$ is defined in Equation* (6.21).

*Proof.* Firstly, we find the dependence on $\beta$ of the Lipschitz constants of the gradient and Hessian of $g$, namely, $L_g$ and $M_g$. From

$$g(x) = f(x) - \langle h(x), \lambda(x) \rangle + \beta \|h(x)\|^2$$

and

$$\nabla g(x) = \nabla f(x) - \mathrm{D}h(x)^*[\lambda(x)] + 2\beta \mathrm{D}h(x)^*[h(x)] - \mathrm{D}\lambda(x)^*[h(x)],$$

it is clear that $\nabla g(x)$ and $\nabla^2 g(x)$ are affine in $\beta$, hence $L_g(\beta)$, $M_g(\beta)$ and $\|\nabla g(x)\|$ are affine functions of $\beta$ as well. Therefore, it is possible to find a constant $c$ such that $L_g \le c\beta$ for all $\beta > \overline{\beta}$. We apply that reasoning to all functions affine in $\beta$, since we are interested in their behaviour for all $\beta > \overline{\beta}$. We notice that $g(x_0)$ is also an affine function of $\beta$. The value $\underline{g}$ does not depend on $\beta$. The formula $K(\beta)$ depends on $\beta$ through the values $g(x_0), \underline{\alpha_1}, \underline{t_1}, L_g$ and $M_g$. Indeed,

$$\underline{\alpha_1} = \min\left(\alpha_{01}, \frac{2(1-c_1)}{L_g}\right)$$
$$\underline{t_1} = \min_{x\in\mathcal{C}} \min\left(\sqrt{\frac{R}{2C_h}}\frac{1}{\|\nabla g(x)\|}, \frac{(2\beta\sigma_{\min}(\mathrm{D}h(x))^2 - \sigma_1(\mathrm{D}h(x))\overline{C_\lambda})R}{2C_h\|\nabla g(x)\|^2}, \frac{1}{2\beta\|\mathrm{D}h(x)\|_{\mathrm{op}}^2}\right)$$

Examination shows that $1/\underline{t_1}$ and $1/\underline{\alpha_1}$ are bounded by affine functions of $\beta$. That is, there exists constants $b_1, b_2$, such that, for all $\beta > \overline{\beta}$, we have $1/\underline{\alpha_1} \le b_1\beta$ and $1/\underline{t_1} \le b_2\beta$. Define

$$\underline{\alpha_2} = \min\left(\alpha_{02}, \frac{3|2c_2-1|\varepsilon_2}{M_g}\right).$$

Using $\varepsilon_2 < 1$, there exists $b_3$ such that,

$$\begin{aligned} 1/\underline{\alpha_2} &= \max\left(1/\alpha_{02}, \frac{M_g}{3|2c_2-1|\varepsilon_2}\right) \\ &\le \varepsilon_2^{-1}\max\left(1/\alpha_{02}, \frac{M_g}{3|2c_2-1|}\right) \\ &\le b_3\beta\varepsilon_2^{-1}. \end{aligned}$$

Finally,

$$\underline{t_2} = \min_{x\in\mathcal{C}}\left(-\sigma_1(\mathrm{D}h(x)) + \sqrt{\sigma_1(\mathrm{D}h(x))^2 + 2C_hR}\right)/2C_h$$

is independent of $\beta$. This implies the existence of a constant $b_4$ such that $1/\underline{t_2} \le b_4$. In addition, for any $a, b > 0$, we have $1/\min(a,b) = \max(1/a, 1/b)$. For all $\beta > \overline{\beta}$, this gives

$$\begin{aligned}
K(\beta) &= (g(x_0) - \underline{g}) \max\left(\frac{1}{c_1\tau_1\varepsilon_1^2 \min(\underline{\alpha_1}, \underline{t_1})}, \frac{1}{c_2\tau_2^2\varepsilon_2 \min\left(\underline{\alpha_2}, \underline{t_2}\right)^2}\right) \\
&= (g(x_0) - \underline{g}) \max\left(\frac{\max(1/\underline{\alpha_1}, 1/\underline{t_1})}{c_1\tau_1\varepsilon_1^2}, \frac{\max\left(1/\underline{\alpha_2}, 1/\underline{t_2}\right)^2}{c_2\tau_2^2\varepsilon_2}\right) \\
&\leq b_0\beta \max\left(\frac{\max(b_1\beta, b_2\beta)}{c_1\tau_1\varepsilon_1^2}, \frac{\max\left(b_3\beta\varepsilon_2^{-1}, b_4\right)^2}{c_2\tau_2^2\varepsilon_2}\right) \\
&\leq b_0\beta \max\left(\varepsilon_1^{-2}, \varepsilon_2^{-3}\right) \max\left(\frac{\max(b_1\beta, b_2\beta)}{c_1\tau_1}, \frac{\max\left(b_3\beta, b_4\right)^2}{c_2\tau_2^2}\right).
\end{aligned}$$

We conclude that there exists $C_3 > 0$, independent of $\varepsilon_1$, $\varepsilon_2$ and $\beta$ such that, for all $\beta \geq \overline{\beta}$, we have

$$K(\beta) \leq C_3 \max\left(\varepsilon_1^{-2}, \varepsilon_2^{-3}\right) \beta^3. \qquad \square$$

We now state the main result of this section, which claims that the plateau scheme in Algorithm 10 returns an $(\varepsilon_1, 2\varepsilon_1, \varepsilon_2 + C\varepsilon_1)$-SOSP point with the same global complexity rate as Algorithm 7 up to logarithmic factors.

**Theorem 6.19.** *Under A12, A13, A14 and A15, Algorithm 10 returns an $(\varepsilon_1, 2\varepsilon_1, \varepsilon_2 + C\varepsilon_1)$-SOSP in at most $\mathcal{O}\left(\max\left\{\varepsilon_1^{-2}, \varepsilon_2^{-3}\right\} \max\left\{\log_\gamma \varepsilon_1^{-2}, \log_\gamma \varepsilon_2^{-3}\right\}\right)$ iterations of Algorithm 7, where $C$ is defined in Corollary 6.11.*

*Proof.* For $C_3$ provided by Theorem 6.18, define the value $C_2 = C_3 \max\left(\varepsilon_1^{-2}, \varepsilon_2^{-3}\right)$ which is independent of $\beta$. Algorithm 10 is guaranteed to converge when $\beta_\ell > \overline{\beta}$ and $\mathrm{LP}_\ell \geq K(\beta_\ell)$, that is, when $\beta_\ell$ is large enough to satisfy A16 and the length of the plateau is large enough to allow Algorithm 7 to converge in a worst-case number of iterations. From Algorithm 10, it is clear that $\beta_\ell \geq \beta_0\gamma^\ell$. Therefore, if $\ell > \lceil \log_\gamma(\overline{\beta}/\beta_0) \rceil$, it follows that

$$\beta_\ell \geq \beta_0\gamma^\ell > \beta_0\gamma^{\log_\gamma(\overline{\beta}/\beta_0)} = \overline{\beta}, \tag{6.33}$$

which indicates for which $\ell$ large enough the condition $\beta_\ell > \overline{\beta}$ is met. Regarding the length of the plateaus, it is clear that $\mathrm{LP}_\ell \geq \mathrm{LP}_0\gamma^{4\ell}$. One can also infer from Algorithm 10 that $\beta_\ell \leq \max(\beta_0, \overline{\beta})\gamma^\ell$. Indeed if $\beta_0 > \overline{\beta}$, then $\beta_\ell = \beta_0\gamma^\ell$ since lines 8 and 9 of Algorithm 10 are not executed. When $\overline{\beta} \geq \beta_0$, $\beta_\ell \leq \overline{\beta}\gamma^\ell$. Using that $K(\beta) \leq C_2\beta^3$ for all $\beta > \overline{\beta}$, we enforce

$$\mathrm{LP}_0\gamma^{4\ell} \geq C_2\left(\max(\beta_0, \overline{\beta})\gamma^\ell\right)^3, \tag{6.34}$$

from which it follows that

$$\mathrm{LP}_\ell \geq \mathrm{LP}_0\gamma^{4\ell} \geq C_2\left(\max(\beta_0, \overline{\beta})\gamma^\ell\right)^3 \geq C_2\beta_\ell^3 \geq K(\beta_\ell).$$

Equation (6.34) simplifies to

$$\gamma^\ell \geq \frac{C_2 \max(\beta_0, \overline{\beta})^3}{LP_0}$$

or

$$\ell \geq \left\lceil \log_\gamma \left( \frac{C_2 \max(\beta_0, \overline{\beta})^3}{\mathrm{LP}_0} \right) \right\rceil .$$

In conclusion, the maximum number of plateaus is

$$\ell^* := \max \left( \left\lceil \log_\gamma \left( \frac{C_2 \max(\beta_0, \overline{\beta})^3}{\mathrm{LP}_0} \right) \right\rceil , \lceil \log_\gamma (\overline{\beta}/\beta_0) \rceil + 1 \right) .$$

The maximum value of $\beta$ is

$$\beta^* = \max(\beta_0, \overline{\beta})\gamma^{\ell^*}.$$

On any plateau the maximum number of iterations of Gradient-Eigenstep is $K(\beta^*)$, because for $\beta_\ell = \beta^*$, the plateau is long enough to allow convergence and $\beta^* \geq \overline{\beta}$. The worst-case number of Gradient-Eigenstep iterations is at most $K(\beta^*)\ell^*$ with

$$K(\beta^*)\ell^* \leq C_3 \max\left( \frac{1}{\varepsilon_1^2}, \frac{1}{\varepsilon_2^3} \right) (\beta^*)^3 \max \left( \left\lceil \log_\gamma \left( \frac{C_3 \max\left(1/\varepsilon_1^2, 1/\varepsilon_2^3\right) \max(\beta_0, \overline{\beta})^3}{\mathrm{LP}_0} \right) \right\rceil , \lceil \log_\gamma (\overline{\beta}/\beta_0) \rceil + 1 \right). \qquad \square$$

## 6.6 Conclusions

In this chapter, we consider optimization problems with smooth equality constraints. Under a regularity condition on the derivative of the constraints (A12), we propose a definition of approximate criticality for problem (P), which has a natural geometric interpretation and extends Riemannian optimality conditions to points near the feasible set. To find such critical points, we consider a smooth penalty function (Fletcher's augmented Lagrangian). We establish connections between the approximate critical points of Fletcher's augmented Lagrangian and the approximate critical points of the original constrained problem (P). We present Algorithm 7, which is shown to reach approximate second-order critical points of (P) in at most $\mathcal{O}(\varepsilon^{-3})$ iterations, the optimal rate in this setting. The only other work to date which achieved this optimal rate for an infeasible method is (Cartis et al., 2019), which uses a different notion of approximate criticality.

The main drawback of our approach, is the necessity to identify a set $\mathcal{C}$, where the differential of the constraint is nonsingular, in order to run the algorithm. Similar smoothness assumptions are made in related works which provide a worst-case complexity analysis (Cifuentes and Moitra, 2019; Xie and Wright, 2021).

Fletcher's augmented Lagrangian may be considered impractical in view of the linear system that must be solved at each iteration to evaluate the multipliers $\lambda(x)$. However, recent works show that it can still lead to the design of efficient algorithms and our contribution further reinforces the theoretical appeal of Fletcher's augmented Lagrangian. As a result, directions of future research emerge. Consider a smooth function $\hat{\lambda} \colon \mathcal{E} \to \mathbb{R}^m$ which coincides on $\mathcal{M}$ with the function $\lambda(x) = (\mathrm{D}h(x)^*)^\dagger [\nabla f(x)]$. This choice of multipliers defines a corresponding function $\hat{g}(x) = \mathcal{L}_\beta(x, \hat{\lambda}(x))$, a variant of the penalty $g$. Recent works (Gao et al., 2019; Xiao et al., 2020; Xiao and Liu, 2021) show that minimizing the function $\hat{g}$ yields efficient algorithms for a particular choice of $\hat{\lambda}$ on the Stiefel manifold. Is there a way to generalize this concept to other manifolds ? What theoretical guarantees can we hope to keep by using $\hat{\lambda}(x)$ instead of $\lambda(x)$ ? Exploring this could yield more practical Lagrangian-based infeasible methods to solve constrained optimization problems with underlying smoothness.

# Bibliography

Ablin, P. and Peyré, G. (2021). Fast and accurate optimization on the orthogonal manifold without retraction. *arXiv preprint arXiv:2102.07432*.

Abraham, R., Marsden, J. E., and Ratiu, T. (2012). *Manifolds, tensor analysis, and applications*, volume 75. Springer Science & Business Media.

Absil, P.-A., Baker, C. G., and Gallivan, K. A. (2007). Trust-region methods on Riemannian manifolds. *Found. Comput. Math.*, 7(3):303–330.

Absil, P.-A., Mahony, R., and Sepulchre, R. (2004). Riemannian geometry of grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematica*, 80(2):199–220.

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

Absil, P.-A. and Malick, J. (2012). Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158.

Adamson, A. and Alexa, M. (2003). Approximating and intersecting surfaces from points. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 230–239.

Agarwal, N., Boumal, N., Bullins, B., and Cartis, C. (2018). Adaptive regularization with cubics on manifolds with a first-order analysis. *arXiv preprint arXiv:1806.00065*.

Andreani, R., Birgin, E. G., Martínez, J. M., and Schuverdt, M. L. (2008). On augmented lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18(4):1286–1309.

Andreani, R., Martínez, J., and Schuverdt, M. (2007). On second-order optimality conditions for nonlinear programming. *Optimization*, 56(5-6):529–542.

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700.

Bai, Y., Duchi, J., and Mei, S. (2019). Proximal algorithms for constrained composite optimization, with applications to solving low-rank sdps. *arXiv preprint arXiv:1903.00184*.

Bai, Y. and Mei, S. (2018). Analysis of sequential quadratic programming through the lens of riemannian optimization. *arXiv preprint arXiv:1805.08756*.

Bajaj, C. L., Bernardini, F., and Xu, G. (1995). Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 109–118.

Balzano, L., Nowak, R., and Recht, B. (2010). Online identification and tracking of subspaces from highly incomplete information. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 704–711. IEEE.

Bandeira, A. S., Boumal, N., and Singer, A. (2017). Tightness of the maximum likelihood semidefinite relaxation for angular synchronization. *Mathematical Programming*, 163(1-2):145–167.

Bellekens, B., Spruyt, V., Berkvens, R., and Weyn, M. (2014). A survey of rigid 3d pointcloud registration algorithms. In *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*, pages 8–13.

Bertsekas, D. P. (1982). *Constrained Optimization and Lagrange Multiplier Methods.* Elsevier.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics.

Bhojanapalli, S., Neyshabur, B., and Srebro, N. (2016). Global optimality of local search for low rank matrix recovery. In *Advances in Neural Information Processing Systems*, pages 3873–3881.

Birgin, E. G., Gardenghi, J., Martínez, J. M., Santos, S. A., and Toint, P. L. (2017). Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. *Mathematical Programming*, 163(1-2):359–368.

Birgin, E. G., Haeser, G., and Ramos, A. (2018). Augmented lagrangians with constrained subproblems and convergence to second-order stationary points. *Computational Optimization and Applications*, 69(1):51–75.

Birgin, E. G. and Martnez, J. M. (2019). Complexity and performance of an augmented lagrangian algorithm.

Bishop, C. M. (2006). Pattern recognition. *Machine learning*, 128(9).

Biswas, P., Liang, T.-C., Toh, K.-C., Ye, Y., and Wang, T.-C. (2006). Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE transactions on automation science and engineering*, 3(4):360–371.

Biswas, P., Toh, K.-C., and Ye, Y. (2008). A distributed sdp approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. *SIAM Journal on Scientific Computing*, 30(3):1251–1277.

Bolte, J., Sabach, S., and Teboulle, M. (2013). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494.

Boumal, N. (2014). *Optimization and estimation on manifolds.* PhD thesis, Universite Catholique de Louvain.

Boumal, N. (2016). Nonconvex phase synchronization. *SIAM Journal on Optimization*, 26(4):2355–2377.

Boumal, N. (2020). An introduction to optimization on smooth manifolds. Available online.

Boumal, N. and Absil, P.-a. (2011). Rtrmc: A riemannian trust-region method for low-rank matrix completion. In *Advances in neural information processing systems*, pages 406–414.

Boumal, N. and Absil, P.-A. (2015). Low-rank matrix completion via preconditioned optimization on the grassmann manifold. *Linear Algebra and its Applications*, 475:200–239.

Boumal, N., Absil, P.-A., and Cartis, C. (2019). Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33.

Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. (2014). Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459.

Boumal, N., Voroninski, V., and Bandeira, A. S. (2020). Deterministic guarantees for burer-monteiro factorizations of smooth semidefinite programs. *Communications on Pure and Applied Mathematics*, 73(3):581–608.

Cambier, L. and Absil, P.-A. (2016). Robust low-rank matrix completion by riemannian optimization. *SIAM Journal on Scientific Computing*, 38(5):S440–S460.

Candes, E. J. and Plan, Y. (2011). Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *IEEE Transactions on Information Theory*, 57(4):2342–2359.

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080.

Cartis, C., Gould, N., and Toint, P. (2012). Complexity bounds for second-order optimality in unconstrained optimization. *Journal of Complexity*, 28(1):93–108.

Cartis, C., Gould, N. I., and Toint, P. L. (2010). On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems. *Siam journal on optimization*, 20(6):2833–2852.

Cartis, C., Gould, N. I., and Toint, P. L. (2011). On the evaluation complexity of composite function minimization with applications to nonconvex nonlinear programming. *SIAM Journal on Optimization*, 21(4):1721–1739.

Cartis, C., Gould, N. I., and Toint, P. L. (2015). On the evaluation complexity of constrained nonlinear least-squares and general constrained nonlinear optimization using second-order methods. *SIAM Journal on Numerical Analysis*, 53(2):836–851.

Cartis, C., Gould, N. I., and Toint, P. L. (2019). Optimality of orders one to three and beyond: characterization and evaluation complexity in constrained nonconvex optimization. *Journal of Complexity*, 53:68–94.

Cartis, C., Gould, N. I. M., and Toint, P. L. (2022). Evaluation complexity of algorithms for nonconvex optimization. *MOS-SIAM Series on Optimization.* (forthcoming).

Cayton, L. (2005). Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 12(1-17):1.

Chaudhury, K. N., Khoo, Y., and Singer, A. (2015). Global registration of multiple point clouds using semidefinite programming. *SIAM Journal on Optimization*, 25(1):468–501.

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155.

Chui, H. and Rangarajan, A. (2003). A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141.

Cifuentes, D. (2019). Burer-monteiro guarantees for general semidefinite programs.

Cifuentes, D. and Moitra, A. (2019). Polynomial time guarantees for the burer-monteiro method.

Conn, A. R., Gould, N. I., and Toint, P. (1991). A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572.

Conn, A. R., Gould, N. I., and Toint, P. L. (2000). *Trust region methods*, volume 1. Siam.

Cox, D., Little, J., O'Shea, D., and Sweedler, M. (1994). Ideals, varieties, and algorithms. *American Mathematical Monthly*, 101(6):582–586.

Cucuringu, M., Lipman, Y., and Singer, A. (2012a). Sensor network localization by eigenvector synchronization over the euclidean group. *ACM Transactions on Sensor Networks (TOSN)*, 8(3):1–42.

Cucuringu, M., Singer, A., and Cowburn, D. (2012b). Eigenvector synchronization, graph rigidity and the molecule problem. *Information and Inference: A Journal of the IMA*, 1(1):21–67.

Dai, W., Kerman, E., and Milenkovic, O. (2012). A geometric approach to low-rank matrix completion. *IEEE Transactions on Information Theory*, 58(1):237–247.

Davenport, M. A. and Romberg, J. (2016). An overview of low-rank matrix recovery from incomplete observations. *arXiv preprint arXiv:1601.06422*.

de Carvalho Bento, G., da Cruz Neto, J. X., and Oliveira, P. R. (2016). A new approach to the proximal point method: convergence on general riemannian manifolds. *Journal of Optimization Theory and Applications*, 168(3):743–755.

Demmel, J., Gu, M., Eisenstat, S., Slapničar, I., Veselić, K., and Drmač, Z. (1999). Computing the singular value decomposition with high relative accuracy. *Linear Algebra and its Applications*, 299(1-3):21–80.

Deutsch, S., Ortega, A., and Medioni, G. (2018). Robust denoising of piece-wise smooth manifolds. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2786–2790. IEEE.

Dhillon, I. S., Heath, J. R., Strohmer, T., and Tropp, J. A. (2008). Constructing packings in grassmannian manifolds via alternating projection. *Experimental mathematics*, 17(1):9–35.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Edelman, A., Arias, T. A., and Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353.

Eftekhari, A., Ongie, G., Balzano, L., and Wakin, M. B. (2019). Streaming principal component analysis from incomplete data. *Journal of Machine Learning Research*, 20(86):1–62.

Eldar, Y. C. and Mishali, M. (2009). Robust recovery of signals from a structured union of subspaces. *IEEE Transactions on Information Theory*, 55(11):5302–5316.

Elhamifar, E. and Vidal, R. (2013). Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781.

Eriksson, B., Balzano, L., and Nowak, R. (2012). High-rank matrix completion.

Fan, J. and Cheng, J. (2018). Matrix completion by deep matrix factorization. *Neural Networks*, 98:34–41.

Fan, J. and Chow, T. W. (2018). Non-linear matrix completion. *Pattern Recognition*, 77:378–394.

Fan, J. and Udell, M. (2019). Online high rank matrix completion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fan, J., Yang, C., and Udell, M. (2020). Robust non-linear matrix factorization for dictionary learning, denoising, and clustering.

Fan, J., Zhang, Y., and Udell, M. (2019). Polynomial matrix completion for missing data imputation and transductive learning.

Fan, J., Zhao, M., and Chow, T. W. S. (2018). Matrix completion via sparse factorization solved by accelerated proximal alternating linearized minimization. *IEEE Transactions on Big Data*, pages 1–1.

Fang, X. and Toh, K.-C. (2013). Using a distributed sdp approach to solve simulated protein molecular conformation problems. In *Distance Geometry*, pages 351–376. Springer.

Fletcher, R. (1970). A class of methods for nonlinear programming with termination and convergence properties. *Integer and nonlinear programming*, pages 157–173.

Forsyth, D. and Ponce, J. (2011). *Computer vision: A modern approach.* Prentice hall.

Gao, B., Liu, X., and Yuan, Y.-x. (2019). Parallelizable algorithms for optimization problems with orthogonality constraints. *SIAM Journal on Scientific Computing*, 41(3):A1949–A1983.

Ge, R., Lee, J. D., and Ma, T. (2016). Matrix completion has no spurious local minimum. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2973–2981. Curran Associates, Inc.

Ghojogh, B. and Crowley, M. (2019). The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial.

Gold, S., Rangarajan, A., Lu, C.-P., Pappu, S., and Mjolsness, E. (1998). New algorithms for 2d and 3d point matching: Pose estimation and correspondence. *Pattern recognition*, 31(8):1019–1031.

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations.* The Johns Hopkins University Press, third edition.

Gong, D., Sha, F., and Medioni, G. (2010). Locally linear denoising on image manifolds. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 265–272.

Goshtasby, A. A. (2005). *2-D and 3-D image registration: for medical, remote sensing, and industrial applications.* John Wiley & Sons.

Goyens, F., Cartis, C., and Eftekhari, A. (2019). Nonlinear matrix recovery. *NeurIPS Workshop Beyond First Order Methods in Machine Learning.*

Goyens, F., Cartis, C., and Eftekhari, A. (2021). Nonlinear matrix recovery using optimization on the grassmann manifold. *arXiv preprint arXiv:2109.06095.*

Goyens, F., Chretien, S., and Cartis, C. (2020). Smoothing of point clouds using riemannian optimization. *ICML Workshop Beyond first order methods in machine learning.*

Grapiglia, G. N. and xiang Yuan, Y. (2019). On the complexity of an augmented lagrangian method for nonconvex optimization.

Grapiglia, G. N. and Yuan, Y.-x. (2021). On the complexity of an augmented lagrangian method for nonconvex optimization. *IMA Journal of Numerical Analysis*, 41(2):1508–1530.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Hao, Z., Ma, S., Chen, H., and Liu, J. (2021). Dataset denoising based on manifold assumption. 2021:1–14.

Harvey, N. J., Karger, D. R., and Yekhanin, S. (2006). The complexity of matrix completion. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1103–1111. Society for Industrial and Applied Mathematics.

Hauser, R. A., Eftekhari, A., and Matzinger, H. F. (2018). Pca by determinant optimisation has no spurious local optima. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1504–1511.

Hein, M. and Maier, M. (2007). Manifold denoising. In *Advances in neural information processing systems*, pages 561–568.

Hill, D. L., Hawkes, D. J., Crossman, J., Gleeson, M., Cox, T., Bracey, E., Strong, A., and Graves, P. (1991). Registration of mr and ct images for skull base surgery using point-like anatomical features. *The British journal of radiology*, 64(767):1030–1035.

Hoffmann, H. (2007). Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874.

Horn, R. and Johnson, C. (1991). Topics in matrix analysis. *Cambridge University Press.*

Hosseini, S. (2015). Convergence of nonsmooth descent methods via kurdyka-lojasiewicz inequality on riemannian manifolds. *Hausdorff Center for Mathematics and Institute for Numerical Simulation, University of Bonn (2015,(INS Preprint No. 1523)).*

Huang, X., Mei, G., Zhang, J., and Abbas, R. (2021). A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690.*

Jain, P., Meka, R., and Dhillon, I. S. (2010). Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945.

Jia, X., Kanzow, C., Mehlitz, P., and Wachsmuth, G. (2021). An augmented lagrangian method for optimization problems with structured geometric constraints.

Jorge Nocedal, S. J. W. (1999). *Numerical Optimization*. Springer.

Keshavan, R. H., Montanari, A., and Oh, S. (2010). Matrix completion from a few entries. *IEEE transactions on information theory*, 56(6):2980–2998.

Keshavan, R. H. and Oh, S. (2009). A gradient descent algorithm on the grassman manifold for matrix completion. *arXiv preprint arXiv:0910.5260*.

Kohler, J. M. and Lucchi, A. (2017). Sub-sampled cubic regularization for non-convex optimization. In *International Conference on Machine Learning*, pages 1895–1904. PMLR.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. 42(8):30–37.

Krishnan, S., Lee, P. Y., Moore, J. B., Venkatasubramanian, S., et al. (2005). Global registration of multiple 3d point sets via optimization-on-a-manifold. In *Symposium on Geometry Processing*, pages 187–196.

Lee, J. M. (2018). *Introduction to Riemannian manifolds*. Springer.

Lerman, G., Zhang, T., et al. (2011). Robust recovery of multiple subspaces by geometric lp minimization. *The Annals of Statistics*, 39(5):2686–2715.

Li, Q. and Tang, G. (2017). The nonconvex geometry of low-rank matrix optimizations with general objective functions. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1235–1239.

Liu, C. and Boumal, N. (2020). Simple algorithms for optimization on Riemannian manifolds with constraints. *Applied Mathematics and Optimization*, 82(3):949–981.

Low, K.-L. (2004). Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10):1–3.

Mirsky, L. (1960). Symmetric gauge functions and unitarily invariant norms. *The quarterly journal of mathematics*, 11(1):50–59.

Mohan, K. and Fazel, M. (2012). Iterative reweighted algorithms for matrix rank minimization. *Journal of Machine Learning Research*, 13(Nov):3441–3473.

Myronenko, A. and Song, X. (2010). Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275.

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization*. Springer US.

Nesterov, Y. (2018). Lectures on convex optimization.

Nguyen, L. T., Kim, J., and Shim, B. (2019). Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Ongie, G., Pimentel-Alarcón, D., Balzano, L., Willett, R., and Nowak, R. D. (2021). Tensor methods for nonlinear matrix completion. *SIAM Journal on Mathematics of Data Science*, 3(1):253–279.

Ongie, G., Willett, R., Nowak, R. D., and Balzano, L. (2017). Algebraic variety models for high-rank matrix completion. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2691–2700, International Convention Centre, Sydney, Australia. PMLR.

Polyak, R. A. (2009). On the local quadratic convergence of the primal–dual augmented lagrangian method. *Optimization Methods & Software*, 24(3):369–379.

Pottmann, H., Leopoldseder, S., and Hofer, M. (2004). Registration without icp. *Computer Vision and Image Understanding*, 95(1):54–71.

Rahimi, A., Recht, B., et al. (2007). Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Rapcsák, T. (1997). *Smooth Nonlinear Optimization in R n*. Springer US.

Recht, B. (2011). A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(Dec):3413–3430.

Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501.

Recht, B. and Ré, C. (2013). Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226.

Royer, C. W., ONeill, M., and Wright, S. J. (2020). A newton-cg algorithm with complexity guarantees for smooth unconstrained optimization. *Mathematical Programming*, 180(1):451–488.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE.

Sahin, M. F., Eftekhari, A., Alacaoglu, A., Latorre, F., and Cevher, V. (2019). An inexact augmented lagrangian framework for nonconvex optimization with nonlinear constraints. *arXiv preprint arXiv:1906.11357*.

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer.

Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Sharp, G. C., Lee, S. W., and Wehe, D. K. (2004). Multiview registration of 3d scenes by minimizing error between coordinate frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1037–1050.

Singer, A. (2006). From graph to manifold laplacian: The convergence rate. *Applied and Computational Harmonic Analysis*, 21(1):128–134.

Stein, C. M. (1981). Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, pages 1135–1151.

Stewart, G. W. (1998). Perturbation theory for the singular value decomposition. Technical report.

Studholme, C., Hill, D. L., and Hawkes, D. J. (1995). Automated 3d registration of truncated mr and ct images of the head. In *BMVC*, volume 95, pages 27–36. Citeseer.

Subrahmonia, J., Cooper, D. B., and Keren, D. (1996). Practical reliable bayesian recognition of 2d and 3d objects using implicit polynomials and algebraic invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):505–519.

Sun, J., Qu, Q., and Wright, J. (2015). When are nonconvex problems not scary? *arXiv preprint arXiv:1510.06096*.

Sun, J., Qu, Q., and Wright, J. (2017). Complete dictionary recovery over the sphere i: Overview and the geometric picture. *IEEE Transactions on Information Theory*, 63(2):853–884.

Sun, J., Qu, Q., and Wright, J. (2018). A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198.

Tam, G. K., Cheng, Z.-Q., Lai, Y.-K., Langbein, F. C., Liu, Y., Marshall, D., Martin, R. R., Sun, X.-F., and Rosin, P. L. (2013). Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–1217.

Tanner, J. and Wei, K. (2013). Normalized iterative hard thresholding for matrix completion. *SIAM Journal on Scientific Computing*, 35(5):S104–S125.

Toh, K.-C. and Yun, S. (2010). An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of optimization*, 6(615-640):15.

Townsend, J., Koep, N., and Weichwald, S. (2016). Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *arXiv preprint arXiv:1603.03236*.

Tsin, Y. and Kanade, T. (2004). A correlation-based approach to robust point set registration. In *European conference on computer vision*, pages 558–569. Springer.

Tzeneva, T. (2011). Global alignment of multiple 3-d scans using eigevector synchronization. *Senior Thesis, Princeton University (supervised by S. Rusinkiewicz and A. Singer)*.

Uschmajew, A. and Vandereycken, B. (2018). On critical points of quadratic low-rank matrix optimization problems. Tech. report (submitted).

Van Kaick, O., Zhang, H., Hamarneh, G., and Cohen-Or, D. (2011). A survey on shape correspondence. In *Computer Graphics Forum*, volume 30, pages 1681–1707. Wiley Online Library.

Vandereycken, B. (2013). Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236.

Wang, B. and Tu, Z. (2013). Sparse subspace denoising for image manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 468–475.

Wen, Z., Yin, W., and Zhang, Y. (2012). Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361.

Weyrich, T., Pauly, M., Keiser, R., Heinzle, S., Scandella, S., and Gross, M. H. (2004). Post-processing of scanned 3d surface data. *SPBG*, 4:85–94.

Williams, J. A. and Bennamoun, M. (2000). Simultaneous registration of multiple point sets using orthonormal matrices. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, volume 4, pages 2199–2202. IEEE.

Xiao, N. and Liu, X. (2021). Solving optimization problems over the stiefel manifold by smooth exact penalty function.

Xiao, N., Liu, X., and Yuan, Y.-x. (2020). A class of smooth exact penalty function methods for optimization problems with orthogonality constraints. *Optimization Methods and Software*, pages 1–37.

Xie, Y. and Wright, S. J. (2019). Complexity of proximal augmented lagrangian for nonconvex optimization with nonlinear equality constraints.

Xie, Y. and Wright, S. J. (2021). Complexity of proximal augmented lagrangian for nonconvex optimization with nonlinear equality constraints. *Journal of Scientific Computing*, 86(3):1–30.

Xu, P., Roosta, F., and Mahoney, M. W. (2020). Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 184(1):35–70.

Xu, R. and Wunsch, D. (2008). *Clustering*, volume 10. John Wiley & Sons.

Zwicker, M., Pauly, M., Knoll, O., and Gross, M. (2002). Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics (TOG)*, 21(3):322–329.

# Appendix A

# Proofs

## A.1 Proofs for Chapter 4

**Proposition A.1.** *For the monomial kernel* $\mathrm{K}_d$*, the Euclidean gradient of the cost function* (4.3) *is given by*

$$\nabla_X f(X,W) = 2dX\left(\mathrm{K}_{d-1}(X)\odot \mathrm{P}_{W_\perp}\right) \qquad \text{and} \qquad \nabla_W f(X,W) = -2\mathrm{K}_d(X)W.$$

*For* $(\Delta_X,\Delta_W)\in\mathbb{R}^{n\times s}\times\mathbb{R}^{s\times r}$*, the application of the Hessian is given by*

$$\nabla^2 f(X,W)\begin{pmatrix}\Delta_X\\ \Delta_W\end{pmatrix} = \begin{pmatrix}\nabla_X^2 f(X,W)[\Delta_X] + \nabla_W\nabla_X f(X,W)[\nabla_W]\\ \nabla_X\nabla_W f(X,W)[\Delta_X] + \nabla_W^2 f(X,W)[\Delta_W]\end{pmatrix}.$$

*where*

$$\begin{aligned}
\nabla_W^2 f(X,W)[\delta_W] &= -2\mathrm{K}_d(X)\delta_W\\
\nabla_X^2 f(X,W)[\Delta_X] &= 2d(d-1)X\left(\mathrm{K}_{d-2}(X)\odot(X^\top\Delta_X+\Delta_X^\top X)\odot\mathrm{P}_{W_\perp}\right) + 2d\Delta_X\left(\mathrm{K}_{d-1}(X)\odot\mathrm{P}_{W_\perp}\right)\\
\nabla_X\nabla_W f(X,W)[\Delta_X] &= -2d\left(\mathrm{K}_{d-1}(X)\odot(X^\top\Delta_X+\Delta_X^\top X)\right)W\\
\nabla_W\nabla_X f(X,W)[\delta_W] &= -2dX\left(\mathrm{K}_{d-1}(X)\odot(W\Delta_W^\top - \Delta_W W^\top)\right).
\end{aligned}$$

*Proof.* First, we make the following Taylor expansion of the monomial kernel. For some $\Delta_X\in\mathbb{R}^{n\times s}$,

$$\mathrm{K}_d(X+\Delta_X) = \mathrm{K}_d(X) + d\mathrm{K}_{d-1}(X)\odot(X^\top\Delta_X+\Delta_X^\top X) + \mathcal{O}(\Delta_X^2),$$

where $\odot$ is an entry-wise product. At $\mathcal{W}\in\mathrm{Grass}(N,r)$, the horizontal lift of tangent vectors is of the form $\Delta_W = W_\perp B\in\mathbb{R}^{N\times r}$ for some $W^\perp\in\mathbb{R}^{N\times(N-r)}$ and $B\in\mathbb{R}^{(N-r)\times r}$. We have

$$\begin{aligned}
\mathrm{P}_{W+\Delta_W} &= WW^\top + W\Delta_W^\top + \Delta_W W^\top\\
&= \mathrm{P}_W + W\Delta_W^\top + \Delta_W W^\top + \mathcal{O}(\Delta_W^2).
\end{aligned}$$

Let us write

$$f(X,W) = \text{trace}\left(\mathrm{K}_d(X) - \mathrm{P}_W\mathrm{K}_d(X)\right).$$

We find the gradient in $X$ using direct computation,

$$\begin{aligned}\nabla_X f(X,W) &= \nabla_X \text{trace}\left(\mathrm{P}_{W_\perp}\mathrm{K}_d(X)\right)\\ &= 2dX\left(\mathrm{K}_{d-1}(X)\odot \mathrm{P}_{W_\perp}\right).\end{aligned}$$

since $\mathrm{P}_{W_\perp}$ is symmetric. Quite naturally we find $\nabla_W f(X,W)$ with the expansion,

$$\begin{aligned}f(X,W+\Delta_W) &= \text{trace}\left(\mathrm{K}_d(X) - \mathrm{P}_{W+\Delta_W}\mathrm{K}_d(X)\right),\\ &= \text{trace}\left(\mathrm{K}_d(X) - (\mathrm{P}_W + W\Delta_W^\top + \Delta_W W^\top)\mathrm{K}_d(X)\right)\\ &= \text{trace}\left(P_{W_\perp}K_d(X) - (W\Delta_W^\top + \Delta_W W^\top)\mathrm{K}(_dX)\right)\\ &= \text{trace}\left(\mathrm{P}_{W_\perp}\mathrm{K}_d(X)\right) - \text{trace}\left((W\Delta_W^\top + \Delta_W W^\top)\mathrm{K}_d(X)\right)\\ &= f(X,W) - \text{trace}\left(W\Delta_W^\top\mathrm{K}_d(X)\right) - \text{trace}\left(\Delta_W W^\top\mathrm{K}(X))\right)\\ &= f(X,W) + \langle \Delta_W, -2\mathrm{K}_d(X)W\rangle,\end{aligned}$$

and we observe $\nabla_W f(X,W) = -2\mathrm{K}_d(X)W$. It follows that $\nabla_W^2 f(X,W)[E] = -2\mathrm{K}_d(X)E$. To find the second derivative in $X$, consider

$$\begin{aligned}\nabla_X f(X+\Delta_X,W) &= 2d(X+\Delta_X\left(\mathrm{K}_{d-1}(X+\Delta_X)\odot \mathrm{P}_{U_\perp}\right)\\ &= 2d(X+\Delta_X)\left(\left[\mathrm{K}_{d-1}(X) + (d-1)\mathrm{K}_{d-2}(X)\odot(X^\top\Delta_X + \Delta_X^\top X)\right]\odot \mathrm{P}_{W_\perp}\right)\\ &= 2dX\left(\mathrm{K}_{d-1}(X)\odot \mathrm{P}_{W_\perp}\right) + 2d(d-1)X\left(\mathrm{K}_{d-2}(X)\odot(X^\top\Delta_X + \Delta_X^\top X)\odot \mathrm{P}_{W_\perp}\right)\\ &\quad + 2d\Delta_X\left(\mathrm{K}_{d-1}(X)\odot \mathrm{P}_{W_\perp}\right) + \mathcal{O}(\Delta_X^2)\\ &= \nabla_X f(X,W) + 2d(d-1)X\left(\mathrm{K}_{d-2}(X)\odot(X^\top\Delta_X + \Delta_X^\top X)\odot \mathrm{P}_{W_\perp}\right)\\ &\quad + 2d\Delta_X\left(\mathrm{K}_{d-1}(X)\odot \mathrm{P}_{W_\perp}\right) + \mathcal{O}(\Delta_X^2).\end{aligned}$$

Thus we identify

$$\nabla_X^2 f(X,W)[\Delta_X] = 2d(d-1)X\left(\mathrm{K}_{d-2}(X)\odot(X^\top\Delta_X + \Delta_X^\top X)\odot \mathrm{P}_{W_\perp}\right) + 2d\Delta_X\left(\mathrm{K}_{d-1}(X)\odot \mathrm{P}_{W_\perp}\right).$$

We now compute the cross derivatives $\nabla_W\nabla_X f(X,W)[\Delta_W]$ and $\nabla_X\nabla_W f(X,W)[\Delta_W]$. Consider

$$\begin{aligned}\nabla_W f(X+\Delta_X,W) &= -2\mathrm{K}_d(X+\Delta_X)W\\ &= -2\left(\mathrm{K}_d(X) + d\mathrm{K}_{d-1}(X)\odot(X^\top\Delta_X + \Delta_X^\top X)\right)W\\ &= -2\mathrm{K}_d(X)W - 2d\left(\mathrm{K}_{d-1}(X)\odot(X^\top\Delta_X + \Delta_X^\top X)\right)W\\ &= \nabla_U f(X,W) - 2d\left(\mathrm{K}_{d-1}(X)\odot(X^\top\Delta_X + \Delta_X^\top X)\right)W,\end{aligned}$$

which yields

$$\nabla_X \nabla_W f(X,W)[\Delta_X] = -2d \left(\mathrm{K}_{d-1}(X) \odot (X^\top \Delta_X + \Delta_X^\top X)\right) W \in s \times r.$$

Similarly we have

$$\begin{aligned}
\nabla_X f(X, W + \Delta_W) &= 2dX \left(\mathrm{K}_{d-1}(X) \odot \mathrm{P}_{(W+\Delta_W)_\perp}\right) \\
&= 2dX \left(\mathrm{K}_{d-1}(X) \odot (\mathrm{P}_{W_\perp} - W\Delta_W^\top - \Delta_W W^\top)\right) \\
&= 2dX \left(\mathrm{K}_{d-1}(X) \odot \mathrm{P}_{W_\perp}\right) - 2dX \left(\mathrm{K}_{d-1}(X) \odot (W\Delta_W^\top - \Delta_W W^\top)\right) \\
&= \nabla_X f(X,W) - 2dX \left(\mathrm{K}_{d-1}(X) \odot (W\Delta_W^\top - \Delta_W W^\top)\right),
\end{aligned}$$

which gives

$$\nabla_W \nabla_X f(X,W)[\nabla_W] = -2dX \left(\mathrm{K}_{d-1}(X) \odot (W\Delta_W^\top - \Delta_W W^\top)\right) \in n \times s.$$

□

**Proposition A.2.** *For the Gaussian kernel* $\mathrm{K}_\sigma$ (Gaussian kernel)*, the Euclidean gradient of the cost function* (4.3) *is given by*

$$\nabla_X f(X,W) = -\frac{2}{\sigma^2} X \left(\mathrm{diag}\left(\mathrm{sum}(\mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}, 1)\right) - \mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}\right) \quad \textit{and} \quad \nabla_W f(X,W) = -2\mathrm{K}_\sigma(X) W.$$

*where* $\mathrm{sum}(\mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}, 1)$ *is the vector with the sum of each column of the matrix* $\mathrm{K}_\sigma \odot \mathrm{P}_{W^\perp}$*.*

*Proof.* Recall that $f(X,W) = \mathrm{trace}\left(\mathrm{P}_{W^\perp} K_\sigma(X,X)\right)$ and we write $\mathrm{P} = \mathrm{P}_{W^\perp}$ and $\mathrm{K} = \mathrm{K}_\sigma(X,X)$ to simplify the notations. Note that the matrices $\mathrm{P} \in \mathbb{R}^{s\times s}$ and $\mathrm{K} \in \mathbb{R}^{s \times s}$ are symmetrical. We have

$$f(X,W) = \sum_{i=1}^{s} \sum_{j=1}^{s} \mathrm{P}_{ij} \mathrm{K}_{ij}.$$

Hence, it comes

$$\begin{aligned}
\nabla_X f(X,W) &= \sum_{i=1}^{s} \sum_{j=1}^{s} \mathrm{P}_{ij} \nabla_X \mathrm{K}_{ij} \\
&= -\frac{1}{\sigma^2} \sum_{i=1}^{s} \sum_{j=1}^{s} \mathrm{P}_{ij} \mathrm{K}_{ij} \left[\dots \quad x_i - x_j \quad \dots \quad x_j - x_i \quad \dots\right] \\
&= -\frac{1}{\sigma^2} \sum_{i=1}^{s} \sum_{j=1}^{s} \mathrm{P}_{ij} \mathrm{K}_{ij} \left(x_i \left[\dots \quad 1 \quad \dots \quad -1 \quad \dots\right] + x_j \left[\dots \quad -1 \quad \dots \quad 1 \quad \dots\right]\right).
\end{aligned}$$

Denote by $a_{ij} \in \mathbb{R}^s$ the vector with entry $i$ equal to 1 and entry $j$ equal to $-1$ and all the other entries are zero. Then,

$$\begin{aligned}\nabla_X f(X, W) &= -\frac{1}{\sigma^2}\left(\sum_{i=1}^{s}\sum_{j=1}^{s} \mathrm{P}_{ij}\mathrm{K}_{ij} x_i a_{ij}^\top - \sum_{i=1}^{s}\sum_{j=1}^{s} \mathrm{P}_{ij}\mathrm{K}_{ij} x_j a_{ij}^\top\right) \\ &= -\frac{1}{\sigma^2}\left(\sum_{i=1}^{s} x_i \sum_{j=1}^{s} \mathrm{P}_{ij}\mathrm{K}_{ij} a_{ij}^\top - \sum_{j=1}^{s} x_j \sum_{i=1}^{s} \mathrm{P}_{ij}\mathrm{K}_{ij} a_{ij}^\top\right) \\ &= -\frac{1}{\sigma^2}\left(X(-W \odot K) + X\mathrm{diag}(\mathrm{sum}(\mathrm{K} \odot \mathrm{P}, 1))\right),\end{aligned}$$

where $\odot$ denotes an entry-wise product. □

## A.2 Proofs for Chapter 6

**Example A.1** (The Stiefel manifold). *Let $\mathcal{E} = \mathbb{R}^{n\times p}$, the Stiefel manifold is defined as*

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

*The manifold corresponds to the defining function $h \colon \mathbb{R}^{n\times p} \to \mathrm{Sym}(p) \colon X \mapsto h(X) = X^\top X - \mathrm{I}_p$, where $\mathrm{Sym}(p)$ is the set of symmetric matrices of size $p$. For any $R < 1$, all $X \in \mathbb{R}^{n\times p}$ such that $\|h(X)\| \leq R$, satisfy $\sigma_{\min}(\mathrm{D}h(X)) \geq 2\sigma_{\min}(X) \geq 2\sqrt{1-R}$. Therefore, A12 is satisfied for any $R < 1$ and $\underline{\sigma} \leq 2\sqrt{1-R}$.*

*Proof.* First note that the set $\mathrm{Sym}(p)$ has dimension $p(p + 1)/2$. Therefore

$$\sigma_{\min}(\mathrm{D}h(X)) = \sigma_{p(p+1)/2}(\mathrm{D}h(X)),$$

by definition. The differential of the defining function $h$ is given by

$$\mathrm{D}h(X) \colon \mathbb{R}^{n\times p} \to \mathrm{Sym}(p) \colon U \mapsto \mathrm{D}h(X)[U] = X^\top U + U^\top X.$$

To find the region where $\mathrm{D}h(X)$ is non-singular, we need to characterize a set of $X \in \mathbb{R}^{n\times p}$ such that $X^\top U + U^\top X$ spans $\mathrm{Sym}(p)$. We show that this set is constituted of all the matrices of full rank. Assume $X$ has rank $p$; we can pick $[V, V_\perp] \in O(n)$ such that $X = VP$, for some invertible $P \in \mathbb{R}^{p\times p}$. Any $U \in \mathbb{R}^{n\times p}$ can be written as $U = VA + V_\perp B$ for some $A \in \mathbb{R}^{p\times p}$ and $B \in \mathbb{R}^{(n-p)\times p}$. We find that $\mathrm{D}h(X)[U] = P^\top A + A^\top P$. Therefore, $\mathrm{D}h(X)[U] = 0$ if and only if $A = (P^\top)^{-1}\Omega$, for some $\Omega \in \mathrm{Skew}(p)$, i.e. $p(p-1)/2$ degrees of freedom and $\Omega^\top + \Omega = 0$. In other words, an antisymmetric $\Omega$ brings no contribution to $\mathrm{D}h(X)[U]$.

Therefore, consider $U = V(P^\top)^{-1}W$, with $W \in \mathrm{Sym}(p)$, this gives $\mathrm{D}h(X)[U] = 2W$. Hence $\mathrm{D}h(X)$ spans $\mathrm{Sym}(p)$ for any full rank $X \in \mathbb{R}^{n\times p}$. By definition,

$$\sigma_{\min}(\mathrm{D}h(X)) = \min_U \frac{\|\mathrm{D}h(X)[U]\|_\mathrm{F}}{\|U\|_\mathrm{F}}$$

Let us express $U$ as a function of $W$. Using $X = VP$ yields

$$U = V(P^\top)^{-1}W = XP^{-1}(P^\top)^{-1}W = X(X^\top X)^{-1}W.$$

This allows to write

$$\begin{aligned}\sigma_{\min}(\mathrm{D}h(X)) &= \min_{W\in\mathrm{Sym}(p)} \frac{2\,\|W\|_\mathrm{F}}{\|X(X^\top X)^{-1}W\|_\mathrm{F}}\\ &\geq \min_{Y\in\mathbb{R}^{p\times p}} \frac{2\,\|Y\|_\mathrm{F}}{\|X(X^\top X)^{-1}Y\|_\mathrm{F}}\\ &= \frac{2}{\sigma_{\min}(X^\dagger)}\\ &= 2\sigma_{\min}(X).\end{aligned}$$

Take a singular value decomposition, $X = U_1\Sigma U_2^\top$,

$$\begin{aligned}\|h(X)\| &= \left\|X^\top X - \mathrm{I}_p\right\|_\mathrm{F}\\ &= \left\|U_1(\Sigma^\top\Sigma - \mathrm{I}_p)U_1^\top\right\|_\mathrm{F}\\ &= \left\|\Sigma^2 - \mathrm{I}_p\right\|_\mathrm{F}.\end{aligned}$$

Take $R > 0$ such that $\|h(X)\| \leq R$. This implies $|\sigma_{\min}(X)^2 - 1| \leq R$. Firstly assume that $\sigma_{\min}(X)^2 < 1$, which gives $1 - \sigma_{\min}(X)^2 \leq R$ or $\sigma_{\min}(X)^2 \geq 1 - R$. This allows to write $\sigma_{\min}(X) \geq \sqrt{1-R}$. Now consider the case $\sigma_{\min}(X)^2 \geq 1$, where it is clear that $\sigma_{\min}(X) \geq \sqrt{1-R}$. In conclusion, for any $R < 1$, all $X$ such that $\|h(X)\| \leq R$, satisfy $\sigma_{\min}(\mathrm{D}h(X)) \geq 2\sigma_{\min}(X) \geq 2\sqrt{1-R}$. □

**Example A.2** (Convex quadratic constraint)**.** *Let $\mathcal{E} = \mathbb{R}^n$ and consider the set $\mathcal{M}$ for $h(x) = x^\top Ax + b^\top x + c$, where $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. For any $R < |h(x^*)|$, where $x^*$ is the minimizer of the quadratic, all $x \in \mathbb{R}^n$ satisfy $\sigma_{\min}(\mathrm{D}h(x)) > 0$. Additionally, $\sigma_{\min} = \min_{x\in\mathcal{C}} \sigma_{\min}(\mathrm{D}h(x)) > 0$ by compactness of $\mathcal{C}$.*

*Proof.* We have $\mathrm{D}h(x) = 2Ax + b$. Since $A$ is invertible, $\mathrm{D}h(x) \in \mathbb{R}^n$ is rank deficient at the unique point where $\mathrm{D}h(x^*) = 0$, that is, $x^* = -\frac{1}{2}A^{-1}b$, the minimizer of the quadratic constraint. We need to find $R$ such that $h(x^*) > R$, so that $h(x) \leq R \implies x \neq x^*$.

$$\begin{aligned}h(x^*) &= (x^*)^\top Ax^* + b^\top x^* + c\\ &= -\frac{1}{2}(A^{-1}b)^\top A - \frac{1}{2}(A^{-1}b) - \frac{1}{2}b^\top(A^{-1}b) + c\\ &= \frac{1}{4}(A^{-1}b)^\top b - \frac{1}{2}b^\top A^{-1}b + c\\ &= -\frac{1}{4}b^\top A^{-1}b^\top + c.\end{aligned}$$

So $\|h(x^*)\| = |-\frac{1}{4}b^\top A^{-1}b^\top + c| = |\frac{1}{2}b^\top x^* + c|$. We can take any $R < \|h(x^*)\|$, then $\forall x \in \mathcal{C}$, we have $\sigma_{\min}(\mathrm{D}h(x)) > 0$. □

**Proposition A.3** ((Bertsekas, 1982), Prop. 4.22)**.** *Let $g(x) = \mathcal{L}_\beta(x, \lambda(x))$ be Fletcher's augmented Lagrangian and assume $\mathcal{M} \subset \mathcal{D}$, where $\mathcal{D} = \{x \in \mathcal{E} : \operatorname{rank}(\mathrm{D}h(x)) = m\}$ and $\mathcal{M} = \{x \in \mathcal{E} : h(x) = 0\}$.*

1. *For any $\beta$, if $x$ is a first-order critical point of* (P)*, then $x$ is a first-order critical point of $g$.*

2. *Let $x \in \mathcal{D}$ and $\beta > \beta_1(x)$. If $x$ is a first-order critical point of $g$, then $x$ is a first-order critical point of* (P)*.*

3. *Let $x$ be a first-order critical point of* (P) *and let $K$ be a compact subset of $\mathcal{D}$. Assume $x$ is the unique global minimum of $f$ over $\mathcal{M} \cap K$ and that $x$ is in the interior of $K$. Then, there exists $\beta$ large enough such that $x$ is the unique global minimum of $g$ over $K$.*

4. *Let $x \in \mathcal{D}$ and $\beta > \beta_1(x)$. If $x$ is a local minimum of $g$, then $x$ is a local minimum of* (P)*.*

*Proof.* 1. If $x$ is a first-order critical point for (P), then $h(x) = 0$ and $\nabla f(x) = \mathrm{D}h(x)^*[\lambda(x)]$ by definition. Therefore,

$$\nabla g(x) = \nabla f(x) - \mathrm{D}h(x)^*[\lambda(x)] + 2\beta \mathrm{D}h(x)^*[h(x)] - \mathrm{D}\lambda(x)^*[h(x)] = 0. \tag{A.1}$$

2. Take $x \in \mathcal{D}$ with $\nabla g(x) = 0$. We find that

$$\begin{aligned}
0 &= \mathrm{D}h(x)[\nabla g(x)] \\
&= \mathrm{D}h(x)\big[\nabla\big(x \mapsto f(x) - \langle h(x), \lambda(x)\rangle + \beta\|h(x)\|^2\big)(x)\big] \\
&= \mathrm{D}h(x)[\nabla f(x) - \mathrm{D}h(x)^*[\lambda(x)] + 2\beta\mathrm{D}h(x)^*[h(x)] - \mathrm{D}\lambda(x)^*[h(x)]] \\
&= \mathrm{D}h(x)\big[\operatorname{grad}_{\mathcal{M}_x} f(x)\big] + 2\beta\mathrm{D}h(x)\left[\mathrm{D}h(x)^*[h(x)]\right] - \mathrm{D}h(x)[\mathrm{D}\lambda(x)^*[h(x)]] \\
&= \{2\beta\mathrm{D}h(x)\mathrm{D}h(x)^* - \mathrm{D}h(x)\mathrm{D}\lambda(x)^*\}\, h(x),
\end{aligned} \tag{A.2}$$

where the term $\mathrm{D}h(x)\big[\operatorname{grad}_{\mathcal{M}_x} f(x)\big]$ vanishes because the Riemannian gradient of $f$ restricted to $\mathcal{M}_x$ is tangent to $\mathcal{M}_x$ at $x$, and by definition this tangent space is the kernel of $\mathrm{D}h(x)$. Using Lemma 6.5,

$$\begin{aligned}
\sigma_{\min}\left(2\beta\mathrm{D}h(x)\mathrm{D}h(x)^* - \mathrm{D}h(x)\mathrm{D}\lambda(x)^*\right) &\geq \sigma_{\min}(2\beta\mathrm{D}h(x)\mathrm{D}h(x)^*) - \sigma_1(\mathrm{D}h(x)\mathrm{D}\lambda(x)^*) \\
&\geq 2\beta\sigma_{\min}^2(\mathrm{D}h(x)) - \sigma_1(\mathrm{D}h(x))C_\lambda(x).
\end{aligned} \tag{A.3}$$

If $\beta > \beta_1(x)$ (Definition 6.5), we see from (A.3) that the linear operator that appears on the right hand side of (A.2) is nonsingular, and therefore (A.2) implies $h(x) = 0$. Going back to (A.1) and using $\nabla g(x) = 0$ together with $h(x) = 0$, it follows that $\nabla f(x) = \mathrm{D}h(x)^*[\lambda(x)]$. This proves that $x$ is a first-order critical point of (P) with multipliers $\lambda(x)$.

3. The set $K$ is compact and therefore, for any $\beta$, there exists a global minimizer of $g$ inside $K$. We proceed by contradiction. Therefore, for any integer $k > 0$, there exists $\beta_k \geq k$ and a global minimizer $x_k$ of $g$ over $K$ such that $x_k \neq x$. This implies

$$g(x_k) \leq g(x) = f(x). \tag{A.4}$$

Hence, $\limsup_{k \longrightarrow \infty} g(x_k) \leq f(x)$. We shall show that $(x_k)_{k\in\mathbb{N}} \longrightarrow x$. Let $\bar{x}$ be a limit point of $(x_k)_{k\in\mathbb{N}}$. Since $\beta_k \longrightarrow \infty$, we have $h(\bar{x}) = 0$. Therefore,

$$f(\bar{x}) = g(\bar{x}) \leq f(x).$$

Given that $x_k \in K$ for all $k$, compactness ensures that $\bar{x} \in K$. Since $x$ is the unique global minimizer of $f$ over $K \cap \mathcal{M}$, it follows that $\bar{x} = x$.

Now we show that there exists some index $k$ such that $x_k = x$. Since $x_k \to x$, we can take an open ball $B$ centered around $x$ such that $x_k \in B$ for $k$ sufficiently large. We also chose $B$ such that $\mathrm{cl}(B) \subset K$. From item 2, for all $\beta \geq \bar{\beta}_1$, every critical point of $g$ inside $B$ is a first-order critical point of (P). Hence, for all $k$ sufficiently large, $x_k$ is a first-order critical point of $f$ on $\mathcal{M}$ (as it is a global min of $g$ inside $B \subset K$, it is a critical point of $g$ and point 2 applies on the compact set $\mathrm{cl}(B)$). This implies $h(x_k) = 0$ and $f(x_k) \leq f(x)$ from (A.4). Since $x$ is the unique global minimizer of $f$ over $K \cap \mathcal{M}$, it follows that $x_k = x$ for all $k$ sufficiently large. This contradicts that $x_k \neq x$ for all $k$ and proves the original statement.

4. From item 2, for all $\beta > \beta_1(x)$, if $x$ is a local minimum of $g$, then $x$ is a first-order critical point of (P). This implies

$$f(x) = g(x)$$

since $h(x) = 0$. From the local optimality of $x$ for $g$, there exists a ball $B \subset \mathcal{E}$ centered at $x$ such that

$$g(x) \leq g(y) \qquad\qquad \text{for all } y \in B.$$

This holds a fortiori for all $y \in B \cap \mathcal{M}$. Combining the last two results gives

$$f(x) = g(x) \leq g(y) = f(y) \qquad\qquad \text{for all } y \in B \cap \mathcal{M} \tag{A.5}$$

where the equalities hold because $x, y \in \mathcal{M}$ imply $h(x) = h(y) = 0$, and the inequality holds owing to $x, y \in B$. Equation (A.5) implies that $x$ is a local minimizer of (P). $\square$