Université catholique de Louvain

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Pôle d'ingénierie mathématique (INMA)
Travail de fin d'études

Promoteur : Prof. Pierre-Antoine Absil

Discrete curve fitting on manifolds

Nicolas Boumal

Louvain-la-Neuve, Juin 2010

Je remercie chaleureusement
Pierre-Antoine Absil, mon promoteur
ainsi que
Quentin Rentmeesters, son doctorant,
pour leur guidance éclairée.

Je dédicace ce travail à
Pierre Bolly, qui m'a introduit aux mathématiques.

Contents

Notations

$\|\cdot\|$	usual 2-norm in $\mathbb{R}^n$, Frobenius norm for matrices	4
$\mathcal{M}, \mathcal{N}$	(usually) smooth, finite-dimensional, Riemannian manifolds	10
x, y, p	points on a manifold	
$T_x\mathcal{M}$	tangent space to $\mathcal{M}$ at x	12
ξ, η, v	tangent vectors to a manifold	12
X, Y	vector fields on a manifold	13
$\mathrm{D}f(x)[\xi]$	derivative of f at x in the direction ξ	14
dist	geodesic distance on a manifold	17
Exp	exponential map on a manifold	19
Log	logarithmic map on a manifold	19
$\mathbb{S}^2$	unit sphere embedded in $\mathbb{R}^3$	34
$\mathbb{H}^n$	set of symmetric matrices	50
$\mathbb{P}^n_+$	set of positive-definite matrices	50

Table 1: *Notations*

Introduction

Let $p_1, p_2, \ldots, p_N$ be N data points in the Euclidean space $\mathbb{R}^n$ with time labels $t_1 \leq t_2 \leq \cdots \leq t_N$. A classic problem arising in many disciplines consists in searching for a curve $\gamma : [t_1, t_N] \to \mathbb{R}^n$ that simultaneously (i) reasonably fits the data and (ii) is sufficiently smooth. This may, among other things, help reduce measurement noise and fill gaps in the data. One way of formalizing this loose description of a natural need is to express γ as the minimizer of an energy function such as

$$E(\gamma) = \frac{1}{2}\sum_{i=1}^{N} \|p_i - \gamma(t_i)\|^2 + \frac{\lambda}{2}\int_{t_1}^{t_N} \|\dot{\gamma}(t)\|^2 \mathrm{d}t + \frac{\mu}{2}\int_{t_1}^{t_N} \|\ddot{\gamma}(t)\|^2 \mathrm{d}t, \tag{1}$$

defined over some suitable curve space Γ. The tuning parameters λ and μ enable the user to tune the balance between the conflicting goals of fitting and smoothness. It is well known (see state of the art, section 1.5) that (i) when $\lambda > 0, \mu = 0$, the optimal γ is piecewise affine and (ii) when $\lambda = 0, \mu > 0$ the optimal γ is an approximating cubic spline (provided Γ contains these curves). Cubic splines are piecewise cubic polynomial curves of class $\mathcal{C}^2$.

We focus on a broader problem. The data points p_i are now allowed to lie on a Riemannian manifold $\mathcal{M}$. $\mathbb{R}^n$ is the simplest example of such a manifold. Other examples treated in this document include the sphere embedded in $\mathbb{R}^3$, which we note $\mathbb{S}^2$, the cone of positive-definite matrices $\mathbb{P}_+^n$ and the special orthogonal group $\mathrm{SO}(3)$. Generalizations of (1) to manifolds have been proposed. In particular, Samir et al. give, mutatis mutandis, the following definition in [SASK09]:

$$E(\gamma) = \frac{1}{2}\sum_{i=1}^{N} \mathrm{dist}^2\left(p_i, \gamma(t_i)\right) + \frac{\lambda}{2}\int_{t_1}^{t_N} \left\langle \dot{\gamma}(t), \dot{\gamma}(t)\right\rangle_{\gamma(t)} \mathrm{d}t + \frac{\mu}{2}\int_{t_1}^{t_N} \left\langle \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}, \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}\right\rangle_{\gamma(t)} \mathrm{d}t, \tag{2}$$

where dist denotes the Riemannian (geodesic) distance on $\mathcal{M}$, $\langle\cdot,\cdot\rangle$ the Riemannian metric, $\dot{\gamma}$ the first derivative of γ and $\frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}$ the second covariant derivative of γ (chapter 2 defines these terms and other relevant elements of Riemannian geometry). Samir et al. solve this problem by computing the gradient of E in the so-called Palais metric and applying a steepest descent scheme to minimize E. In essence, the problem is a variational problem in infinite dimension. The (necessary) discretization for implementation purposes arises at the very end of the algorithm design process. In conclusion of [SASK09], the authors wonder what would happen if one were to discretize (2) directly. The present work is an attempt to answer that question.

The main contribution of this document lies in the proposed discretization of the problem. The curve space is reduced to the set of sequences of N_d points on the manifold, $\Gamma = \mathcal{M} \times \ldots \times \mathcal{M}$ (N_d copies of $\mathcal{M}$). For a discrete curve $\gamma = (\gamma_1, \ldots, \gamma_{N_\mathrm{d}}) \in \Gamma$, each γ_i is associated to a fixed time τ_i such that $t_1 = \tau_1 < \tau_2 < \cdots < \tau_{N_\mathrm{d}} = t_N$. We propose the following discretization of (2):

$$E(\gamma) = \frac{1}{2}\sum_{i=1}^{N} \mathrm{dist}^2\left(p_i, \gamma_{s_i}\right) + \frac{\lambda}{2}\sum_{i=1}^{N_\mathrm{d}} \alpha_i \left\langle v_i, v_i\right\rangle_{\gamma_i} + \frac{\mu}{2}\sum_{i=1}^{N_\mathrm{d}} \beta_i \left\langle a_i, a_i\right\rangle_{\gamma_i}. \tag{3}$$

Here, the indices s_i are chosen such that τ_{s_i} is closest (ideally equal) to t_i. The tangent vectors v_i and a_i, rooted at γ_i, are approximations to the velocity and acceleration vectors $\dot{\gamma}(t_i)$ and $\frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}(t_i)$. We compute these with generalized finite differences, which we introduce in this work. The weights α_i and β_i are chosen based on a suitable numerical integration scheme like, e.g., the trapezium method, such that the sums effectively discretize the integrals present in (2). We also

propose other ways of discretizing (2). Interestingly, they all come down to the same discretization in $\mathbb{R}^n$ but, in general, differ on manifolds. Since the curve space is now finite-dimensional, we may apply the optimization algorithms from [AMS08] to minimize (3). Chapter 3 covers the details.

Measurements on manifolds arise naturally in applications. To cite but a few examples:

- Positions on Earth closely resemble points on $\mathbb{S}^2$;
- Configurations of rigid bodies in space are fully described by the position of their center of mass and their orientation, which jointly constitute a point of $\mathrm{SE}(3) = \mathbb{R}^3 \times \mathrm{SO}(3)$, the special Euclidean group. Rigid body motion estimation thus comes down to regression on SE(3). The same mathematics can be used for smooth camera transitions in computer graphics;
- Diffusion-Tensor MRI measures (noisy) positive-definite matrices in the brain for medical imaging purposes;
- Shapes (seen as closed curves) can be measured, e.g., by contour detectors applied to video streams. Shapes belong to the shape space, a complex manifold we plan on working with in the future.

Each time an application uses data belonging to (tractable) manifolds, noise reduction (i.e., smoothing) of the measurements and interpolation can be carried out by the generic techniques developed in this document. For the control engineer, our algorithms may be helpful, e.g., in case the control variables from the systems at hand belong to manifolds.

We successfully implemented our methods in $\mathbb{R}^n$ (chapter 1), on the sphere $\mathbb{S}^2$ (chapter 4), in the cone of positive-definite matrices $\mathbb{P}_+^n$ (chapter 5) and on the special orthogonal group SO(3) (appendix C). For $\mathbb{P}_+^n$ and SO(3), we constructed explicit formulas for the gradients of real-valued functions involving the matrix logarithm. Our results correspond nicely to what one would expect based on the continuous case. Our short answer to the question Samir et al. ask is that the regression problem on these manifolds can, in general, be solved satisfactorily via our direct discretization approach.

Chapter 1

Discrete curve fitting in $\mathbb{R}^n$

In this work, we show how one can find a discrete representation of a curve lying on a manifold that strikes a (tunable) balance between data fitting and smoothness. Before we give a general formulation of that problem (along with a proper definition of fitting and smoothness on manifolds), it is helpful to investigate the Euclidean case. This will help us take conscience of how the vector space structure of $\mathbb{R}^n$ simplifies the whole process of formalizing and solving the problem. Later on, we will use these observations to establish a list of tools we should have on manifolds to adequately generalize some of the more fundamental concepts that can sometimes be hidden behind generic linear combinations.

First off, we give a formal definition of the regression problem for time-labeled data in $\mathbb{R}^n$ in the continuous case, i.e., what we are looking for is a suitable smooth curve defined over some time interval and taking values in $\mathbb{R}^n$. An objective function (energy function) will be introduced to capture the balance we are seeking between fitting and smoothness. The second step consists in proposing a discretization for that objective. As we will see, it is very natural to use finite differences for that matter. Doing so, it will be sufficient to describe a curve using a finite set of sampling points. We will solve the discrete problem in a least-squares framework. We will then investigate where the vector space structure of $\mathbb{R}^n$ has been used, and where we should generalize the problem formulation for the case of non-vector spaces. More specifically, in the following chapters, we will consider finite-dimensional, smooth Riemannian manifolds, namely: the sphere and the set of positive-definite matrices. We close this chapter by a short state of the art.

1.1 The problem in a continuous setting

Let us consider a collection of N weighted, time-labeled points p_i in $\mathbb{R}^n$: $\{(w_i, t_i, p_i) : i = 1, \ldots, N\}$, such that the time labels t_i are increasing with i and such that the weights w_i are non-negative. We are searching for a $\mathcal{C}^k$ (smooth "enough") function $\gamma : [t_1, t_N] \to \mathbb{R}^n$ such that γ approximately fits the data and such that γ is not too "wiggly". The former means that we would like $\gamma(t_i)$ to be close to p_i (even more so when w_i is high) whereas the latter means that we would like the derivatives of γ to have a small norm, effectively discouraging sharp turns and detours. Furthermore, we would like to be able to tune the balance between those two competing objectives.

The following objective (energy) function, defined over the set of differentiable functions γ from $[t_1, t_N]$ to $\mathbb{R}^n$, $\mathcal{C}^k([t_1, t_N], \mathbb{R}^n)$ for some appropriate k, captures the problem description:

$$E(\gamma) = \frac{1}{2}\sum_{i=1}^{N} w_i \|\gamma(t_i) - p_i\|^2 + \frac{\lambda}{2}\int_{t_1}^{t_N} \|\dot{\gamma}(t)\|^2 + \frac{\mu}{2}\int_{t_1}^{t_N} \|\ddot{\gamma}(t)\|^2 \tag{1.1}$$

We use the usual 2-norm in $\mathbb{R}^n$, $\|x\|^2 = x^T x$. The two parameters λ and μ enable us to adjust the importance of the penalties on the 2-norms of the first derivative $\dot{\gamma}$ and the second derivative $\ddot{\gamma}$. Playing with these parameters lets us move the minimizer of E from near interpolation (piecewise linear or cubic splines) to near linear regression.

The curve space over which E has to be minimized to obtain our regression model has infinite dimension. Minimizing E is a variational problem, which can be complicated, not to mention that our goal in this document is to solve the problem on manifolds rather than on $\mathbb{R}^n$. The added complexity of the continuous approach is the main reason why, in the next section, we discretize equation (1.1).

For the sake of completeness, let us mention that the continuous objective (1.1) has been generalized to Riemannian manifolds to take this form:

$$E(\gamma) = \frac{1}{2}\sum_{i=1}^{N} \operatorname{dist}^2(\gamma(t_i), p_i) + \frac{\lambda}{2}\int_{t_1}^{t_N} \langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)} \,\mathrm{d}t + \frac{\mu}{2}\int_{t_1}^{t_N} \left\langle \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}, \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2} \right\rangle_{\gamma(t)} \mathrm{d}t, \tag{1.2}$$

where dist denotes the Riemannian (geodesic) distance, $\langle \cdot, \cdot \rangle$ the Riemannian metric, $\dot{\gamma}$ the first derivative of γ and $\frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}$ the second covariant derivative of γ. Among others (see section 1.5), in [SASK09], Samir et al. have solved regression problems based on this energy function. Their strategy was to discretize as late as possible in the process of writing minimization algorithms. In this document, we take the opposite strategy. We discretize curves immediately, reducing the curve space to a finite dimensional manifold, hence reverting to the optimization setting already covered in [AMS08]. This is in direct response to the concluding remarks of [SASK09].

1.2 Objective discretization

Instead of optimizing (1.1) over all functions in $\mathcal{C}^k([t_1, t_N], \mathbb{R}^n)$, we would like to restrict ourselves to a finite dimensional curve space Γ. A sensible way to do that would be to consider a finite basis of smooth functions, then to optimize (1.1) over the space spanned by said basis. This approach works great in the Euclidean case. It can be generalized to some extent, and at the cost of increased complexity, to manifolds, see section 1.5.

We would like to treat the Euclidean regression problem in a way that will help us work out the Riemannian case. This is why, even in this section devoted to the Euclidean case, we study the simple discretization obtained by sampling curves in time.

More precisely, we consider the discrete curve space $\Gamma = \mathbb{R}^n \times \ldots \times \mathbb{R}^n \equiv \mathbb{R}^{n \times N_\mathrm{d}}$ consisting of all sequences of N_d points in $\mathbb{R}^n$ (the d subscript stands for *discrete*). The sampling times

$\tau_1, \ldots, \tau_{N_\mathrm{d}}$ are fixed.

For each discrete curve $\gamma = (\gamma_1, \ldots, \gamma_{N_\mathrm{d}}) \in \Gamma$, there is an underlying continuous curve $\gamma(t)$ such that $\gamma(\tau_i) = \gamma_i$. Giving a precise definition of such a $\gamma(t)$ would yield a clean way of discretizing (1.1). We refrain from doing that because this is the specific point that makes matters intricate on manifolds. In the Euclidean case though, one could use, e.g., shape functions like the ones we use in finite element methods.

Let us review the three components of (1.1) and see how we can discretize them without explicitly constructing an underlying $\gamma(t)$.

Misfit penalty The first term of (1.1) penalizes misfit between the curve γ and the N data points p_i. We introduce indices $s_1, \ldots, s_N$ such that t_i is closest to τ_{s_i} and define the misfit penalty, E_d, as:

$$E_\mathrm{d} : \Gamma \to \mathbb{R}^+ : \gamma \mapsto E_\mathrm{d}(\gamma) = \frac{1}{2} \sum_{i=1}^{N} w_i \|\gamma_{s_i} - p_i\|^2. \tag{1.3}$$

We can always choose the sampling times τ_i such that, for some s_i, $t_i = \tau_{s_i}$. There may be multiple data points associated to the same discretization point, i.e., $s_i = s_j$ for some $i \neq j$.

Velocity penalty The second term of (1.1) penalizes a kind of L^2-norm of the first derivative of $\gamma(t)$. Two things need to be discretized here: (i) the derivative $\dot{\gamma}$ and (ii) the integral. Mainstream numerical methods can help us here. We define the velocity penalty, E_v, as:

$$E_\mathrm{v} : \Gamma \to \mathbb{R}^+ : \gamma \mapsto E_\mathrm{v}(\gamma) = \frac{1}{2} \sum_{i=1}^{N_\mathrm{d}} \alpha_i \|v_i\|^2, \tag{1.4}$$

where the velocity approximations v_i are obtained via finite differences and the coefficients α_i correspond to the weights in, e.g., the trapezium rule for numerical integration. For the latter, any method suited for fixed, non-homogeneous sampling is fine.

We delay the explicit construction of the formulas for v_i, which is standard material, to section 3.1. The important feature is that $v_i \approx \dot{\gamma}(t_i)$ is a linear combination of γ_i and its neighbor(s). At the end points, v_0 and v_{N_d} are defined as unilateral finite differences.

Acceleration penalty The third term of (1.1) penalizes the same kind of L^2-norm of the second derivative of $\gamma(t)$. The discretization process is very similar to what was done for E_v. We introduce the acceleration penalty E_a as:

$$E_\mathrm{a} : \Gamma \to \mathbb{R}^+ : \gamma \mapsto E_\mathrm{a}(\gamma) = \frac{1}{2} \sum_{i=1}^{N_\mathrm{d}} \beta_i \|a_i\|^2. \tag{1.5}$$

Again, the acceleration vector a_i at γ_i can be obtained via finite differences as a linear combination of γ_i and its neighbors, except at the end points for which we need unilateral differences. The appropriate formulas can be found in section 3.1. The β_i's are adequate weights for a numerical integration method, which need not be the same as the α_i's used for E_v.

The complete objective function is the following, with tuning parameters λ and μ:

$$E : \Gamma \to \mathbb{R}^+ : \gamma \mapsto E(\gamma) = E_\mathrm{d}(\gamma) + \lambda E_\mathrm{v}(\gamma) + \mu E_\mathrm{a}(\gamma) \tag{1.6}$$

We state without proof that, as the maximum discretization step $\Delta\tau_i = \tau_{i+1} - \tau_i$ goes to zero, the optimal discrete objective value and the optimal continuous objective value become equivalent. This is a consequence of Taylor's theorem and of the definition of Riemann integrals.

In the next section, we show how (1.6) can be minimized by solving a sparse linear system in a least-squares sense.

1.3 Least-squares formulation

First off, let us observe that the objective function E is decoupled along the n dimensions of the problem. Hence, we can focus on solving the problem in $\mathbb{R}^n$ with $n = 1$. The objective function E:

$$E(\gamma) = \frac{1}{2}\sum_{i=1}^{N} w_i \|\gamma_{s_i} - p_i\|^2 + \frac{\lambda}{2}\sum_{i=1}^{N_\mathrm{d}} \alpha_i \|v_i\|^2 + \frac{\mu}{2}\sum_{i=1}^{N_\mathrm{d}} \beta_i \|a_i\|^2, \tag{1.7}$$

can be rewritten as a sum of vector 2-norms:

$$E(\gamma) = \frac{1}{2}\left\| \begin{pmatrix} \sqrt{w_1}(\gamma_{s_1} - p_1) \\ \vdots \\ \sqrt{w_N}(\gamma_{s_N} - p_N) \end{pmatrix} \right\|^2 + \frac{1}{2}\left\| \begin{pmatrix} \sqrt{\lambda\alpha_1} v_1 \\ \vdots \\ \sqrt{\lambda\alpha_{N_\mathrm{d}}} v_{N_\mathrm{d}} \end{pmatrix} \right\|^2 + \frac{1}{2}\left\| \begin{pmatrix} \sqrt{\mu\beta_1} a_1 \\ \vdots \\ \sqrt{\mu\beta_{N_\mathrm{d}}} a_{N_\mathrm{d}} \end{pmatrix} \right\|^2.$$

The vectors appearing in this expression are affine combinations of the variables γ_i and hence can be rewritten as (temporarily considering γ as a column vector):

$$\begin{aligned} E(\gamma) &= \frac{1}{2}\|P_\mathrm{d}\gamma - q_\mathrm{d}\|^2 + \frac{1}{2}\|P_\mathrm{v}\gamma\|^2 + \frac{1}{2}\|P_\mathrm{a}\gamma\|^2 \\ &= \frac{1}{2}\gamma^T(P_\mathrm{d}^T P_\mathrm{d} + P_\mathrm{v}^T P_\mathrm{v} + P_\mathrm{a}^T P_\mathrm{a})\gamma - q_\mathrm{d}^T P_\mathrm{d}\gamma + \frac{q_\mathrm{d}^T q_\mathrm{d}}{2} \\ &= \frac{1}{2}\gamma^T A\gamma - b^T\gamma + c, \end{aligned}$$

for adequate matrices $P_\mathrm{d} \in \mathbb{R}^{N\times N_\mathrm{d}}$, $P_\mathrm{v}, P_\mathrm{a} \in \mathbb{R}^{N_\mathrm{d}\times N_\mathrm{d}}$, a vector $q_\mathrm{d} \in \mathbb{R}^N$, and the obvious definitions for A, b and c. The vector q_d is the only object that depends on the data points p_i. The matrices $P_\mathrm{d}, P_\mathrm{v}$ and P_a depend on the sampling times τ_i as well as on the various weights introduced. Hence, A does not depend on the data and must be constructed only once to solve a problem in $\mathbb{R}^n$, even for $n > 1$. Differentiating and setting the gradient E to zero, we straightforwardly obtain the linear system:

$$A\gamma = b,$$

which needs to be solved n times, for n different b vectors. This problem has a rich structure because of the nature of the constraints. Namely, A is 5-diagonal and positive-definite. The regression problem in $\mathbb{R}^n$ can thus reliably be solved in $\mathcal{O}(nN_\mathrm{d})$ operations. This is a standard result from numerical linear algebra, see for example [VD08].

The discrete regression problem reduces to a highly structured quadratic optimization problem without constraints, which can be solved, e.g., with an LU decomposition and n back-substitutions. If one wants to add constraints (possibly forcing interpolation by some or all of the points p_i or constraining speed/acceleration to respect lower and/or upper bounds on some time intervals), modern Quadratic Programming solvers can come in handy.

1.4 What if we generalize to Riemannian manifolds?

In the next chapters, we generalize to Riemannian manifolds. In doing so, the lack of a vector space structure raises two issues:

- How can we generalize the objective E? Finite differences, since they are linear combinations, do not make sense on manifolds. Same goes for the Euclidean distance. We will need substitutes for these elements. As we will see in section 3.1, the linear combinations appearing in finite differences are not arbitrary: they make use of particular vectors that, in a Euclidean space, lead from one point to another in the same space. We exploit that.
- Since the objective will not be quadratic anymore, how can we minimize it? We will need optimization algorithms on manifolds. We will investigate iterative descent methods. For this to work, we will need tools to transform curve guesses into other curves, moving in some direction in the curve space.

In the next chapter, we study a few properties and tools of Riemannian geometry. These will enable us to give a precise meaning to the loose terms used in this section. In the next section, we give a brief review of the state of the art.

1.5 Short state of the art

Several existing methods for tackling the conflicting nature of the curve fitting problem (i.e., goodness of fit vs regularity) rely on turning the fitting task into an optimization problem where one of the criteria becomes the objective function and the other criterion is turned into a constraint.

Let us first consider the case where the goodness of fit is optimized under a regularity constraint. When $\mathcal{M} = \mathbb{R}^n$, a possible regularity constraint, already considered by Lagrange, is to restrict the curve γ to the family of polynomial functions of degree not exceeding m, $(m \leq N-1)$. This least-squares problem cannot be straightforwardly generalized to Riemannian manifolds because of the lack of an equivalent of polynomial curves on Riemannian manifolds. The notion of polynomial does not carry to $\mathcal{M}$ in an obvious way. An exception is the case $m = 1$; the polynomial functions in $\mathbb{R}^n$ are then straight lines, whose natural generalization on Riemannian manifolds are geodesics. The problem of fitting geodesics to data on a Riemannian manifold $\mathcal{M}$ was considered in [MS06] for the case where $\mathcal{M}$ is the special orthogonal group $\mathrm{SO}(n)$ or the unit sphere $\mathbb{S}^n$.

The other case is when a regularity criterion is optimized under constraints on the goodness of fit. In this situation, it is common to require a perfect fit: the curve fitting problem becomes an interpolation problem. When $\mathcal{M} = \mathbb{R}^n$, the interpolating curves γ that minimize $\frac{1}{2}\int_0^1 \|\ddot{\gamma}(t)\|^2 \mathrm{d}t$ (in the appropriate function space) are known as *cubic splines*. For the case where $\mathcal{M}$ is a nonlinear manifold, several results on interpolation can be found in the literature. Crouch and Silva Leite [CS91, CS95] generalized cubic splines to Riemannian manifolds, defined as curves γ that minimize, under interpolation conditions, the function

$$\frac{1}{2}\int_{t_1}^{t_N} \left\langle \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}(t), \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}(t) \right\rangle_{\gamma(t)} \mathrm{d}t,$$

where $\frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}$ denotes the (Levi-Civita) second covariant derivative and $\langle \cdot, \cdot \rangle_x$ stands for the Riemannian metric on $\mathcal{M}$ at x. They gave a necessary condition for optimality in the form of a fourth-order differential equation, which generalizes a result of Noakes et al. [NHP89]. Splines of class $\mathcal{C}^k$ were generalized to Riemannian manifolds by Camarinha et al. [CSC95]. Still in the context of interpolation on manifolds, but without a variational interpretation, we mention the literature on splines based on generalized Bézier curves, defined by a generalization to manifolds of the de Casteljau algorithm; see [CKS99, Alt00, PN07]. Recently, Jakubiak et al. [JSR06] presented a geometric two-step algorithm to generate splines of an arbitrary degree of smoothness in Euclidean spaces, then extended the algorithm to matrix Lie groups and applied it to generate smooth motions of 3D objects.

Another approach to interpolation on manifolds consists of mapping the data points onto the tangent space at a particular point of $\mathcal{M}$, then computing an interpolating curve in the tangent space, and finally mapping the resulting curve back to the manifold. The mapping can be defined, e.g., by a rolling procedure, see [HS07, KDL07].

Yet another approach, that we focus on in this work, consists in optimizing (1.2) directly without constraints on γ other than differentiability conditions. While the concept is well known in $\mathbb{R}^n$, its generalization to Riemannian manifolds is more recent.

In [MSH06], the objective function is defined by equation (1.2) with $\mu = 0$, over the class of all piecewise smooth curves $\gamma : [0, 1] \to \mathcal{M}$, where λ (> 0) is a smoothing parameter. Solutions to this variational problem are piecewise smooth geodesics that best fit the given data. As shown in [MSH06], when λ goes to $+\infty$, the optimal curve converges to a single point which, for certain

classes of manifolds, is shown to be the Riemannian mean of the data points. When λ goes to zero, the optimal curve goes to a broken geodesic on $\mathcal{M}$ interpolating the data points.

In [MS06], the objective function is defined by equation (1.2) with $\lambda = 0$, over a certain set of admissible $\mathcal{C}^2$ curves. The authors give a necessary condition of optimality that takes the form of a fourth-order differential equation involving the covariant derivative and the curvature tensor along with certain regularity conditions at the time instants t_i, $i = 1, \ldots, N$ [MS06, Th. 4.4]. The optimal curves are *approximating cubic splines*: they are approximating because in general $\gamma(t_i)$ differs from p_i, and they are cubic splines because they are obtained by smoothly piecing together segments of cubic polynomials on $\mathcal{M}$, in the sense of Noakes et al. [NHP89]. It is also shown in [MS06, Prop. 4.5] that, as the smoothing parameter μ goes to $+\infty$, the optimal curve converges to the best least-squares geodesic fit to the data points at the given instants of time. When μ goes to zero, the approximating cubic spline converges to an interpolating cubic spline [MS06, Prop. 4.6].

In summary, while the concept of smoothing spline has been satisfactorily generalized to Riemannian manifolds, the emphasis has been laid on studying properties of these curves rather than on proposing efficient methods to compute them. In this work, we focus on computing discrete representations of such curves.

Chapter 2

Elements of Riemannian geometry

In the first chapter, we showed how easily the discrete regression problem can be stated and solved in a vector space. We then made a few statements about where that structure is used and what needs to be done to generalize the concepts to other spaces.

In this chapter, we give a brief overview of some Riemannian geometry elements. We start by giving a proper definition of manifolds as sets accompanied by atlases, which are sets of charts. We will not explicitly use charts in our applications. However, defining charts and atlases is a necessary effort to build the higher level tools we do use later on. Loosely, manifolds are sets that locally resemble $\mathbb{R}^n$. Charts are the mathematical concept that embody this idea. At each point of a manifold we can define a vector space, called the tangent space, that captures this essential property. Inner products are then defined as positive-definite forms on each tangent space. Considering spaces where these inner products vary continuously on the manifold leads to the concept of Riemannian manifolds. Using these inner products we give a natural definition of distance between points and gradients of real-valued functions on the manifold. Subsequently, the exponential and logarithmic maps are introduced. They are the key concepts that, later on, will enable us to generalize finite differences and descent methods for optimization. Finally, we introduce parallel translation, which is the tool that lets us compare tangent vectors rooted at different points of a manifold.

Should you not be familiar with this material, we suggest you think of the general manifold $\mathcal{M}$ used henceforth simply as the sphere. It is also helpful to think about how the concepts specialize in the case of $\mathbb{R}^n$. This chapter is mainly based on [AMS08, Hai09, Boo86]. All the (beautiful) figures come from the book [AMS08] by Absil et al. I thank the authors for them.

2.1 Charts and manifolds

In this document, we are mainly concerned with embedded Riemannian manifolds (we will define these terms shortly). Nevertheless, it is important to give a proper definition of smooth manifolds first. Intuitively, manifolds are sets that can be locally identified with patches of $\mathbb{R}^n$. These identifications are called charts. A set of charts that covers the whole set is called an atlas for the set. The set and the atlas together constitute a manifold. More formally:

Definition 2.1.1 (chart)**.** *Let M be a set. A* chart *of M is a pair (U, φ) where $U \subset M$ and φ 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 $p \in U$, the elements of $\varphi(p) = (x_1, \ldots, x_n)$ are called the coordinates of p in the chart (U, φ).*

Definition 2.1.2 (compatible charts)**.** *Two charts (U, φ) and (V, ψ) of M, of dimensions n and m respectively, are* smoothly compatible *($\mathcal{C}^\infty-$compatible) if 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 $\mathbb{R}^m$,*
- *$\psi \circ \varphi^{-1} : \varphi(U \cap V) \to \psi(U \cap V)$ is a smooth diffeomorphism (i.e., a smooth invertible function with smooth inverse).*

When $U \cap V \neq \emptyset$, the latter implies $n = m$.

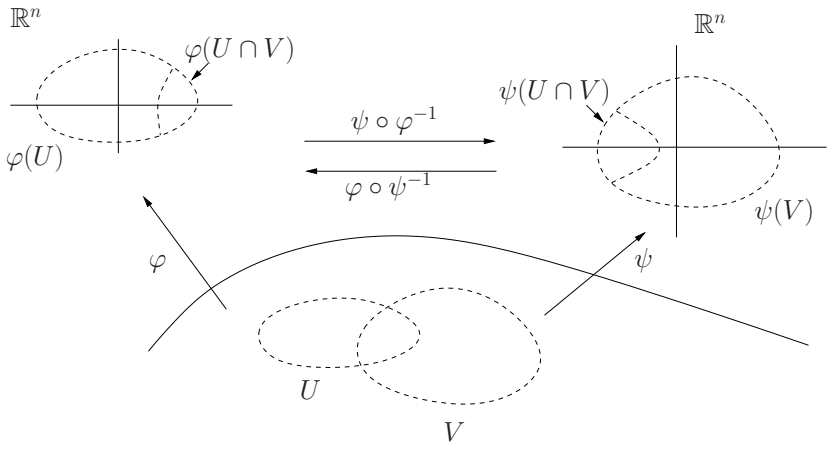

Figure 2.1: *Charts. Figure courtesy of Absil et al., [AMS08].*

Definition 2.1.3 (atlas)**.** *A set $\mathcal{A}$ of pairwise smoothly compatible charts $\{(U_i, \varphi_i), i \in I\}$ 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. Given an atlas $\mathcal{A}$, one can generate a unique maximal atlas $\mathcal{A}^+$. Such an atlas contains $\mathcal{A}$ as well as all the charts compatible with $\mathcal{A}$. Classically, we define:

Definition 2.1.4 (manifold)**.** *A* smooth manifold *is a pair $\mathcal{M} = (M, \mathcal{A}^+)$, where M is a set and $\mathcal{A}^+$ is a maximal atlas of M.*

Example 2.1.5. *The vector space $\mathbb{R}^n$ can be endowed with an obvious manifold structure. Simply consider $\mathcal{M} = (\mathbb{R}^n, \mathcal{A}^+)$ where the atlas $\mathcal{A}^+$ contains the identity map $(\mathbb{R}^n, \varphi)$, $\varphi : U = \mathbb{R}^n \to \mathbb{R}^n : x \mapsto \varphi(x) = x$. Defined as such, $\mathcal{M}$ is called a* Euclidean space*. We always use the notation $\mathbb{R}^n$ regardless of whether we consider the vector space or the Euclidean space.*

Often times, we will refer to M when we really mean $\mathcal{M}$ and vice versa. Once the differential manifold structure is clearly stated, no confusion is possible. For example, the notation $\mathcal{M} \subset \mathbb{R}^n$ means $M \subset \mathbb{R}^n$.

Definition 2.1.6 (dimension)**.** *Given a manifold $\mathcal{M} = (M, \mathcal{A}^+)$, if all the charts of $\mathcal{A}^+$ have the same dimension n, we call n the* dimension of the manifold.

We need one last definition to assess smoothness of curves defined on manifolds:

Definition 2.1.7 (smooth mapping)**.** *Let $\mathcal{M}$ and $\mathcal{N}$ be two smooth manifolds. A mapping $f : \mathcal{M} \to \mathcal{N}$ is of* class $\mathcal{C}^k$ *if, for all p in $\mathcal{M}$, there is a chart (U, φ) of $\mathcal{M}$ and (V, ψ) of $\mathcal{N}$ such that $p \in U$, $f(U) \subset V$ and*

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

is of class $\mathcal{C}^k$. The latter is called the local expression of f in the charts (U, φ) and (V, ψ). It maps open sets of $\mathbb{R}^n$, hence all classic tools apply.

This definition does not depend on the choice of charts.

In the next section, we introduce tangent spaces and tangent vectors. Those are objects that give a local representation of manifolds as vector spaces.

2.2 Tangent spaces and tangent vectors

As is customary in differential geometry, we will define tangent vectors as equivalence classes of functions. This surprising detour from the very simple idea underlying tangent vectors (namely that they *point* in directions one can follow at a given point on a manifold), once again, comes from the lack of a vector space structure. We first construct a simpler definition. In $\mathbb{R}^n$, one can define derivatives for curves $c : \mathbb{R} \to \mathbb{R}^n$:

$$c'(0) := \lim_{t \to 0} \frac{c(t) - c(0)}{t}.$$

The difference appearing in the numerator does not, in general, make sense for manifolds. For manifolds embedded in $\mathbb{R}^n$ however (like, e.g., the sphere), we can still make sense of the definition with the appropriate space identifications. A simple definition of tangent spaces in this setting follows.

Definition 2.2.1 (tangent spaces for manifolds embedded in $\mathbb{R}^n$)**.** *Let $\mathcal{M} \subset \mathbb{R}^n$ be a smooth manifold. The* tangent space *at $x \in \mathcal{M}$, noted $T_x\mathcal{M}$, is the vector subspace of $\mathbb{R}^n$ defined by:*

$$T_x\mathcal{M} = \{v \in \mathbb{R}^n \ : \ v = c'(0) \text{ for some smooth } c : \mathbb{R} \to \mathcal{M} \text{ such that } c(0) = x\}.$$

The dimension of $T_x\mathcal{M}$ is the dimension of a chart of $\mathcal{M}$ containing x.

The following theorem is useful when dealing with such manifolds defined by equality constraints on the Cartesian coordinates.

Theorem 2.2.2. *Let $\mathcal{M}$ be a subset of $\mathbb{R}^n$. (1) and (2) are equivalent:*

(1) $\mathcal{M}$ is a smooth submanifold of $\mathbb{R}^n$ of dimension $n - m$,

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

Furthermore, the tangent space at x is given by $T_x\mathcal{M} = \ker \mathrm{d}f_x$.

We did not define the term submanifold properly. Essentially, *(1)* means that $\mathcal{M}$ is a manifold endowed with the differential structure naturally inherited from $\mathbb{R}^n$.

Example 2.2.3. *An example of a smooth, two-dimensional submanifold of $\mathbb{R}^3$ is the sphere $\mathbb{S}^2 = \{x \in \mathbb{R}^3 \ : \ x^T x = 1\}$. Use $f : \mathbb{R}^3 \to \mathbb{R} : x \mapsto f(x) = x^T x - 1$ in theorem 2.2.2. The tangent spaces are then $T_x\mathbb{S}^2 = \{v \in \mathbb{R}^3 \ : \ v^T x = 0\}$. We illustrate this on figure 2.2.*

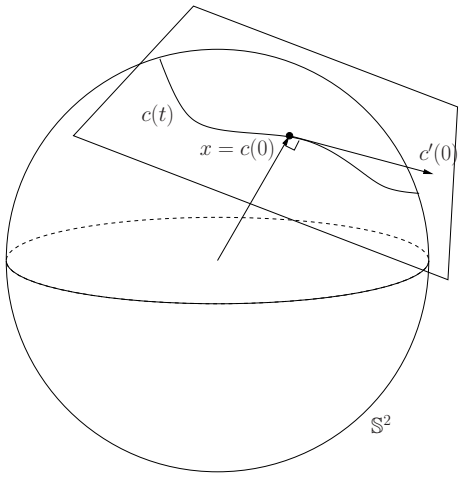

Figure 2.2: *Tangent space on the sphere. Since $\mathbb{S}^2$ is an embedded submanifold of $\mathbb{R}^3$, the tangent space $T_x\mathbb{S}^2$ can be pictured as the plane tangent to the sphere at x, with origin at x. Figure courtesy of Absil et al., [AMS08].*

In general, $\mathcal{M}$ is not embedded in $\mathbb{R}^n$. We now give a more general definition of tangent vectors that does not need the manifold to be embedded in a vector space. Let $\mathcal{M}$ be a smooth manifold and p be a point on $\mathcal{M}$. We define:

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

the set of differentiable curves on $\mathcal{M}$ passing through p at $t = 0$. Here, $c \in \mathcal{C}^1$ is to be understood with the definition 2.1.7, with the obvious manifold structure on open intervals of $\mathbb{R}$ derived from example 2.1.5. We define an equivalence relation on C_p, noted $\sim$. Let (U, φ) be a chart of $\mathcal{M}$ such that $p \in U$, and let $c_1, c_2 \in C_p$. Then, $c_1 \sim c_2$ if and only if $\varphi \circ c_1$ and $\varphi \circ c_2$ have same derivative at 0, i.e.:

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

It is easy to prove that this is independent of the choice of chart.

Definition 2.2.4 (tangent space, tangent vector)**.** *The* tangent space *to $\mathcal{M}$ at p, noted $T_p\mathcal{M}$, is the quotient space:*

$$T_p\mathcal{M} = C_p/\sim$$

Given $c \in C_p$, the equivalence class $[c]$ is an element of $T_p\mathcal{M}$ that we call a tangent vector *to $\mathcal{M}$ at p.*

The mapping

$$\theta_p^\varphi : T_p\mathcal{M} \to \mathbb{R}^n : [c] \mapsto \theta_p^\varphi([c]) = \frac{\mathrm{d}}{\mathrm{d}t}\varphi(c(t))\bigg|_{t=0}$$

is bijective and naturally defines a vector space structure over $T_p\mathcal{M}$. This structure, again, is independent of the chart. When $\mathcal{M} \subset \mathbb{R}^n$, it is possible to build a vector space isomorphism (i.e., an invertible linear map) proving that the two definitions 2.2.1 and 2.2.4 are, essentially, equivalent.

Definition 2.2.5 (tangent bundle)**.** *Let $\mathcal{M}$ be a smooth manifold. The* tangent bundle*, noted $T\mathcal{M}$, is the set:*

$$T\mathcal{M} = \coprod_{p \in \mathcal{M}} T_p\mathcal{M},$$

where $\coprod$ stands for disjoint union. We define π as the natural projection on the roots of vectors, i.e., $\pi(\xi) = p$ if and only if $\xi \in T_p\mathcal{M}$.

The tangent bundle is itself a manifold. This enables us to define vector fields on manifolds as smooth mappings.

Definition 2.2.6 (vector field)**.** *A* vector field *is a smooth mapping from $\mathcal{M}$ to $T\mathcal{M}$ such that $\pi \circ X = \mathrm{Id}$, the identity map. The vector at p will be written as X_p or $X(p)$ and lies in $T_p\mathcal{M}$.*

An important example of a vector field is the gradient of a scalar field on a manifold, which we will define later on and extensively use in our optimization algorithms.

Let us introduce an alternative notation for tangent vectors to curves (velocity vectors) that will make things look more natural in the sequel. Given a curve of class $\mathcal{C}^1$, $\gamma : [a, b] \to \mathcal{M}$, and $t \in [a, b]$, define another such curve on $\mathcal{M}$:

$$\gamma_t : [a - t, b - t] \to \mathcal{M} : \tau \mapsto \gamma_t(\tau) = \gamma(t + \tau).$$

This curve is such that $\gamma_t(0) = \gamma(t) \triangleq p$. $[\gamma_t] \in T_p\mathcal{M}$ is a vector tangent to γ at time t. We propose to write

$$\dot{\gamma}(t) \triangleq [\gamma_t].$$

When using definition 2.2.1, $\dot{\gamma}(t)$ is identified with $\gamma'(t)$. This notation has the advantage of capturing the nature of this vector (a *velocity* vector).

2.3 Inner products and Riemannian manifolds

Exploiting the vector space structure of tangent spaces, we define inner products as follows:

Definition 2.3.1 (inner product)**.** *Let $\mathcal{M}$ be a smooth manifold and fix $p \in \mathcal{M}$. An* inner product *$\langle \cdot, \cdot \rangle_p$ on $T_p\mathcal{M}$ is a bilinear, symmetric positive-definite form on $T_p\mathcal{M}$, i.e., $\forall \xi, \zeta, \eta \in T_p\mathcal{M}$, $a, b \in \mathbb{R}$:*

- $\langle a\xi + b\zeta, \eta \rangle_p = a \langle \xi, \eta \rangle_p + b \langle \zeta, \eta \rangle_p$,
- $\langle \xi, \zeta \rangle_p = \langle \zeta, \xi \rangle_p$,
- $\langle \xi, \xi \rangle_p \geq 0$, $\langle \xi, \xi \rangle_p = 0 \Leftrightarrow \xi = 0$.

Often, when it is clear from the context that ξ and η are rooted at p, i.e., $\xi, \eta \in T_p\mathcal{M}$, we write $\langle \xi, \eta \rangle$ instead of $\langle \xi, \eta \rangle_p$. The norm of a tangent vector $\xi \in T_p\mathcal{M}$ is $\|\xi\|_p = \sqrt{\langle \xi, \xi \rangle_p}$.

Let $\mathcal{M}$ be a smooth manifold of dimension n and p be a point on $\mathcal{M}$. Let (U, φ) be a chart of $\mathcal{M}$ such that $p \in U$. Let $(e_1, \ldots, e_n)$ be a basis of $\mathbb{R}^n$. We define the tangent vectors $E_i = (\theta_p^\varphi)^{-1}(e_i)$ for i ranging from 1 to n. We state that every vector $\xi \in T_p\mathcal{M}$ can be uniquely decomposed as $\xi = \sum_{i=1}^n \xi^i E_i$, i.e., $(E_1, \ldots, E_n)$ is a basis of $T_p\mathcal{M}$. Now using the inner product of definition 2.3.1, we define $g_{ij} = \langle E_i, E_j \rangle_p$. Hence, decomposing $\zeta = \sum_{i=1}^n \zeta^i E_i$ like we did for ξ, we have:

$$\langle \xi, \zeta \rangle_p = \sum_{i=1}^n \sum_{j=1}^n g_{ij} \xi^i \zeta^j = \hat{\xi}^T G_p \hat{\zeta},$$

where we defined the vectors $\hat{\xi} = (\xi^1, \ldots, \xi^n)^T$, $\hat{\zeta} = (\zeta^1, \ldots, \zeta^n)^T$ and the matrix G_p such that $(G_p)_{ij} = \langle E_i, E_j \rangle_p$. The matrix G_p is symmetric, positive-definite.

Definition 2.3.2 (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 $\mathcal{M}$, i.e., for each $p \in \mathcal{M}$, $g_p(\cdot, \cdot)$ is an inner product on $T_p\mathcal{M}$.*

In this definition, *smoothly varying* means that the functions $(G_p)_{ij}$, which are functions from U to $\mathbb{R}$, are smooth, in the sense described in definition 2.1.7, with the obvious manifold structure on $\mathbb{R}$.

As is customary, we will often refer to a Riemannian manifold $(\mathcal{M}, g)$ simply as $\mathcal{M}$, when the metric is clear from the context. From now on, when we quote the term manifold, unless otherwise stated, we refer to a smooth, finite-dimensional Riemannian manifold.

2.4 Scalar fields on manifolds and the gradient vector field

Let $\mathcal{M}$ be a smooth manifold. A *scalar field* on $\mathcal{M}$ is a smooth function $f : \mathcal{M} \to \mathbb{R}$. We now generalize the concept of directional derivatives to scalar fields on manifolds.

Definition 2.4.1 (directional derivative)**.** *The* directional derivative *of a scalar field f on $\mathcal{M}$ at $p \in \mathcal{M}$ in the direction $\xi = [c] \in T_p\mathcal{M}$ is the scalar:*

$$\mathrm{D}f(p)[\xi] = \frac{\mathrm{d}}{\mathrm{d}t} f(c(t))\bigg|_{t=0}$$

It is easily checked that this definition does not depend on the choice of c, the representative of the equivalence class ξ. In the above notation, the brackets around ξ are a convenient way of denoting that ξ is the direction. They do not mean that we are considering some sort of equivalence class of ξ (that would not make sense).

The following definition is of major importance for our purpose. It generalizes gradients of scalar fields to manifolds.

Definition 2.4.2 (gradient)**.** *Let f be a scalar field on a smooth, finite-dimensional Riemannian manifold $\mathcal{M}$. The* gradient of f at p*, denoted by* $\operatorname{grad} f(p)$*, is defined as the unique element of $T_p\mathcal{M}$ satisfying:*

$$\mathrm{D}f(p)[\xi] = \langle \operatorname{grad} f(p), \xi \rangle_p, \quad \forall \xi \in T_p\mathcal{M}$$

Thus, $\operatorname{grad} f : \mathcal{M} \to T\mathcal{M}$ *is a vector field on $\mathcal{M}$.*

For a scalar field f on a Euclidean space, $\operatorname{grad} f$ is the usual gradient, which we note ∇f. Remarkably, and similarly to the Euclidean case, the gradient defined above is the steepest-ascent vector field and the norm $\| \operatorname{grad} f(p) \|_p$ is the steepest slope of f at p.

Based on this definition, one way to derive an expression for the gradient of a scalar field f is to work out an expression for the directional derivative of f, according to definition 2.4.1, then to write it as an inner product suitable for direct identification. That might not be practical. Theorem 2.4.5 provides an alternative, often shorter, way for Riemannian submanifolds of Euclidean spaces. The proof was derived independently but undoubtedly resembles existing proofs. We first need a definition.

Definition 2.4.3 (Riemannian submanifold)**.** *A* Riemannian submanifold $\mathcal{M}$ *of $\mathbb{R}^n$ is a submanifold of $\mathbb{R}^n$ (as indirectly defined by theorem 2.2.2) equipped with the Riemannian metric inherited from $\mathbb{R}^n$, i.e., the metric on $\mathcal{M}$ is obtained by restricting the metric on $\mathbb{R}^n$ to $\mathcal{M}$.*

Riemannian submanifolds of $\mathbb{R}^n$ are quite common and easy to picture. In particular, since such manifolds are embedded in $\mathbb{R}^n$, following definition 2.2.1, the tangent space $T_x\mathcal{M}$ is a vector subspace of $\mathbb{R}^n$. We can thus define (unique) orthogonal projectors

$$P_x : \mathbb{R}^n \to T_x\mathcal{M} : v \mapsto P_x v, \text{ and}$$
$$P_x^\perp : \mathbb{R}^n \to T_x^\perp\mathcal{M} : v \mapsto P_x^\perp v$$

such that $v = P_x v + P_x^\perp v$. This follows from the direct sum decomposition $\mathbb{R}^n = T_x\mathcal{M} \oplus T_x^\perp\mathcal{M}$.

Example 2.4.4 (continued from example 2.2.3)**.** *The Riemannian metric on the sphere is obtained by restricting the metric on $\mathbb{R}^3$ to $\mathbb{S}^2$. Hence, for $x \in \mathbb{S}^2$ and $v_1, v_2 \in T_x\mathbb{S}^2$, $\langle v_1, v_2 \rangle_x = v_1^T v_2$. The orthogonal projector on the tangent space $T_x\mathbb{S}^2$ is $P_x = I - xx^T$.*

Theorem 2.4.5. *Let $\mathcal{M} \subset \mathbb{R}^n$ be a Riemannian submanifold of $\mathbb{R}^n$ equipped with the inner product $\langle \cdot, \cdot \rangle$ and let $\overline{f} : \mathbb{R}^n \to \mathbb{R}$ be a smooth scalar field. Let $f = \overline{f}\big|_{\mathcal{M}}$ be the restriction from $\overline{f}$ to $\mathcal{M}$. Then:*

$$\operatorname{grad} f : \mathcal{M} \to T\mathcal{M} : x \mapsto \operatorname{grad} f(x) = P_x \nabla \overline{f}(x),$$

where P_x is the orthogonal projector from $\mathbb{R}^n$ onto $T_x\mathcal{M}$.

Proof. Let $x \in \mathcal{M}$. For all $v \in T_x\mathcal{M}$, consider a smooth curve $c_v : \mathbb{R} \to \mathcal{M}$ such that $c_v(0) = x$ and $c_v'(0) = v$. Then, by definition 2.4.1:

$$\begin{aligned}
\mathrm{D}f(x)[v] &= \frac{\mathrm{d}}{\mathrm{d}t} f(c_v(t))\bigg|_{t=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \overline{f}(c_v(t))\bigg|_{t=0} \\
&= \left\langle \nabla \overline{f}(x), c_v'(0) \right\rangle \\
&\text{Decompose the gradient in its tangent and normal components:} \\
&= \left\langle P_x \nabla \overline{f}(x) + P_x^\perp \nabla \overline{f}(x), v \right\rangle \\
&v \text{ is a tangent vector, hence:} \\
&= \left\langle P_x \nabla \overline{f}(x), v \right\rangle
\end{aligned}$$

The result follows immediately from definition 2.4.2. □

2.5 Connections and covariant derivatives

Let $\mathcal{M}$ be a Riemannian manifold and X, Y be vector fields on $\mathcal{M}$. We would like to define the derivative of Y at x in the direction X_x. If $\mathcal{M}$ were a Euclidean space, we would write:

$$\mathrm{D}Y(x)[X_x] = \lim_{t \to 0} \frac{Y(x + tX_x) - Y(x)}{t}$$

Of course, when $\mathcal{M}$ is not a vector space, the above equation does not make sense because $x + tX_x$ is undefined. Furthermore, even if we give meaning to this sum - and we will in section 2.7 - $Y(x + tX_x)$ and $Y(x)$ would not belong to the same vector spaces. Hence their difference would be undefined too.

To overcome these difficulties, we need the concept of *connection*. This can be generically defined for manifolds. Since we are mainly interested in Riemannian manifolds, we focus on the so called Riemannian, or Levi-Civita connections. The derivatives defined via these connections are notably interesting because they give a coordinate-free means of defining acceleration along a curve (i.e., the derivative of the velocity vector) as well as the Hessian of a scalar field (i.e., the derivative of the gradient vector field). By coordinate-free, we mean that the choice of charts is made irrelevant.

We now quickly go over the definition of affine connection for manifolds and the Levi-Civita theorem, specific to Riemannian manifolds. We then show a result for Riemannian submanifolds and promptly specialize it to submanifolds of Euclidean spaces. The latter is the important result for our study and comes in a simple form. The reader can safely skim through the technical definitions that we have to introduce in order to get there.

Definition 2.5.1 (affine connection)**.** *Let $\mathcal{X}(\mathcal{M})$ denote the set of smooth vector fields on $\mathcal{M}$ and $\mathcal{F}(\mathcal{M})$ denote the set of smooth scalar fields on $\mathcal{M}$. An* affine connection ∇ *on a manifold $\mathcal{M}$ is a mapping*

$$\nabla : \mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M}) \to \mathcal{X}(\mathcal{M})$$

which is denoted by $(X, Y) \mapsto \nabla_X Y$ and satisfies the following properties:

(1) $\mathcal{F}(\mathcal{M})$-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' law): $\nabla_X(fY) = (Xf)Y + f\nabla_X Y$,

in which $X, Y, Z \in \mathcal{X}(\mathcal{M})$, $f, g \in \mathcal{F}(\mathcal{M})$ and $a, b \in \mathbb{R}$. We have used a standard interpretation of vector fields as derivations on $\mathcal{M}$. The notation Xf stands for a scalar field on $\mathcal{M}$ such that $Xf(p) = \mathrm{D}f(p)[X_p]$. The above properties should be compared to the usual properties

of derivations in $\mathbb{R}^n$. Every smooth manifold admits infinitely many affine connections. This approach is called an axiomatization: we state the properties we desire in the definition, then only investigate whether such objects exist.

Definition 2.5.2 (covariant derivative)**.** *The vector field $\nabla_X Y$ is called the* covariant derivative *of Y with respect to X for the affine connection ∇.*

Essentially, at each point $p \in \mathcal{M}$, $\nabla_X Y(p)$ is a vector telling us how the vector field Y is changing in the direction X_p. This interpretation shows that it is non-essential that the vector fields X and Y be defined over all of $\mathcal{M}$. We can define covariant derivatives along curves. Typically, Y would be defined on the trajectory of a curve γ and the derivative direction $X_{\gamma(t)}$ would be equal to $\dot{\gamma}(t)$, the tangent vector to the curve at $\gamma(t)$, i.e., the velocity vector.

The following example shows a natural affine connection in Euclidean space.

Example 2.5.3. *In $\mathbb{R}^n$, the classical directional derivative defines an affine connection:*

$$(\nabla_X Y)_x = \lim_{t \to 0} \frac{Y(x + tX_x) - Y(x)}{t} = \mathrm{D}Y(x)[X_x]$$

This should give us confidence that the definition 2.5.1 is a good definition. As often, the added structure of Riemannian manifolds makes for stronger results. The Levi-Civita theorem singles out one particular affine connection for each Riemannian manifold.

Theorem 2.5.4 (Levi-Civita)**.** *On a Riemannian manifold $\mathcal{M}$ there exists a unique affine connection ∇ 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 $X, Y, Z \in \mathcal{X}(\mathcal{M})$. 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 field defined by $[X, Y]f = X(Yf) - Y(Xf), \forall f \in \mathcal{F}(\mathcal{M})$, again using the interpretation of vector fields as derivations. Not surprisingly, the connection exposed in example 2.5.3 is the Riemannian connection on Euclidean spaces. We always assume that the Levi-Civita connection is the one we use. The next theorem is an important result about the Riemannian connection of a submanifold of a Riemannian manifold taken from [AMS08]. This situation is illustrated on figure 2.3.

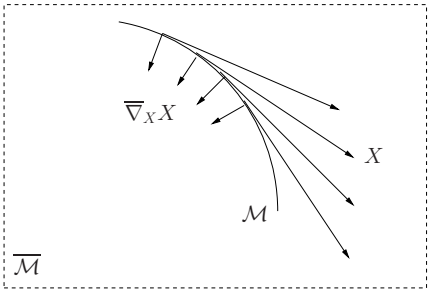

Figure 2.3: *Riemannian connection $\overline{\nabla}$ in a Euclidean space $\overline{\mathcal{M}}$ applied to a tangent vector field X to a circle. We observe that $\overline{\nabla}_X X$ is not tangent to the circle, hence simply restricting $\overline{\nabla}$ to the circle is not an option. As theorem 2.5.5 shows, we need to project $(\overline{\nabla}_X X)_x$ on the tangent space $T_x\mathcal{M}$ to obtain $(\nabla_X X)_x$. Figure courtesy of Absil et al., [AMS08].*

Theorem 2.5.5. *Let $\mathcal{M}$ be a Riemannian submanifold of a Riemannian manifold $\overline{\mathcal{M}}$ and let ∇ and $\overline{\nabla}$ denote the Riemannian connections on $\mathcal{M}$ and $\overline{\mathcal{M}}$. Then,*

$$(\nabla_X Y)_p = P_p(\overline{\nabla}_X Y)_p$$

for all $X_p \in T_p\mathcal{M}$ and $Y \in \mathcal{X}(\mathcal{M})$.

In particular, for Euclidean spaces (example 2.5.3),

$$(\nabla_X Y)_x = P_x(\mathrm{D}Y(x)[X_x]) \tag{2.1}$$

This means that the Riemannian connection on $\mathcal{M}$ can be computed via a classical directional derivative in the embedding space followed by a projection on the tangent space. Equation (2.1) is the most important result of this section. It gives us a practical means of computing derivatives of vector fields on the sphere for example, which is useful in chapter 4.

The acceleration along a curve can now be defined.

Definition 2.5.6 (acceleration along a curve)**.** *Let $\gamma : \mathbb{R} \to \mathcal{M}$ be a $\mathcal{C}^2$ curve on $\mathcal{M}$. The* acceleration *along γ is given by:*

$$\frac{\mathrm{D}^2}{\mathrm{d}t^2}\gamma(t) = \frac{\mathrm{D}}{\mathrm{d}t}\dot{\gamma}(t) = \nabla_{\dot{\gamma}(t)}\dot{\gamma}(t).$$

In the above equation, we abused the notation for $\dot{\gamma}$ which is tacitly supposed to be smoothly extended to a vector field $X \in \mathcal{X}(\mathcal{M})$ such that $X_{\gamma(t)} = \dot{\gamma}(t)$. For submanifolds of Euclidean spaces, by equation (2.1) *and using definition 2.2.1 for tangent vectors, this reduces to:*

$$\frac{\mathrm{D}^2}{\mathrm{d}t^2}\gamma(t) = P_{\gamma(t)}\frac{\mathrm{d}^2}{\mathrm{d}t^2}\gamma(t).$$

An axiomatic version of this definition is given in [AMS08, p. 102]. It has the added advantage of showing linearity of the operator $\frac{\mathrm{D}}{\mathrm{d}t}$ as well as a natural product and composition rule.

2.6 Distances and geodesic curves

The availability of inner products on the tangent spaces makes for an easy definition of curve length and distance.

Definition 2.6.1 (length of a curve)**.** *The* length of a curve *of class $\mathcal{C}^1$, $\gamma : [a, b] \to \mathcal{M}$, on a Riemannian manifold $(\mathcal{M}, g)$, with $\langle \xi, \eta \rangle_p \triangleq g_p(\xi, \eta)$, is defined by*

$$L(\gamma) = \int_a^b \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)}}\, \mathrm{d}t = \int_a^b \|\dot{\gamma}(t)\|_{\gamma(t)}\, \mathrm{d}t.$$

If $\mathcal{M}$ is embedded in $\mathbb{R}^n$, $\dot{\gamma}(t)$ can be replaced by $\gamma'(t)$, where γ is considered to be a function from $[a, b]$ to $\mathbb{R}^n$ and with the right definition of g.

Definition 2.6.2 (Riemannian distance)**.** *The* Riemannian distance *(or* geodesic distance*) on $\mathcal{M}$ is given by:*

$$\mathrm{dist} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}^+ : (p, q) \mapsto \mathrm{dist}\,(p, q) = \inf_{\gamma \in \Gamma} L(\gamma),$$

where Γ is the set of all $\mathcal{C}^1$ curves $\gamma : [0, 1] \to \mathcal{M}$ such that $\gamma(0) = p$ and $\gamma(1) = q$.

Under very reasonable conditions (see [AMS08, p. 46]), one can show that the Riemannian distance defines a metric. The definition above captures the idea that the distance between two points is the length of the shortest path joining these two points. In a Euclidean space, such a path would simply be the line segment joining the points. Another characteristic of line segments seen as curves with arclength parameterization is that they have zero acceleration. The next definition generalizes the concept of straight lines, preserving this zero acceleration characteristic, to Riemannian manifolds.

Definition 2.6.3 (geodesic curve)**.** *A curve $\gamma : \mathbb{R} \to \mathcal{M}$ is a* geodesic curve *if and only if $\frac{\mathrm{D}^2}{\mathrm{d}t^2}\gamma(t) \equiv 0$, i.e., if it has zero acceleration on all its domain.*

For close points, geodesic curves are shortest paths. It is not true for any two points on a geodesic though. Indeed, think of two points on the equator of the unit sphere in $\mathbb{R}^3$. The equator itself, parameterized by arclength, is a geodesic. Following this geodesic, one can join the two points via a path of length r or a path of length $2\pi - r$. Unless $r = \pi$, one of these paths is bound to be suboptimal.

2.7 Exponential and logarithmic maps

Exponentials are mappings that, given a point x on a manifold and a tangent vector ξ at x, generalize the concept of "$x + \xi$". In a Euclidean space, the sum $x + \xi$ is a point in space that can be reached by leaving x in the direction ξ. With exponentials, $\mathrm{Exp}_x(\xi)$ is a point on the manifold that can be reached by leaving x and moving in the ξ direction while remaining *on* the manifold. Furthermore, the trajectory followed is a geodesic (zero acceleration) and the distance traveled equals the norm of ξ.

Definition 2.7.1 (exponential map)**.** *Let $\mathcal{M}$ be a Riemannian manifold and $x \in \mathcal{M}$. For every $\xi \in T_x\mathcal{M}$, there exists an open interval $I \ni 0$ and a unique geodesic $\gamma(t; x, \xi) : I \to \mathcal{M}$ 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*

$$\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M} : \xi \to \mathrm{Exp}_x(\xi) = \gamma(1; x, \xi)$$

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

Exponentials can be expensive to compute. The following concept, *retractions*, has a simpler definition that captures the most important aspects of exponentials as far as we are concerned. Essentially, we drop the requirement that the trajectory by a geodesic, as well as the equality between distance traveled and $\|\xi\|$. It was introduced by Absil et al. in [AMS08]. Figure 2.4 illustrates the concept.

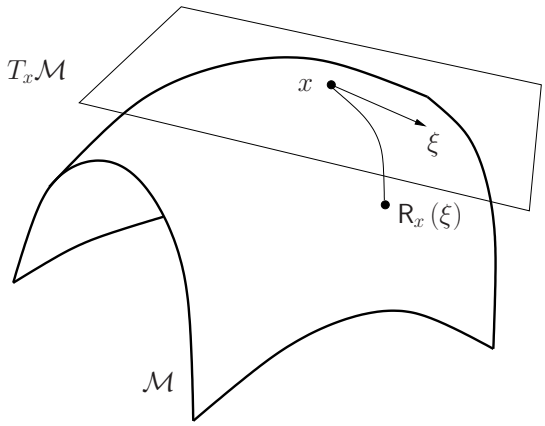

Figure 2.4: *Retraction. Figure courtesy of Absil et al., [AMS08].*

Definition 2.7.2 (retraction)**.** *A* retraction *on a manifold $\mathcal{M}$ is a smooth mapping R from the tangent bundle $T\mathcal{M}$ onto $\mathcal{M}$ with the following properties. Let R_x denote the restriction of R to $T_x\mathcal{M}$.*

(1) $\mathsf{R}_x(0) = x$*, where* 0 *is the zero element of* $T_x\mathcal{M}$.

(2) The differential $(\mathrm{D}\mathsf{R}_x)_0 : T_0(T_x\mathcal{M}) \equiv T_x\mathcal{M} \to T_x\mathcal{M}$ *is the identity map on* $T_x\mathcal{M}$*,* $(\mathrm{D}\mathsf{R}_x)_0 = \mathrm{Id}$ *(local rigidity).*

Equivalently, the local rigidity condition can be stated as: $\forall \xi \in T_x\mathcal{M}$, the curve $\gamma_\xi : t \mapsto \mathsf{R}_x(t\xi)$ satisfies $\dot{\gamma}_\xi(0) \triangleq [\gamma_\xi] = \xi$. In particular, an exponential map is a retraction. One can think of retractions as mappings that share the important properties we need with the exponential map, while being defined in a flexible enough way that we will be able to propose retractions that are, computationally, cheaper than exponentials. Retractions are the core concept needed to generalize descent algorithms to manifolds, see sections 3.4 and 3.5.

A related concept is the logarithmic map. Not surprisingly, it is defined as the inverse mapping of the exponential map. For two points x and y, logarithms generalize the concept of "$y - x$".

Definition 2.7.3 (logarithmic map)**.** *Let $\mathcal{M}$ be a Riemannian manifold. We define*

$$\mathrm{Log}_x : \mathcal{M} \to T_x\mathcal{M} : y \mapsto \mathrm{Log}_x(y) = \xi, \text{ such that } \mathrm{Exp}_x(\xi) = y \text{ and } \|\xi\|_x = \mathrm{dist}(x, y).$$

Given a root point x and a target point y, the logarithmic map returns a tangent vector at x pointing toward y and such that $\| \mathrm{Log}_x(y) \| = \mathrm{dist}(x, y)$. As is, this definition is not perfect. There might indeed be more than one eligible ξ. For example, think of the sphere $\mathbb{S}^2$ and place x and y at the poles: for any vector $\eta \in T_x\mathbb{S}^2$ such that $\|\eta\| = \pi$, we have $\mathrm{Exp}_x(\eta) = y$. For a more careful definition of the logarithm, see, e.g., [dC92]. As long as x and y are not "too far apart", this definition is satisfactory. Again, computing this map can be computationally expensive. Plus, we will need to differentiate it with respect to x and y later on. We pragmatically introduce, in this work, the notion of *generalized logarithm*, just like we introduced retractions as proxies for the exponential.

The inverse function theorem on manifolds can be stated like so, see, e.g., [Boo86].

Theorem 2.7.4 (inverse function theorem)**.** *Let $\mathcal{M}, \mathcal{N}$ be smooth manifolds and let $f : \mathcal{M} \to \mathcal{N}$ be a smooth function. If the differential of f at $x \in \mathcal{M}$,*

$$\mathrm{D}f_x : T_x\mathcal{M} \to T_{f(x)}\mathcal{N},$$

is a vector space isomorphism (i.e., an invertible linear map), then there exists an open neighborhood $U \subset \mathcal{M}$ of x such that the restriction

$$f|_U : U \to f(U)$$

is a diffeomorphism (i.e., a smooth invertible function with smooth inverse). Furthermore, writing $g = f|_U$, $(\mathrm{D}g^{-1})_{g(x)} = (\mathrm{D}g_x)^{-1}$.

Let us consider a retraction R on a smooth manifold $\mathcal{M}$ and let us consider a point $x \in \mathcal{M}$. The map

$$\mathsf{R}_x : T_x\mathcal{M} \to \mathcal{M}$$

is a smooth function from $\mathcal{M}$ to the smooth manifold $T_x\mathcal{M}$. Furthermore, according to the definition 2.7.2 for retractions, the differential

$$(\mathrm{D}\mathsf{R}_x)_0 : T_0(T_x\mathcal{M}) \equiv T_x\mathcal{M} \to T_x\mathcal{M}$$

is the identity map Id on $T_x\mathcal{M}$ which, of course, is a vector space isomorphism. By the inverse function theorem, there exists a neighborhood $U \subset T_x\mathcal{M}$ of 0, and a smooth function

$$L_x : \mathsf{R}_x(U) \subset \mathcal{M} \to U$$

such that $L_x(\mathsf{R}_x(\xi)) = \xi, \forall \xi \in U$. In particular, $L_x(x) = 0$ since $\mathsf{R}_x(0) = x$. Since $\mathrm{Id}^{-1} = \mathrm{Id}$, we also have that the differential

$$(\mathrm{D}L_x)_x : T_x\mathcal{M} \to T_0(T_x\mathcal{M}) \equiv T_x\mathcal{M}$$

is the identity map on $T_x\mathcal{M}$. We introduce a new definition for the purpose of this work.

Definition 2.7.5 (generalized logarithms)**.** *A* generalized logarithm *is a smooth mapping L from $\mathcal{M} \times \mathcal{M}$ to $T\mathcal{M}$ with the following properties. Let $\mathsf{L}_x : \mathcal{M} \to T_x\mathcal{M}$ denote the restriction from L to $\{x\} \times \mathcal{M}$, with $x \in \mathcal{M}$.*

(1) $\mathsf{L}_x(x) = 0$, where 0 is the zero element of $T_x\mathcal{M}$.

(2) The differential $(\mathrm{D}\mathsf{L}_x)_x : T_x\mathcal{M} \to T_0(T_x\mathcal{M}) \equiv T_x\mathcal{M}$ is the identity map on $T_x\mathcal{M}$, $(\mathrm{D}\mathsf{L}_x)_x = \mathrm{Id}$.

Based on the considerations above, generalized logarithms can be constructed as extensions of local inverses of retractions. In particular, we can think of Log as a generalized logarithm constructed from the retraction Exp.

Let us exhibit another way of constructing generalized logarithms on a smooth manifold $\mathcal{M}$. Given $x \in \mathcal{M}$ and a smooth scalar field $f_x : \mathcal{M} \to \mathbb{R}$, let us consider the smooth map

$$L_x : \mathcal{M} \to T_x\mathcal{M} : y \mapsto L_x(y) = f_x(y) \cdot \mathrm{Log}_x(y).$$

Exploiting the fact that Log is a generalized logarithm, we verify that $L_x(x) = 0$ and that

$$(\mathrm{D}L_x)_x = f_x(x) \cdot (\mathrm{D}\,\mathrm{Log}_x)(x) + \mathrm{Log}_x(x) \cdot \mathrm{D}f_x(x) = f_x(x) \cdot \mathrm{Id}$$

is the identity map on $T_x\mathcal{M}$ if and only if $f_x(x) = 1$. Hence, when $\mathrm{Log}_x(y)$ admits an expression in the form of a vector scaled by a complicated non-linear function, a generalized logarithm may sometimes be constructed by (carefully) getting rid of the scaling factor. We do this for the sphere in section 4.1.

2.8 Parallel translation

In Euclidean spaces, it is natural to compare vectors rooted at different points in space, so much that the notion of root of a vector is utterly unimportant. On manifolds, each tangent vector belongs to a tangent space specific to its root point. Vectors from different tangent spaces cannot be compared immediately. We need a mathematical tool capable of *transporting* vectors between tangent spaces while retaining the information they contain.

The proper tool from differential geometry for this is called *parallel translation*. Let us consider two points $x, y \in \mathcal{M}$, a vector $\xi \in T_x\mathcal{M}$ and a curve γ on $\mathcal{M}$ such that $\gamma(0) = x$ and $\gamma(1) = y$. We introduce X, a vector field defined along the trajectory of γ and such that $X_x = \xi$ and $\nabla_{\dot\gamma(t)} X(\gamma(t)) \equiv 0$. We say that X is *constant* along γ. The transported vector is X_y; it depends on γ.

In general, computing X_y requires one to solve a differential equation on $\mathcal{M}$. Just like we introduced retractions as a simpler proxy for exponentials and generalized logarithms for logarithms, we now introduce the concept of *vector transport* as a proxy for parallel translation. Again, this concept was first described by Absil et al. in [AMS08].

Roughly speaking, the notion of vector transport tells us how to transport a vector $\xi \in T_x\mathcal{M}$ from a point $x \in \mathcal{M}$ to a point $\mathsf{R}_x(\eta) \in \mathcal{M}$, $\eta \in T_x\mathcal{M}$. We first introduce the *Whitney sum* then quote the definition of vector transport.

$$T\mathcal{M} \oplus T\mathcal{M} = \{(\eta, \xi) : \eta, \xi \in T_x\mathcal{M}, x \in \mathcal{M}\}$$

Hence $T\mathcal{M} \oplus T\mathcal{M}$ is the set of pairs of tangent vectors belonging to a same tangent space. In the next definition, one of them will be the vector to transport and the other will be the vector along which to transport. This definition is illustrated on figure 2.5.

Definition 2.8.1 (vector transport)**.** *A* vector transport *on a manifold $\mathcal{M}$ is a smooth mapping*

$$T\mathcal{M} \oplus T\mathcal{M} \to T\mathcal{M} : (\eta, \xi) \mapsto \mathsf{T}_\eta(\xi) \in T\mathcal{M}$$

satisfying the following properties for all $x \in \mathcal{M}$:

(1) (associated retraction) There exists a retraction R, called the retraction associated with T, such that $\mathsf{T}_\eta(\xi) \in T_{\mathsf{R}_x(\eta)}\mathcal{M}$.

(2) (consistency) $\mathsf{T}_0(\xi) = \xi$ for all $\xi \in T_x\mathcal{M}$.

(3) (linearity) $\mathsf{T}_\eta(a\xi + b\zeta) = a\mathsf{T}_\eta(\xi) + b\mathsf{T}_\eta(\zeta)$, $\forall \eta, \xi, \zeta \in T_x\mathcal{M}, a, b \in \mathbb{R}$

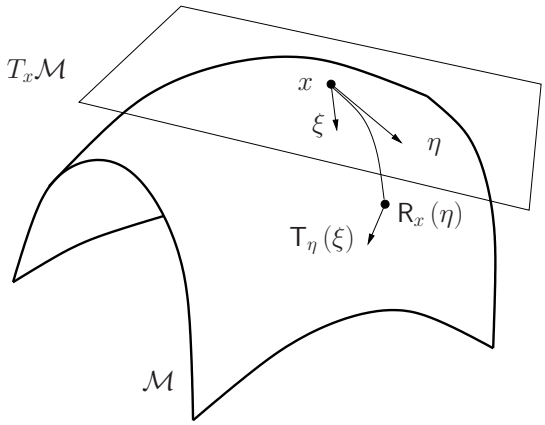

Figure 2.5: *Vector transport. Figure courtesy of Absil et al., [AMS08].*

In the following, we will use this definition to give a geometric version of the non-linear conjugate gradients method, section 3.5. We give the retraction and the associated transport we use on the sphere, chapter 4, in the next example taken from [AMS08, p. 172].

Example 2.8.2. *A valid retraction on the sphere $\mathbb{S}^2$ is given by:*

$$\mathsf{R}_x(v) = \frac{x+v}{\|x+v\|}.$$

An associated vector transport is:

$$\mathsf{T}_\eta(\xi) = \frac{1}{\|x+\eta\|}\left(I - \frac{(x+\eta)(x+\eta)^T}{(x+\eta)^T(x+\eta)}\right)\xi$$

On the right-hand side, x, η and ξ are to be treated as elements of $\mathbb{R}^3$. The scaling factor is not critical, according to definition 2.8.1. In our implementation for chapter 4, we chose to normalize such that $\|\mathsf{T}_\eta(\xi)\| = \|\xi\|$.

Chapter 3

Objective generalization and minimization

The second chapter gave us useful tools to replace the forbidden linear combinations that appear in the Euclidean case by reasonable equivalents in the more general setting of manifolds.

In this third chapter, we use these tools to introduce what we call geometric finite differences. These are then straightforwardly plugged into our original problem objective function to obtain a generalized objective. In what constitutes a second part of this chapter, we successively give a quick reminder of the steepest descent method and non-linear conjugate gradient methods for the minimization of a real valued function in $\mathbb{R}^n$. These reviews are followed by geometric versions, where we simply replaced every forbidden operation by its geometric counterpart. We close this chapter with a few suggestions for the step choosing algorithms. These algorithms need not be adapted, since in both the Euclidean and the Riemannian settings they are concerned with loosely minimizing real functions of one real variable.

Sadly, the theoretical convergence results we had for these methods in the Euclidean case do not all hold in our geometric setting, or at least cannot be proven as easily. Still, some useful results can be found in [AMS08]. As we will see in chapters 4 and 5, numerical results tend to be reassuring on these matters.

3.1 Geometric finite differences

We start by considering a smooth function $x : \mathbb{R} \to \mathbb{R}^n$. We assume that we know $x(t)$ at times t_0, $t_0 + \Delta t_\mathrm{f}$ and $t_0 - \Delta t_\mathrm{b}$ (the subscripts f and b stand for *forward* and *backward*). The goal is to derive a formula for the velocity $\dot{x}(t_0)$ and the acceleration $\ddot{x}(t_0)$ based on these values. As is customary, we write down the following Taylor expansions of x about t_0:

$$\begin{aligned} x(t_0 + \Delta t_\mathrm{f}) &= x(t_0) + \Delta t_\mathrm{f}\dot{x}(t_0) + \frac{\Delta t_\mathrm{f}^2}{2}\ddot{x}(t_0) + \mathcal{O}(\Delta t_\mathrm{f}^3) \\ x(t_0 - \Delta t_\mathrm{b}) &= x(t_0) - \Delta t_\mathrm{b}\dot{x}(t_0) + \frac{\Delta t_\mathrm{b}^2}{2}\ddot{x}(t_0) + \mathcal{O}(\Delta t_\mathrm{b}^3) \end{aligned} \tag{3.1}$$

We introduce some notations here to simplify the expressions:

$$\begin{gathered} x_\mathrm{f} = x(t_0 + \Delta t_\mathrm{f}),\ x_\mathrm{b} = x(t_0 - \Delta t_\mathrm{b}),\ x_0 = x(t_0), \\ \dot{x}_0 = \dot{x}(t_0),\ \ddot{x}_0 = \ddot{x}(t_0) \text{ and } \Delta t = \max(|\Delta t_\mathrm{f}|, |\Delta t_\mathrm{b}|). \end{gathered}$$

Rewriting, we get:

$$\begin{aligned} x_\mathrm{f} &= x_0 + \Delta t_\mathrm{f}\dot{x}_0 + \frac{\Delta t_\mathrm{f}^2}{2}\ddot{x}_0 + \mathcal{O}(\Delta t^3) \\ x_\mathrm{b} &= x_0 - \Delta t_\mathrm{b}\dot{x}_0 + \frac{\Delta t_\mathrm{b}^2}{2}\ddot{x}_0 + \mathcal{O}(\Delta t^3) \end{aligned} \tag{3.2}$$

This is a linear system in $\dot{x}_0$ and $\ddot{x}_0$. Solving it yields:

$$\begin{aligned} \dot{x}_0 &= \frac{1}{\Delta t_\mathrm{f} + \Delta t_\mathrm{b}} \frac{1}{\Delta t_\mathrm{f}\Delta t_\mathrm{b}} \left[\Delta t_\mathrm{b}^2(x_\mathrm{f} - x_0) - \Delta t_\mathrm{f}^2(x_\mathrm{b} - x_0)\right] + \mathcal{O}(\Delta t^2) \\ \ddot{x}_0 &= \frac{2}{\Delta t_\mathrm{f} + \Delta t_\mathrm{b}} \frac{1}{\Delta t_\mathrm{f}\Delta t_\mathrm{b}} \left[\Delta t_\mathrm{b}(x_\mathrm{f} - x_0) + \Delta t_\mathrm{f}(x_\mathrm{b} - x_0)\right] + \mathcal{O}(\Delta t) \end{aligned} \tag{3.3}$$

As expected, our formulas are linear combinations of the values x_b, x_0 and x_f. Arbitrary linear combinations do not, in general, make sense on manifolds, since manifolds do not, in general, have a vector space structure. However, we have assembled the terms of (3.3) in such a way that the differences $x_\mathrm{f} - x_0$ and $x_\mathrm{b} - x_0$ are apparent. Those may be interpreted as vectors rooted at x_0 and pointing toward, respectively, x_f and x_b. Using the logarithmic map introduced in section 2.7, we now propose a generalization of (3.3). We call the following geometric finite differences:

$$\begin{aligned} \dot{x}_0 &\approx \frac{1}{\Delta t_\mathrm{f} + \Delta t_\mathrm{b}} \frac{1}{\Delta t_\mathrm{f}\Delta t_\mathrm{b}} \left[\Delta t_\mathrm{b}^2 \operatorname{Log}_{x_0}(x_\mathrm{f}) - \Delta t_\mathrm{f}^2 \operatorname{Log}_{x_0}(x_\mathrm{b})\right] \\ \ddot{x}_0 &\approx \frac{2}{\Delta t_\mathrm{f} + \Delta t_\mathrm{b}} \frac{1}{\Delta t_\mathrm{f}\Delta t_\mathrm{b}} \left[\Delta t_\mathrm{b} \operatorname{Log}_{x_0}(x_\mathrm{f}) + \Delta t_\mathrm{f} \operatorname{Log}_{x_0}(x_\mathrm{b})\right] \end{aligned} \tag{3.4}$$

x_b, x_0 and x_f are now considered to be on a manifold $\mathcal{M}$. The objects $\dot{x}_0$ and $\ddot{x}_0$ are vectors in the tangent space $T_{x_0}\mathcal{M}$.

It is interesting to observe that the vector $\ddot{x}_0$ is zero if $x(t)$ is a geodesic curve, which is a desirable property. The converse is also true, i.e., if $\ddot{x}_0 = 0$, then there exists a geodesic $\gamma(t)$ fitting the data. We give an outline of a proof. Simply build γ as follows:

$$\gamma(t) = \begin{cases} \operatorname{Exp}_{x_0}\left(\frac{t_0 - t}{\Delta t_\mathrm{b}} \operatorname{Log}_{x_0}(x_\mathrm{b})\right) & \text{if } t < t_0, \\ \operatorname{Exp}_{x_0}\left(\frac{t - t_0}{\Delta t_\mathrm{f}} \operatorname{Log}_{x_0}(x_\mathrm{f})\right) & \text{if } t \geq t_0. \end{cases}$$

By construction, γ is a geodesic before and after t_0 and is continuous. γ is also globally a geodesic. Indeed, we can compute the velocity vector $\dot{\gamma}(t_0)$ "coming from the left":

$$\dot{\gamma}(t_0^-) = \frac{-1}{\Delta t_\mathrm{b}} \operatorname{Log}_{x_0}(x_\mathrm{b}),$$

and "coming from the right":

$$\dot{\gamma}(t_0^+) = \frac{1}{\Delta t_\mathrm{f}} \operatorname{Log}_{x_0}(x_\mathrm{f}).$$

Assuming $\ddot{x}_0 = 0$, these two velocity vectors agree, hence γ is a geodesic.

Similar developments yield unilateral finite differences for end points. Considering $x_0 = x(t_0)$, $x_1 = x(t_0 + \Delta t_1)$ and $x_2 = x(t_0 + \Delta t_1 + \Delta t_2)$ and defining $\Delta t = \max(|\Delta t_1|, |\Delta t_2|)$, we get:

$$\begin{aligned}
\dot{x}_0 &\approx \frac{\Delta t_1 + \Delta t_2}{\Delta t_1 \Delta t_2} \operatorname{Log}_{x_0}(x_1) - \frac{\Delta t_1}{\Delta t_2(\Delta t_1 + \Delta t_2)} \operatorname{Log}_{x_0}(x_2) \\
\ddot{x}_0 &\approx \frac{-2}{\Delta t_1 \Delta t_2} \operatorname{Log}_{x_0}(x_1) + \frac{2}{\Delta t_2(\Delta t_1 + \Delta t_2)} \operatorname{Log}_{x_0}(x_2)
\end{aligned} \tag{3.5}$$

Since the formulas for the acceleration are only of order 1, we may as well derive order 1 formulas for the velocity. Either of the following is fine:

$$\begin{aligned}
\dot{x}_0 &\approx \frac{1}{\Delta t_\mathrm{f}} \operatorname{Log}_{x_0}(x_\mathrm{f}) \\
\dot{x}_0 &\approx \frac{-1}{\Delta t_\mathrm{b}} \operatorname{Log}_{x_0}(x_\mathrm{b})
\end{aligned}$$

We generalized finite differences in a way that makes sense on any smooth manifold equipped with a logarithm. We built this generalization based on our interpretation of the linear combinations appearing in the classical finite differences used in the discrete objective, equation (1.6). Instead, we could have tried to directly discretize terms such as $\frac{\mathrm{D}^2}{\mathrm{d}t^2}\gamma(t)$, appearing in the generalized continuous objective, equation (1.2). In the case of Riemannian submanifolds of $\mathbb{R}^n$, the definition 2.5.6 for the acceleration would have led us to compute the finite differences in the ambient space, then project the result back on the tangent space. As it turns out, on the sphere, this is equivalent to computing the geometric finite differences with a particular generalized logarithm, definition 2.7.5. We use that generalized logarithm in chapter 4.

3.2 Generalized objective

In section 1.2, we defined an objective function for discrete curve fitting in Euclidean spaces. Equations (1.3), (1.4) and (1.5) used classical tools such as Euclidean distances and finite differences, the latter being used to evaluate velocity and acceleration along a discrete curve. We now have new tools to rewrite these elements in a very similar form:

$$\begin{aligned}
E(\gamma) &= E_\mathrm{d}(\gamma) + \lambda E_\mathrm{v}(\gamma) + \mu E_\mathrm{a}(\gamma) \\
E_\mathrm{d}(\gamma) &= \frac{1}{2} \sum_{i=1}^{N} w_i \operatorname{dist}^2(p_i, \gamma_{s_i}) \\
E_\mathrm{v}(\gamma) &= \frac{1}{2} \sum_{i=1}^{N_\mathrm{d}} \alpha_i \|v_i\|_{\gamma_i}^2 \\
E_\mathrm{a}(\gamma) &= \frac{1}{2} \sum_{i=1}^{N_\mathrm{d}} \beta_i \|a_i\|_{\gamma_i}^2
\end{aligned} \tag{3.6}$$

This time, the data $p_1, \ldots, p_N$ lives on a manifold $\mathcal{M}$. The function E and its components E_d, E_v and E_a are defined over a curve space $\Gamma = \mathcal{M} \times \ldots \times \mathcal{M}$, which is simply N_d copies of the manifold $\mathcal{M}$. It is a product manifold, and as such inherits many of the properties of $\mathcal{M}$. In particular, if $\mathcal{M}$ is finite dimensional, which will always be the case in this document, then so is Γ. This is useful because the definition of the gradient, definition 2.4.2, applies.

The distance $\operatorname{dist}(\cdot, \cdot)$ is the geodesic distance on $\mathcal{M}$ (if available), or a proxy for it. The tangent vectors for velocity v_i and acceleration a_i at points γ_i are computed according to the formulas given in section 3.1. If the logarithmic map needed for geometric finite differences has

a too complicated form, a proxy for it can be used too. If even coming up with a generalized logarithm proves difficult, one can envision the alternative generalized objectives described in section 3.3. To the best of our knowledge, this is a novel objective function for the regression problem on manifolds.

Other generalizations can be proposed. We will discuss some of them in the next section. Interestingly, these alternative formulations all come down to finite differences in the Euclidean case, although they may not be equivalent in the general, Riemannian case.

3.3 Alternative discretizations of the objective*

In this optional section, we propose two alternative discretizations of the second order term in (1.2),

$$\int_{t_1}^{t_N} \left\langle \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2}, \frac{\mathrm{D}^2\gamma}{\mathrm{d}t^2} \right\rangle_{\gamma(t)} \mathrm{d}t. \tag{3.7}$$

The geometric finite differences make for clean developments and can be used to derive higher-order formulas in a straightforward way. The discretizations we present here are limited to the second order. As we will see, they both come down to finite differences when specialized to the Euclidean case, but differ on manifolds. We give explicit formulas on the sphere $\mathbb{S}^2$ to illustrate this. We introduce $\mathbb{S}^2$ as a manifold in section 4.1. The reader may want to read that section first.

Both our alternatives are based on the idea that (3.7) penalizes departure from straightness. Let us consider three points on a manifold γ_1, γ_2 and γ_3 associated with times τ_1, τ_2 and τ_3. Define $\Delta\tau_1 = \tau_2 - \tau_1$ and $\Delta\tau_2 = \tau_3 - \tau_2$. If there was a geodesic $\gamma(t)$ such that $\gamma(\tau_i) = \gamma_i$, $\forall i \in \{1, 2, 3\}$ and such that the length of γ between τ_i and τ_{i+1} equals the geodesic distance between γ_i and γ_{i+1}, $\forall i \in \{1, 2\}$, then γ_2 would be equal to

$$\gamma_* = \mathrm{Exp}_{\gamma_1}\left(\frac{\Delta\tau_1}{\Delta\tau_1 + \Delta\tau_2}\,\mathrm{Log}_{\gamma_1}(\gamma_3)\right),$$

and γ_3 would be equal to

$$\gamma_\star = \mathrm{Exp}_{\gamma_1}\left(\frac{\Delta\tau_1 + \Delta\tau_2}{\Delta\tau_1}\,\mathrm{Log}_{\gamma_1}(\gamma_2)\right).$$

Our first alternative penalizes terms of the form $\mathrm{dist}^2(\gamma_2, \gamma_*)$ whereas our second alternative penalizes terms of the form $\mathrm{dist}^2(\gamma_3, \gamma_\star)$. Let us consider $\Delta\tau_1 = \Delta\tau_2 \triangleq \Delta\tau$ for ease of notations. In this particular case, we could define γ_* as the *mean* of γ_1 and γ_3. Specializing the penalties to a Euclidean space, we obtain:

$$\begin{aligned}
\mathrm{dist}^2(\gamma_2, \gamma_*) &= \left\| \left(\gamma_1 + \frac{1}{2}(\gamma_3 - \gamma_1)\right) - \gamma_2 \right\|^2 = \frac{1}{4}\left\|\gamma_1 - 2\gamma_2 + \gamma_3\right\|^2 \\
\mathrm{dist}^2(\gamma_3, \gamma_\star) &= \left\|(\gamma_1 + 2(\gamma_2 - \gamma_1)) - \gamma_3\right\|^2 = \left\|\gamma_1 - 2\gamma_2 + \gamma_3\right\|^2
\end{aligned}$$

We recognize classical finite differences for acceleration at γ_2. Hence, these penalties certainly make sense -and are equivalent up to a scaling factor- in a Euclidean space. They are not equivalent on the sphere. We show this explicitly.

Let $(x, y) \mapsto \langle x, y\rangle = x^T y$ be our inner product on $\mathbb{R}^3$. When $\gamma_i \in \mathbb{S}^2 = \{x \in \mathbb{R}^3 : \langle x, x\rangle = 1\}$, $\forall i \in \{1, 2, 3\}$, γ_* can be computed as:

$$\gamma_* = \mathrm{mean}(\gamma_1, \gamma_3) \triangleq \frac{\gamma_1 + \gamma_3}{\|\gamma_1 + \gamma_3\|} = \frac{\gamma_1 + \gamma_3}{\sqrt{2\,(1 + \langle\gamma_1, \gamma_3\rangle)}}.$$

Since $\mathrm{dist}^2(x, y) = \arccos^2(\langle x, y\rangle)$ on $\mathbb{S}^2$ (see table 4.1), the associated penalty is

$$\mathrm{dist}^2(\gamma_2, \gamma_*) = \arccos^2\left(\frac{\langle\gamma_2, \gamma_1 + \gamma_3\rangle}{\sqrt{2\,(1 + \langle\gamma_1, \gamma_3\rangle)}}\right).$$

This is a rather involved function considering that we will need to differentiate it. We now compute $\gamma_\star$. The latter is a point on the sphere lying in the plane defined by γ_1 and γ_2 and such that the geodesic distance, i.e., the angle between γ_1 and γ_2 equals the angle between γ_2 and $\gamma_\star$. We can formalize this as:

$$\begin{aligned}\langle \gamma_\star, \gamma_\star \rangle &= 1 \\ \gamma_\star &= \alpha_1 \gamma_1 + \alpha_2 \gamma_2 \\ \langle \gamma_1, \gamma_2 \rangle &= \langle \gamma_2, \gamma_\star \rangle\end{aligned}$$

This system in (α_1, α_2) has two solutions. Either $\gamma_\star = \gamma_1$, which we discard as not interesting, or:

$$\gamma_\star = 2 \langle \gamma_1, \gamma_2 \rangle \gamma_2 - \gamma_1.$$

The associated penalty takes the nice form:

$$\operatorname{dist}^2(\gamma_3, \gamma_\star) = \arccos^2\left(2 \langle \gamma_1, \gamma_2 \rangle \langle \gamma_2, \gamma_3 \rangle - \langle \gamma_1, \gamma_3 \rangle\right).$$

We would like to exploit this to propose a cheap discretization of (3.7) on $\mathbb{S}^2$ for uniform time sampling. When consecutive γ_i's are close to each other, the geodesic distance can be approximated by the Euclidean distance in the ambient space $\mathbb{R}^3$. We have:

$$\|a - b\|^2 = 2(1 - \langle a, b \rangle), \quad a, b \in \mathbb{S}^2$$

Using this instead of $\arccos^2(\langle a, b \rangle)$ (this corresponds to a first order Taylor approximation), we propose a simple form for E_a on $\mathbb{S}^2$:

$$E_\mathrm{a} = \frac{1}{\Delta\tau^3} \sum_{i=2}^{N_\mathrm{d}-1} 1 - 2 \langle \gamma_{i-1}, \gamma_i \rangle \langle \gamma_i, \gamma_{i+1} \rangle + \langle \gamma_{i-1}, \gamma_{i+1} \rangle \tag{3.8}$$

We neglected the endpoints, which is not a liability since this formula is intended for fine time sampling. The $1/\Delta\tau^3$ factor is explained as follows. In the Euclidean case, the penalty is given by:

$$E_\mathrm{a} = \frac{1}{2} \sum_{i=2}^{N_\mathrm{d}-1} \Delta\tau \left\| \frac{\gamma_{i+1} - 2\gamma_i + \gamma_{i-1}}{\Delta\tau^2} \right\|^2 = \frac{1}{2} \sum_{i=2}^{N_\mathrm{d}-1} \frac{1}{\Delta\tau^3} \|\gamma_{i+1} - 2\gamma_i + \gamma_{i-1}\|^2$$

As we showed earlier, in the Euclidean case, $\operatorname{dist}^2(\gamma_3, \gamma_\star) = \|\gamma_3 - 2\gamma_2 + \gamma_1\|^2$, hence we identified corresponding terms. The derivatives of (3.8), seen as a scalar field in the ambient space $\mathbb{R}^{3 \times N_\mathrm{d}}$, are given by:

$$\begin{aligned}\Delta\tau^3 \frac{\partial E_\mathrm{a}}{\partial \gamma_1} &= \gamma_3 - 2 \langle \gamma_2, \gamma_3 \rangle \gamma_2 \\ \Delta\tau^3 \frac{\partial E_\mathrm{a}}{\partial \gamma_2} &= \gamma_4 - 2(\langle \gamma_1, \gamma_2 \rangle + \langle \gamma_3, \gamma_4 \rangle)\gamma_3 - 2 \langle \gamma_2, \gamma_3 \rangle \gamma_1 \\ \Delta\tau^3 \frac{\partial E_\mathrm{a}}{\partial \gamma_k} &= \gamma_{k+2} - 2(\langle \gamma_{k-1}, \gamma_k \rangle + \langle \gamma_{k+1}, \gamma_{k+2} \rangle)\gamma_{k+1} \\ &\quad + \gamma_{k-2} - 2(\langle \gamma_{k-2}, \gamma_{k-1} \rangle + \langle \gamma_k, \gamma_{k+1} \rangle)\gamma_{k-1}, \quad \forall k \in \{3, \dots, N_\mathrm{d} - 2\} \\ \Delta\tau^3 \frac{\partial E_\mathrm{a}}{\partial \gamma_{N_\mathrm{d}-1}} &= \gamma_{N_\mathrm{d}-3} - 2(\langle \gamma_{N_\mathrm{d}-3}, \gamma_{N_\mathrm{d}-2} \rangle + \langle \gamma_{N_\mathrm{d}-1}, \gamma_{N_\mathrm{d}} \rangle)\gamma_{N_\mathrm{d}-2} - 2 \langle \gamma_{N_\mathrm{d}-2}, \gamma_{N_\mathrm{d}-1} \rangle \gamma_{N_\mathrm{d}} \\ \Delta\tau^3 \frac{\partial E_\mathrm{a}}{\partial \gamma_{N_\mathrm{d}}} &= \gamma_{N_\mathrm{d}-2} - 2 \langle \gamma_{N_\mathrm{d}-2}, \gamma_{N_\mathrm{d}-1} \rangle \gamma_{N_\mathrm{d}-1}\end{aligned}$$

Both E_a and its gradient can thus be computed with simple inner products of close points on $\mathbb{S}^2$ and elementary operations such as sums, subtractions and multiplication. This is cheap and reliable. Regretfully, this nice property is lost when we try to generalize to non-homogeneous time sampling.

3.4 A geometric steepest descent algorithm

The steepest descent algorithm is probably the most straightforward means of minimizing a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ without constraints. Following Nocedal and Wright in [NW99], we give a reminder of the method in algorithm 1. We delay the details of how one can choose the step length to section 3.6. Of course, whether or not the sequence generated by algorithm 1 converges to a critical point of f (i.e., a point x^* such that $\nabla f(x^*) = 0$) depends on the step choosing algorithm. The initial guess can be constructed based on a priori knowledge of the problem at hand. For practical purposes, one should, of course, integrate a stopping criterion into the algorithm.

Algorithm 1: Steepest descent method in $\mathbb{R}^n$

input : A function $f : \mathbb{R}^n \to \mathbb{R}$ and its gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$;
An initial guess $x_0 \in \mathbb{R}^n$.
output: A sequence $x_0, x_1, \ldots$ converging toward a critical point of f.
for $k = 0, 1, \ldots$ **do**
  if $\|\nabla f(x_k)\| \neq 0$ **then**
    $d_k := -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$
    $\alpha_k :=$ choose step length
    $x_{k+1} := x_k + \alpha_k d_k$
  else
    $x_{k+1} = x_k$
  end
end

We would like to use the tools developed in the previous chapter to generalize algorithm 1 to a scalar field f on a manifold $\mathcal{M}$. We already defined the gradient $\operatorname{grad} f : \mathcal{M} \to T\mathcal{M}$ in definition 2.4.2. What we still need is the right interpretation for the update line of the steepest descent algorithm, namely $x_{k+1} := x_k + \alpha_k d_k$. In the geometric context, x_k is a point on the manifold and $\alpha_k d_k$ is a tangent vector to $\mathcal{M}$ at x_k. The summation of these two elements is undefined. However, the meaning of the sum is clear: starting at x_k, we follow the vector $\alpha_k d_k$ until we reach a new point, which we name x_{k+1}. Retractions, definition 2.7.2, and in particular the exponential map do exactly this. We can thus rewrite the algorithm as algorithm 2, in very much the same way.

Algorithm 2: Geometric steepest descent method on $\mathcal{M}$

input : A scalar field $f : \mathcal{M} \to \mathbb{R}$ and its gradient $\operatorname{grad} f : \mathcal{M} \to T\mathcal{M}$;
A retraction R on $\mathcal{M}$; An initial guess x_0 on $\mathcal{M}$.
output: A sequence $x_0, x_1, \ldots$ converging toward a critical point of f.
for $k = 0, 1, \ldots$ **do**
  if $\|\operatorname{grad} f(x_k)\|_{x_k} \neq 0$ **then**
    $d_k := -\frac{\operatorname{grad} f(x_k)}{\|\operatorname{grad} f(x_k)\|_{x_k}}$
    $\alpha_k :=$ choose step length
    $x_{k+1} := \mathsf{R}_{x_k}(\alpha_k d_k)$
  else
    $x_{k+1} = x_k$
  end
end

Such a generalization can be found in [AMS08], along with proper analysis giving, notably, sufficient conditions to guarantee convergence toward a critical point of the scalar field f. Such a treatment of the method is not our main focus in this document. We thus propose to simply state the relevant theorem from [AMS08] without proof. The following results are concerned with general descent methods, i.e., d_k is not forced to point in the opposite direction of the gradient.

Global convergence results then depend upon the directions sequence $d_0, d_1, \ldots$ as well as on the step lengths sequence $\alpha_0, \alpha_1, \ldots$

Definition 3.4.1 (gradient-related sequence)**.** *Let f be a scalar field on $\mathcal{M}$. The sequence $\{d_0, d_1, \ldots\}$, $d_k \in T_{x_k}\mathcal{M}$, is* gradient-related *if, for any subsequence $\{x_k\}_{k\in K}$ of $\{x_k\}$ that converges to a non-critical point of f, the corresponding subsequence $\{d_k\}_{k\in K}$ is bounded and satisfies*

$$\limsup_{k\to\infty, k\in K} \langle \operatorname{grad} f(x_k), d_k \rangle_{x_k} < 0.$$

The above definition essentially forces the directions d_k to be descent directions *in the end*. Obviously, the steepest descent sequence

$$\left\{ d_k = -\frac{\operatorname{grad} f(x_k)}{\| \operatorname{grad} f(x_k) \|} \right\}$$

is gradient-related. The next definition, based on Armijo's backtracking procedure, forces the decrease in f to be sufficient at each step (and simultaneously defines what *sufficient* means), based on the steepness of the chosen direction. It differs slightly from [AMS08].

Definition 3.4.2 (Armijo point)**.** *Given a cost function f on $\mathcal{M}$ with retraction R, a point $x \in \mathcal{M}$, a tangent vector $d \in T_x\mathcal{M}$ and scalars $\overline{\alpha} > 0, \beta, \sigma \in (0,1)$, the* Armijo point *is $\mathsf{R}_x\left(d^A\right) = \mathsf{R}_x\left(t^A d\right) = \mathsf{R}_x\left(\beta^m \overline{\alpha} d\right)$, where m is the smallest nonnegative integer such that*

$$f(x) - f(\mathsf{R}_x(\beta^m \overline{\alpha} d)) \geq \sigma \left\langle -\operatorname{grad} f(x), \beta^m \overline{\alpha} d \right\rangle_x$$

The real number t^A is the Armijo step size*.*

When d is a descent direction, such a point and step size are guaranteed to exist. The next theorem uses both these definitions to make a statement about the convergence properties of variations of algorithm 2.

Theorem 3.4.3. *Let $\{x_k\}$ be an infinite sequence of iterates generated by a variation of algorithm 2 such that the sequence $\{d_k\}$ is gradient-related and such that, at each step, $f(x_k) - f(x_{k+1}) \geq c(f(x_k) - f(\mathsf{R}_{x_k}\left(t_k^A d_k\right)))$, where $c \in (0,1)$ and t_k^A is the Armijo step size for given parameters. Further assume that the initial level set $\mathcal{L} = \{x \in \mathcal{M} : f(x) \leq f(x_0)\}$ is compact (which is certainly the case if $\mathcal{M}$ itself is compact). Then,*

$$\lim_{k\to\infty} \| \operatorname{grad} f(x_k) \| = 0$$

We do not need to use the Armijo step size per se. Any step choosing algorithm assuring sufficient descent will do. The Armijo point, though, is a nice way of defining *sufficient* descent. It also makes it possible to prove theorem 3.4.3 fairly independently of the actual directions and steps used. In the end, one needs to provide a problem-specific, efficient algorithm, as we will do in the application chapters.

3.5 A geometric non-linear conjugate gradient algorithm

In $\mathbb{R}^n$, the steepest descent algorithm is known for its linear convergence rate. It should not come as a surprise that the geometric version of it, which we presented in the previous section, is equally slow (although it is a bit more technical to establish it). In order to achieve faster convergence rates, one could resort to second-order methods, like Newton or quasi-Newton schemes. This is unpractical (but not impossible) in a geometric setting, since it requires a proper definition of the Hessian of a scalar field f, i.e., the derivative of a vector field, as well as a means to compute it. The affine connections defined in section 2.5 are the right tool, but do not lend themselves to easy calculations. The objective functions we deal with in this document being quite involved, we decided to postpone such investigations to later work and to focus on, hopefully, superlinear schemes based on first order derivatives only.

The non-linear conjugate gradient method is such a scheme in $\mathbb{R}^n$. Algorithm 3 is Nocedal and Wright's version of it, see [NW99], based on Fletcher-Reeves coefficients. The unit-length update directions d_k have been introduced so that the step length chosen would always correspond to the actual displacement vector's length. This way, it is easier to compare the steepest descent step lengths and the NLCG step lengths.

Algorithm 3: Non-linear conjugate gradient method in $\mathbb{R}^n$

input: A function $f : \mathbb{R}^n \to \mathbb{R}$ and its gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$;
An initial guess $x_0 \in \mathbb{R}^n$.
Evaluate $f_0 = f(x_0), \nabla f_0 = \nabla f(x_0)$
$p_0 := -\nabla f_0$
$k := 0$
while $\nabla f_k \neq 0$ **do**
$d_k = \frac{p_k}{\|p_k\|}$
$\alpha_k :=$ choose step length
$x_{k+1} := x_k + \alpha_k d_k$
Evaluate $\nabla f_{k+1} = \nabla f(x_{k+1})$
$\beta_{k+1}^{\mathrm{FR}} := \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\|\nabla f_k\|^2}$
$p_{k+1} := -\nabla f_{k+1} + \beta_{k+1}^{\mathrm{FR}} p_k$
$k := k + 1$
end

An alternative algorithm, termed the Polak-Ribière version of algorithm 3, uses the following β's:

$$\beta_{k+1}^{\mathrm{PR}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2}$$

The update directions d_k are not guaranteed to be descent directions unless we impose some constraints on the α_k's. Indeed, we have:

$$\langle p_k, \nabla f_k \rangle = -\|\nabla f_k\|^2 + \beta_k \langle p_{k-1}, \nabla f_k \rangle ,$$

which needs to be negative for d_k to be a descent direction. This can be guaranteed by using the well-known strong Wolfe conditions on the α_k's. For a step length choosing algorithm which cannot guarantee a descent direction for the NLCG method, an easy workaround is to accomplish a steepest descent iteration every time p_k is not a descent direction. Theorem 3.4.3 tells us that such an algorithm would still converge. Interestingly, the particular choice $\beta_k = 0$ makes the algorithm revert to the steepest descent method altogether.

In [NW99], Nocedal and Wright mention a wealth of variations on algorithm 3 as well as useful and interesting results concerning convergence and numerical performances. It is not our aim to provide an exhaustive treatment of CG algorithms in this document. In the application chapters later on, we will illustrate performances of the methods outlined here.

We now generalize algorithm 3 to manifolds. Such a generalization can be found in [AMS08]. The update step $x_{k+1} := x_k + \alpha_k d_k$ can be modified in the very same way we did for algorithm 2. The direction update though, $p_{k+1} := -\nabla f_{k+1} + \beta_{k+1} p_k$, is tricky. Indeed, p_{k+1} and $\operatorname{grad} f_{k+1}$ live in $T_{x_{k+1}}\mathcal{M}$ but p_k is in $T_{x_k}\mathcal{M}$. Hence, the sum is undefined. The vector transport concept, definition 2.8.1, comes in handy here. It enables us to transport vector p_k from $T_{x_k}\mathcal{M}$ to $T_{x_{k+1}}\mathcal{M}$, hence giving us a reasonable substitute for which the sum with $-\operatorname{grad} f_{k+1}$ is defined. Same thing goes for the β_k's:

$$\begin{aligned} \beta_{k+1}^{\mathrm{FR}} &= \frac{\langle \operatorname{grad} f_{k+1}, \operatorname{grad} f_{k+1} \rangle}{\| \operatorname{grad} f_k \|^2} \\ \beta_{k+1}^{\mathrm{PR}} &= \frac{\langle \operatorname{grad} f_{k+1}, \operatorname{grad} f_{k+1} - \mathsf{T}_{\alpha_k d_k}(\operatorname{grad} f_k) \rangle}{\| \operatorname{grad} f_k \|^2} \end{aligned} \tag{3.9}$$

Algorithm 4 synthesizes these remarks.

Algorithm 4: Geometric non-linear conjugate gradient method on $\mathcal{M}$

input: A scalar field $f : \mathcal{M} \to \mathbb{R}$ and its gradient $\operatorname{grad} f : \mathcal{M} \to T\mathcal{M}$;
An initial guess $x_0 \in \mathcal{M}$.
Evaluate $f_0 = f(x_0), \operatorname{grad} f_0 = \operatorname{grad} f(x_0)$
$p_0 := -\operatorname{grad} f_0$
$k := 0$
while $\operatorname{grad} f_k \neq 0$ **do**
  $d_k = \frac{p_k}{\|p_k\|}$
  $\alpha_k :=$ choose step length
  $x_{k+1} := \mathsf{R}_{x_k}(\alpha_k d_k)$
  Evaluate $\operatorname{grad} f_{k+1} = \operatorname{grad} f(x_{k+1})$
  $\beta_{k+1} :=$ cf. equation (3.9)
  $p_{k+1} := -\operatorname{grad} f_{k+1} + \beta_{k+1}\mathsf{T}_{\alpha_k d_k}(p_k)$
  $k := k + 1$
end

When minimizing a quadratic objective in n variables, the CG method converges to the exact solution in at most n iterations. It is therefore customary to *restart* the non-linear CG method every n iterations or so, by applying a steepest descent step (which boils down to setting $\beta_k = 0$). This way, old, irrelevant information is erased and the algorithm can start anew. In particular, if the objective is globally convex and is a strictly convex quadratic in a neighborhood about the minimizer, the algorithm will eventually enter that neighborhood and restart in it, hence reverting to a linear CG method and converging in at most n additional steps. In our algorithms, we decide to restart every n iterations where n is the dimension of $\mathcal{M}$.

Non-linear CG methods and their convergence properties are not well understood in $\mathbb{R}^n$, let alone on manifolds. We thus stop here, and rely on numerical tests in the application chapters to assess the performances of algorithm 4.

3.6 Step choosing algorithms

To choose the step length in algorithms 2 and 4, we need to solve a line search problem, i.e., compute

$$\alpha^* = \arg\min_{\alpha>0} \phi(\alpha) = \arg\min_{\alpha>0} f(\mathsf{R}_x(\alpha p)).$$

Despite the fact that our original problem uses manifolds, the line search problem only uses real functions: we need to (crudely) minimize a function $\phi : \mathbb{R} \to \mathbb{R}$, knowing that $\phi'(0) < 0$ (descent direction). A lot of work has been done for this problem in the optimization community. Because of this, we keep the present section to a minimum, referring the reader to, e.g., [NW99].

We single out one algorithm that proved particularly interesting in our case. Since the regression problem is quadratic in $\mathbb{R}^n$, we expect the objective function to "look like" a quadratic in a neighborhood around the minimizer on a manifold, precisely because manifolds locally resemble $\mathbb{R}^n$. Hence, it makes sense to expect the line search functions to be almost quadratic around $\alpha = 0$ when we get near the minimum of f. This is indeed the case when working on the sphere. These considerations led us to choose algorithm 5 as our chief step choosing algorithm. It is based on an Armijo backtracking technique with automatic choice of the initial guess α_1.

In practice, we add cubic interpolation to algorithm 5 as well for $i \geq 2$ and check that the sequence $(\alpha_1, \alpha_2, \ldots)$ does not decrease too rapidly, along with a few other ad hoc numerical considerations. Following Nocedal and Wright in [NW99], we picked $c = 10^{-4}$. We also implemented algorithms ensuring respect of the Wolfe conditions from [NW99] and actual minimization of ϕ

Algorithm 5: Backtracking-interpolation step choosing algorithm

require: A small constant $c \in (0, 1)$, an iteration limit N, a line search function ϕ and its derivative ϕ' (only computed at 0).
input : The previous step chosen α_{prev} and the previous slope $\phi'_{0,\text{prev}}$.
output: A step α respecting the sufficient decrease condition $\phi(0) - \phi(\alpha) \geq -c\alpha\phi'(0)$, or 0 if N iterations did not suffice.

Compute $\phi_0 := \phi(0)$ and $\phi'_0 := \phi'(0)$
Set $\alpha_1 := 2\alpha_{\text{prev}} \frac{\phi'_{0,\text{prev}}}{\phi'_0}$
for $i = 1, 2, \ldots$ **do**
  if $i > N$ **then**
    return 0
  else
    Compute $\phi_i = \phi(\alpha_i)$
    if $\phi_0 - \phi_i \geq -c\alpha_i\phi'_0$ **then**
      return α_i
    else
      $\alpha_{i+1} = -\frac{\alpha_i^2\phi'_0}{2\left(\phi_i - \phi_0 - \alpha_i\phi'_0\right)}$
    end
  end
end

with Matlab's fminsearch algorithm. These performed well too, but we do not use them for the results shown in chapters 4 and 5.

Chapter 4

Discrete curve fitting on the sphere

The previous chapter constitutes a formal definition of the regression problem we intend to solve on manifolds and of minimization algorithms on manifolds. These algorithms could, in principle, be used to solve the regression problem at hand. Still, tractability has not been assessed yet and is a fundamental factor of success for any numerical method. Mostly, tractability will be determined by:

- the complexity of the formulas for geodesic distances and logarithmic maps as well as their derivatives,
- the convergence rate of the minimization algorithms, and
- the amount of data as well as the sampling precision required.

The intricacy of writing the code (mostly inherent to the complexity of working out the gradient of the objective) and the actual performances are shown to be reasonable on the sphere. It will be clear, however, that the results of this chapter do not transpose to every manifold.

One of the main arguments justifying the geometric approach instead of, e.g., a direct use of spherical coordinates, is the absence of any type of singularity problem at the poles. In this chapter, we give explicit formulas for all the elements needed to apply the non-linear conjugate gradient scheme to discrete curve fitting on the sphere $\mathbb{S}^2$, i.e., the unit sphere embedded in $\mathbb{R}^3$. As we will see, the difficulty of writing an algorithm to solve the problem can be reduced by using proxies for, e.g., the logarithmic map. This can have an influence on the objective function's global minimum, but this influence can be made arbitrarily small by refining the curve sampling. We then show some results for different scenarios and discuss performances as well as tricks to speed things up a bit.

Set:	$\mathbb{S}^2 = \{x \in \mathbb{R}^3 \ : \ x^T x = 1\}$
Tangent spaces:	$T_x\mathbb{S}^2 = \{v \in \mathbb{R}^3 \ : \ x^T v = 0\}$
Projector:	$P_x : \mathbb{R}^3 \to T_x\mathbb{S}^2 : v \mapsto P_x(v) = (I - xx^T)v$
Inner product:	$\langle v, w \rangle_x = \langle v, w \rangle = v^T w$ (induced metric)
Vector norm:	$\|v\|_x = \sqrt{\langle v, v \rangle_x} = \|v\|$
Distance:	$\operatorname{dist}(x, y) = \arccos(x^T y)$
Exponential:	$\operatorname{Exp}_x(v) = x\cos(\|v\|) + \frac{v}{\|v\|}\sin(\|v\|)$
Logarithm:	$\operatorname{Log}_x(y) = \frac{\operatorname{dist}(x,y)}{\|P_x(y-x)\|} P_x(y - x)$
Mean:	$\operatorname{mean}(x, y) = \frac{x+y}{\|x+y\|}$

Table 4.1: *Geometric toolbox for the Riemannian manifold* $\mathbb{S}^2$

4.1 $\mathbb{S}^2$ geometric toolbox

We define the set $\mathbb{S}^2$, the unit sphere embedded in $\mathbb{R}^3$, and give it a natural Riemannian manifold structure inherited from $\mathbb{R}^3$ (i.e., $\mathbb{S}^2$ is a submanifold of the Euclidean space $\mathbb{R}^3$). From now on, we indiscriminately use the symbol $\mathbb{S}^2$ to refer to the set or the manifold, just like we often do when we use the symbol $\mathcal{M}$ instead of the actual set the manifold is built upon and vice versa. Table 4.1 contains a few closed-form expressions for the geometric tools we will use henceforth. In the next few paragraphs, we go over these formulas and give a word of explanation.

$\mathbb{S}^2$ being a submanifold of $\mathbb{R}^3$ defined by an equality constraint, theorem 2.2.2 applies and we are entitled to use the identification between tangent vectors and elements of $\mathbb{R}^3$, definition 2.2.1. Defining $f : \mathbb{R}^3 \to \mathbb{R} : x \mapsto f(x) = x^T x - 1$, the differential $\mathrm{d}f_x = 2x^T$ yields $\ker \mathrm{d}f_x = T_x\mathbb{S}^2 = \{v \in \mathbb{R}^3 \ : \ x^T v = 0\}$. We thus think of tangent vectors as actual vectors of $\mathbb{R}^3$ tangent to the sphere.

It is readily checked that $P_x \circ P_x = P_x$ and that $P_x = P_x^T$ (i.e., P_x is an orthogonal projector). Furthermore, $\operatorname{Im} P_x = T_x\mathbb{S}^2$ and $\ker P_x = T_x^\perp\mathbb{S}^2$ (the orthogonal complement of $T_x\mathbb{S}^2$). The Riemannian metric as well as the norm are inherited from $\mathbb{R}^3$ because of the submanifold nature of $\mathbb{S}^2$.

The geodesic distance between p and q, almost by definition, is the angle separating p and q in radians. It corresponds to the length of the shortest arc of great circle joining the two points. Great circles are geodesic curves for $\mathbb{S}^2$. A great circle is a unit-radius circle centered at the origin. To simplify, when computing the distance between close points, one could use the distance in the embedding space: $\|x - y\|^2 = 2(1 - x^T y)$ for $x, y \in \mathbb{S}^2$. Proxies for the geodesic distance are compared on figure 4.1.

The curve $t \mapsto \operatorname{Exp}_x(tv)$ describes a great circle passing through x at $t = 0$ and such that $\frac{\mathrm{d}}{\mathrm{d}t}\operatorname{Exp}_x(tv)\big|_{t=0} = v$, hence Exp is indeed the exponential map for $\mathbb{S}^2$ (this can be more thoroughly checked by computing the second covariant derivative and verifying that it vanishes). While the closed-form expression for the map is not too complicated, we suggest using an even simpler map instead, called a retraction, see definition 2.7.2. The retraction we use on $\mathbb{S}^2$ is the following:

$$\mathsf{R}_x : T_x\mathbb{S}^2 \to \mathbb{S}^2 : v \mapsto \mathsf{R}_x(v) = \frac{x + v}{\|x + v\|} = \frac{x + v}{\sqrt{1 + \|v\|^2}}. \tag{4.1}$$

Using this retraction, it is impossible to reach points further away from x then a right angle (i.e., it is impossible to leave the hemisphere whose pole is x). As descent methods usually perform small steps anyway, this will not be an issue.

The Log map listed in table 4.1 is indeed the inverse of the exponential map, and as such is the logarithmic map on $\mathbb{S}^2$. This time, the expression is quite complicated (remember that we

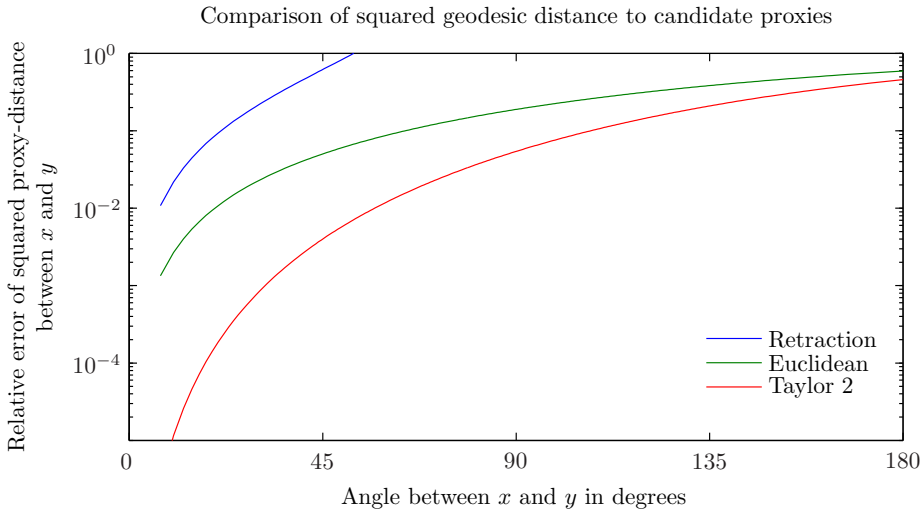

Figure 4.1: *Different proxies can be envisioned to replace the geodesic distance between x and y on $\mathbb{S}^2$ to lower the computational burden. All of them are not valuable though. This plot illustrates the relative error between the genuine squared geodesic distance and (1) the squared Euclidean distance (in the embedding space) -which coincides with a first order Taylor development of $x^T y \mapsto \arccos^2(x^T y)$, (2) a second order Taylor development and, least interestingly, (3) the squared norm that a vector v at x pointing toward y should have such that $\mathsf{R}_x(v) = y$. In this work, we always use the genuine geodesic distance.*

need to differentiate it). We propose a generalized logarithm, definition 2.7.5:

$$\mathsf{L}_x : \mathbb{S}^2 \to T_x\mathbb{S}^2 : y \mapsto \mathsf{L}_x(y) = P_x(y - x) = y - (x^T y)x. \tag{4.2}$$

This is indeed a generalized logarithm. Following the discussion at the end of section 2.7:

$$\mathsf{L}_x(y) = f_x(y) \cdot \mathrm{Log}_x(y), \quad f_x(y) = \frac{\|P_x(y-x)\|}{\mathrm{dist}(x,y)} = \frac{\sqrt{1-(x^T y)^2}}{\arccos(x^T y)},$$

where f_x is smoothly extended by defining $f_x(x) = \lim_{x^T y \to 1} f_x(y) = 1$. L_x maps points on the sphere to tangent vectors at x, pointing in the same direction as the genuine logarithmic map, but with a smaller norm. The error on the norm can be made arbitrarily small by constraining y to be close enough to x. We illustrate this fact by figure 4.2.

4.2 Curve space and objective function

Section 3.2 gave us a generic definition of the curve space and of the objective function to minimize over that space for a regression problem on a manifold $\mathcal{M}$. In this section, we specialize the definitions to $\mathcal{M} = \mathbb{S}^2$. We simply have that the curve space Γ consists of N_d copies of the sphere, i.e.,

$$\Gamma = \mathbb{S}^2 \times \ldots \times \mathbb{S}^2 = \left(\mathbb{S}^2\right)^{N_\mathrm{d}}.$$

Since Γ is just a product manifold based on $\mathbb{S}^2$, the tools we listed in table 4.1 for $\mathbb{S}^2$ can readily be extended to Γ component wise. For the sake of completeness, we show this explicitly in table 4.2. This second toolbox will mainly be useful for the optimization algorithms. In order to define the objective function, one only needs the first toolbox.

Building on sections 3.1 and 3.2, we propose the following energy function for the regression problem on $\mathbb{S}^2$:

$$E : \Gamma \to \mathbb{R} : \gamma \mapsto E(\gamma) = E_\mathrm{d}(\gamma) + \lambda E_\mathrm{v}(\gamma) + \mu E_\mathrm{a}(\gamma) \tag{4.3}$$

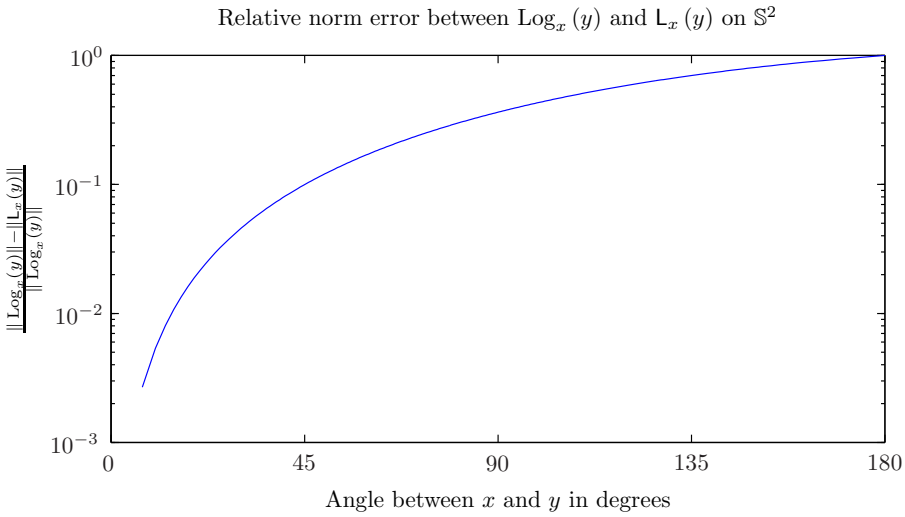

Figure 4.2: *Both* $\mathrm{Log}_x(y)$ *and* $\mathsf{L}_x(y)$ *point in the same direction, but the latter has a smaller norm. This plot illustrates the relative error introduced by considering the generalized logarithm* (4.2) *instead of the true logarithm as a function of the angle separating* x *and* y*. In particular, for angles under* 12.6 *degrees (or* 0.22 *radians), the error is under 1%, then quickly decays. Since we deal with finer samplings anyway, the additional error is a very cheap price to pay in comparison to the decrease in complexity.*

Set:	$\Gamma = \mathbb{S}^2 \times \ldots \times \mathbb{S}^2$ (N_d copies)
Tangent spaces:	$T_\gamma\Gamma = T_{\gamma_1}\mathbb{S}^2 \times \ldots \times T_{\gamma_{N_\mathrm{d}}}\mathbb{S}^2$
Projector:	$P_\gamma : \mathbb{R}^{3\times N_\mathrm{d}} \to T_\gamma\Gamma : v \mapsto P_\gamma(v) = w,\ w_i = P_{\gamma_i}(v_i)$
Inner product:	$\langle v, w\rangle_\gamma = \sum_{i=1}^{N_\mathrm{d}} \langle v_i, w_i\rangle_{\gamma_i}$
Vector norm:	$\|v\|_\gamma = \sqrt{\langle v, v\rangle_\gamma}$
Exponential:	$\mathrm{Exp}_\gamma(v) = \tilde{\gamma},\ \tilde{\gamma}_i = \mathrm{Exp}_{\gamma_i}(v_i)$

Table 4.2: *Geometric toolbox for the Riemannian manifold* $\Gamma = \mathbb{S}^2 \times \ldots \times \mathbb{S}^2$

$$
\begin{aligned}
E_\mathrm{d}(\gamma) &= \frac{1}{2}\sum_{i=1}^{N} w_i \arccos^2(p_i^T\gamma_{s_i}) \\
E_\mathrm{v}(\gamma) &= \frac{1}{2}\sum_{i=1}^{N_\mathrm{d}-1} \frac{1}{\Delta\tau_i}\arccos^2(\gamma_i^T\gamma_{i+1}) \\
E_\mathrm{a}(\gamma) &= \frac{1}{2}\left[\frac{\Delta\tau_1}{2}\|a_1\|^2 + \sum_{i=2}^{N_\mathrm{d}-1}\frac{\Delta\tau_{i-1}+\Delta\tau_i}{2}\|a_i\|^2 + \frac{\Delta\tau_{N_\mathrm{d}-1}}{2}\|a_{N_\mathrm{d}}\|^2\right]
\end{aligned}
\tag{4.4}
$$

E_v follows from choosing the forward order 1 formula for velocity as well as the rectangle method for the weights α_i. In particular, $\alpha_{N_\mathrm{d}} = 0$, hence we do not need backward differences. E_v has a very simple form because the norm of $\mathrm{Log}_x(y)$ is simply the geodesic distance between x and y. We choose the trapezium method for the weights β_i. The main reason for doing so is purely aesthetic, as the weights appearing in the finite differences formulas cancel out the integration weights. The $\Delta\tau_i$'s are defined as: $\Delta\tau_i = \tau_{i+1} - \tau_i$.

Replacing Log by L in the geometric finite differences for acceleration, equations (3.4) and (3.5), we *define* the a_i's as follows:

$$
a_1 = \frac{-2}{\Delta\tau_1\Delta\tau_2}\mathsf{L}_{\gamma_1}(\gamma_2) + \frac{2}{\Delta\tau_2(\Delta\tau_1+\Delta\tau_2)}\mathsf{L}_{\gamma_1}(\gamma_3)
$$

$$a_i = \frac{2}{\Delta\tau_{i-1} + \Delta\tau_i}\left[\frac{1}{\Delta\tau_i}\mathsf{L}_{\gamma_i}(\gamma_{i+1}) + \frac{1}{\Delta\tau_{i-1}}\mathsf{L}_{\gamma_i}(\gamma_{i-1})\right], \quad \forall i \in 2 \dots N_\mathrm{d} - 1$$
$$a_{N_\mathrm{d}} = \frac{-2}{\Delta\tau_{N_\mathrm{d}-1}\Delta\tau_{N_\mathrm{d}-2}}\mathsf{L}_{\gamma_{N_\mathrm{d}}}(\gamma_{N_\mathrm{d}-1}) + \frac{2}{\Delta\tau_{N_\mathrm{d}-2}(\Delta\tau_{N_\mathrm{d}-1} + \Delta\tau_{N_\mathrm{d}-2})}\mathsf{L}_{\gamma_{N_\mathrm{d}}}(\gamma_{N_\mathrm{d}-2})$$

These will be easier to differentiate, hence making the next section less cumbersome. Remarkably, the a_i's defined above via simplified geometric differences are the same as the ones we would obtain by computing the finite differences in the ambient space $\mathbb{R}^3$ then by projecting the results back on the tangent planes using the P_{γ_i}'s.

4.3 Gradient of the objective

Since $\mathbb{S}^2$ is a Riemannian submanifold of $\mathbb{R}^3$, Γ is a Riemannian submanifold of $\mathbb{R}^{3\times N_\mathrm{d}}$ and, as such, theorem 2.4.5 applies. We can therefore write:

$$\operatorname{grad} E : \Gamma \to T\Gamma : \gamma \mapsto \operatorname{grad} E(\gamma) = P_\gamma \nabla\overline{E}(\gamma).$$

In the above equation, $\overline{E}$ is a scalar field on an open set of the ambient space $\mathbb{R}^{3\times N_\mathrm{d}}$ containing Γ and such that $\overline{E}\big|_\Gamma = E$; $T\Gamma$ is the tangent bundle of Γ, see definition 2.2.5.

This makes it a lot easier to derive the gradient of E, equation (4.3). We just need to provide the gradients on Γ of the functions defined in equations (4.4), interpreted as smooth scalar fields defined on an open set of $\mathbb{R}^{3\times N_\mathrm{d}}$ containing Γ.

The following atoms are enough to compute $\nabla\overline{E}$, hence $\operatorname{grad} E$ (with $x, q \in \mathbb{S}^2$, $g : \mathbb{R}^3 \to \mathbb{R}^3$):

$$\nabla\left(x \mapsto \arccos^2(q^T x)\right)(x) = \frac{-2\arccos(q^T x)}{\sqrt{1-(q^T x)^2}}q$$
$$\nabla\left(x \mapsto \|g(x)\|^2\right)(x) = 2(\mathrm{J}g(x))^T g(x)$$
$$\mathrm{J}\left(x \mapsto \mathsf{L}_q(x)\right)(x) = P_q$$
$$\mathrm{J}\left(x \mapsto \mathsf{L}_x(q)\right)(x) = -(x^T q)I - xq^T$$

J denotes the Jacobian, ∇ denotes the gradient of real functions. A completely explicit derivation of $\operatorname{grad} E$ would be overly cumbersome. The material above is sufficient to obtain the needed expressions. The derivation does not present any challenges, since it reduces to a standard derivation of a real function followed by a projection (theorem 2.4.5).

4.4 Results and comments

We implemented algorithm 4 for $\mathcal{M} = \mathbb{S}^2 \times \ldots \times \mathbb{S}^2 = \Gamma$ to optimize objective (4.3). We skip over the technical details of the implementation. The material exposed in this document still being a proof of concept, we do not deem it important to delve into such details yet. The accompanying code was written with care. The interested reader can inspect it for further information.

We divide this section in two parts. The first one qualitatively exposes the kind of solutions we expect based on the tuning parameters λ and μ. Practical results produced by our algorithms are shown to illustrate the adequacy between predicted and obtained curves. The second part deals with the behavior our algorithms exhibit when converging toward some solutions. In particular, we show that CG methods are usually more efficient than the steepest descent method and behave better (essentially reducing oscillations around the solution). Also, what we later on refer to as our refinement technique, algorithm 6, is shown to speed up convergence and increase reliability.

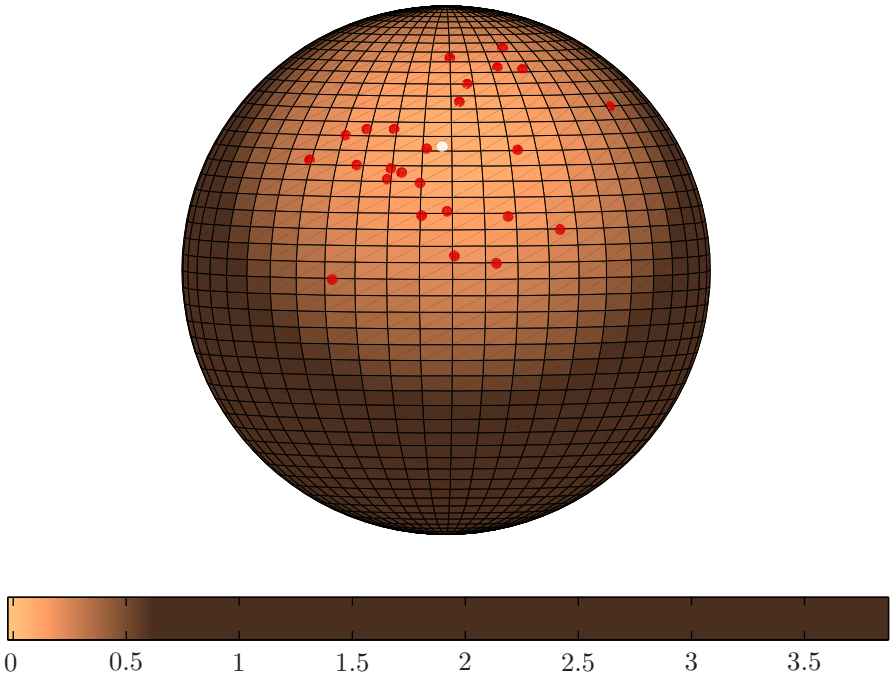

Figure 4.3: *The red points are the data on the sphere, with identical weights. The color on the sphere gives the value of the scalar field E, i.e., the sum of squared geodesic distances to the data. The minimum of E corresponds to the mean of the data, the single white dot. A very good initial guess is to project the arithmetic mean in $\mathbb{R}^n$ back on the sphere by normalizing it. Convergence to six digits of accuracy is achieved in typically under four iterations.*

4.4.1 Types of solutions

Zeroth order

Setting $\lambda = \mu = 0$ yields an objective function E that completely disregards smoothness and just tries to fit the discretization points γ_i to the associated data. Discretization points with no data point attached become irrelevant. This makes little sense unless $N_{\mathrm{d}} = 1$, in which case the only point we search for corresponds to the weighted mean of the data on the manifold. In this setting, we ignore the time labels t_i. The objective we minimize is:

$$E : \mathbb{S}^2 \to \mathbb{R} : \gamma \mapsto E(\gamma) = \frac{1}{2} \sum_{i=1}^{N} w_i \arccos^2(p_i^T \gamma)$$

This is to be compared to the usual definition of the arithmetic mean in $\mathbb{R}^n$:

$$\overline{p} = \arg\min_{\gamma \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{N} w_i \|p_i - \gamma\|^2 = \frac{\sum_{i=1}^{N} w_i p_i}{\sum_{i=1}^{N} w_i}$$

Of course, the explicit formula for $\overline{p}$ as a linear combination of the p_i's does not make sense on $\mathbb{S}^2$ and we therefore use our numerical algorithms instead[1]. Figure 4.3 shows an example. An excellent initial guess for the CG algorithm is the Euclidean mean projected back on the sphere, i.e., $\gamma^0 = \frac{\overline{p}}{\|\overline{p}\|}$. Typically, convergence is achieved to 6 significant digits in under 4 iterations.

First order

In $\mathbb{R}^n$, setting $\lambda > 0$ and $\mu = 0$ disregards second order derivatives, yielding piecewise affine regression curves. Solving the corresponding problem on $\mathbb{S}^2$, we expect solutions to be piecewise

[1] Interestingly, since $\mathbb{S}^2$ is compact and E is continuous, we can just as easily define and compute the point that is as far away as possible of the data as the maximizer of E, which was not possible in $\mathbb{R}^n$.

geodesic. This is indeed what we observe. This means that a solution is completely specified by its breaking points, i.e., one γ_i associated to each different data time label t_i. In the continuous case, Machado et al. showed the same behavior in [MSH06]. It would be interesting to check whether the breaking points are the same following both the continuous and our discrete approach.

As λ goes to 0, we expect the solutions to converge toward piecewise geodesic interpolation, since reducing the distance between the curve and the data will become increasingly important. As λ goes to infinity, we expect the curve to collapse toward the mean of the data, since the length of the curve will be highly penalized. We show a few numerical results corroborating our intuition. Figure 4.4 shows different solutions for three different λ's.

Second order

In $\mathbb{R}^n$, setting $\lambda = 0$ and $\mu > 0$ disregards first order derivatives and penalizes second order derivatives, yielding solutions in a well-known class: cubic splines. Cubic splines are piecewise cubic curves constrained to be of class $\mathcal{C}^2$ (in the continuous case). These are characterized by a continuous, piecewise affine acceleration profile $\ddot{\gamma}(t)$ along each dimension, with breaking points at the data times. This is indeed what we obtain when solving the problem in $\mathbb{R}^n$ as described in chapter 1 (acceleration is computed with standard finite differences).

We may expect to observe this same behaviour on $\mathbb{S}^2$. The results depicted on figure 4.5 (where, for visualization, a minus sign has been added to the acceleration for times $t > \frac{1}{2}$ since the curvature changed at that point) indeed exhibits a compatible acceleration profile. But figure 4.6 shows a different situation, where the acceleration profile is only continuous and piecewise differentiable: the profile segments are curved. This surprising result was already predicted and quantified in a continuous setting by Machado and Silva Leite in [MS06, Thm 4.4]. The result essentialy gives a differential equation for the connecting segments and continuity contraints, notably showing that the fourth covariant derivative of γ does not, in general, vanish. Figure 4.6 shows that a similar effect exists in the discrete case. It would be interesting to investigate whether the effects are quantitatively the same or not. We expect this to be the case.

Tuning μ, again, gives the user access to a whole family of curves. It is important to realize that, regardless of μ, the length of the curve is not penalized. As μ goes to 0, fitting the data becomes increasingly important, yielding near interpolation. As μ goes to infinity, we expect the curve to converge toward a geodesic regression of the data. This is indeed the behavior we can see on figure 4.5 and figure 4.6.

Geodesic regression is such an important special case that we give another illustration of it on figure 4.7. Of course, our general approach is not well suited because of the need to make μ increase to infinity. Plus, we deal with too many degrees of freedom considering the simplicity of the targeted object. Machado and Silva Leite, in [MS06], exploited the fact that a geodesic is completely defined by a point p on $\mathbb{S}^2$ and a vector $v \in T_p\mathbb{S}^2$ to exhibit a set of necessary conditions for a geodesic to be optimal for the regression problem. Sadly, these equations are complicated and the authors do not try to solve them explicitly. In future work, we would like to try our descent algorithms on the set of pairs (p, v), which can be identified with the tangent bundle, hence is a manifold in its own respect. The tangent bundle of a Riemannian manifold can be endowed with a metric in various ways. This is a question worth investigating.

Mixed

When λ and μ are both positive, there is no clear classification of the solutions, even in $\mathbb{R}^n$. Reasoning on the first and second order families described above can give some intuition as to how one should tune the parameters to obtain the desired curve. We show an example on figure 4.8.

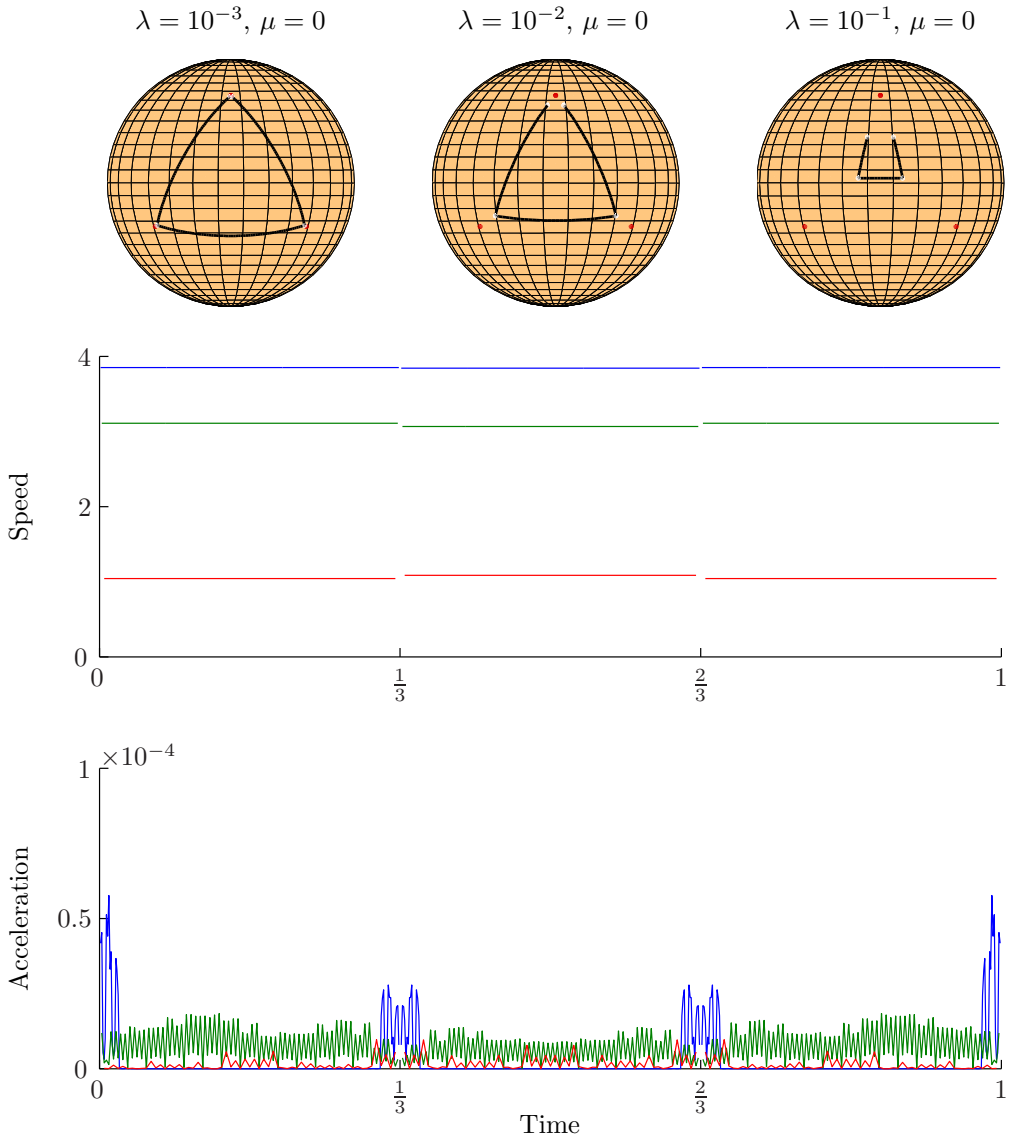

Figure 4.4: *In this figure, $p_1 = p_4$ is the upper point. By tuning λ with μ set to zero, one can generate a family of piecewise geodesic curves ranging from near interpolation to a collapsed curve onto the mean point of the data. We only need to compute the end points and the breaking points, i.e., the $\gamma(t_i)$'s. Numerical experiments indeed confirm that optimizing a finer curve with γ_i's associated to intermediate times τ_i gives the same breaking points (up to numerical precision), while the additional points lie on the geodesics joining them, as they should. The optimal objective values will also be the same. These statements hold for the definition of E_v given in equation* (4.4)*, since integration of a constant function with the rectangle method is exact. This observation is of great practical importance. Indeed, we know exactly how to choose the discretization of γ to optimize at minimal computational cost. We also know that setting the γ_i's on the data points is a good initial guess for the descent method. On this example, convergence to a curve such that the norm of* $\operatorname{grad} E$ *is less than* 10^{-8} *is achieved in, respectively, 6, 11 and 15 CG iterations with the Fletcher-Reeves coefficients and the backtracking-interpolation step choosing algorithm. The speed and acceleration profiles have been computed using true geometric finite differences, equations* (3.4) *and* (3.5)*. Data at the breaking points has been removed. For the spheres from left to right, the speed and acceleration profile colors are respectively blue, green and red.*

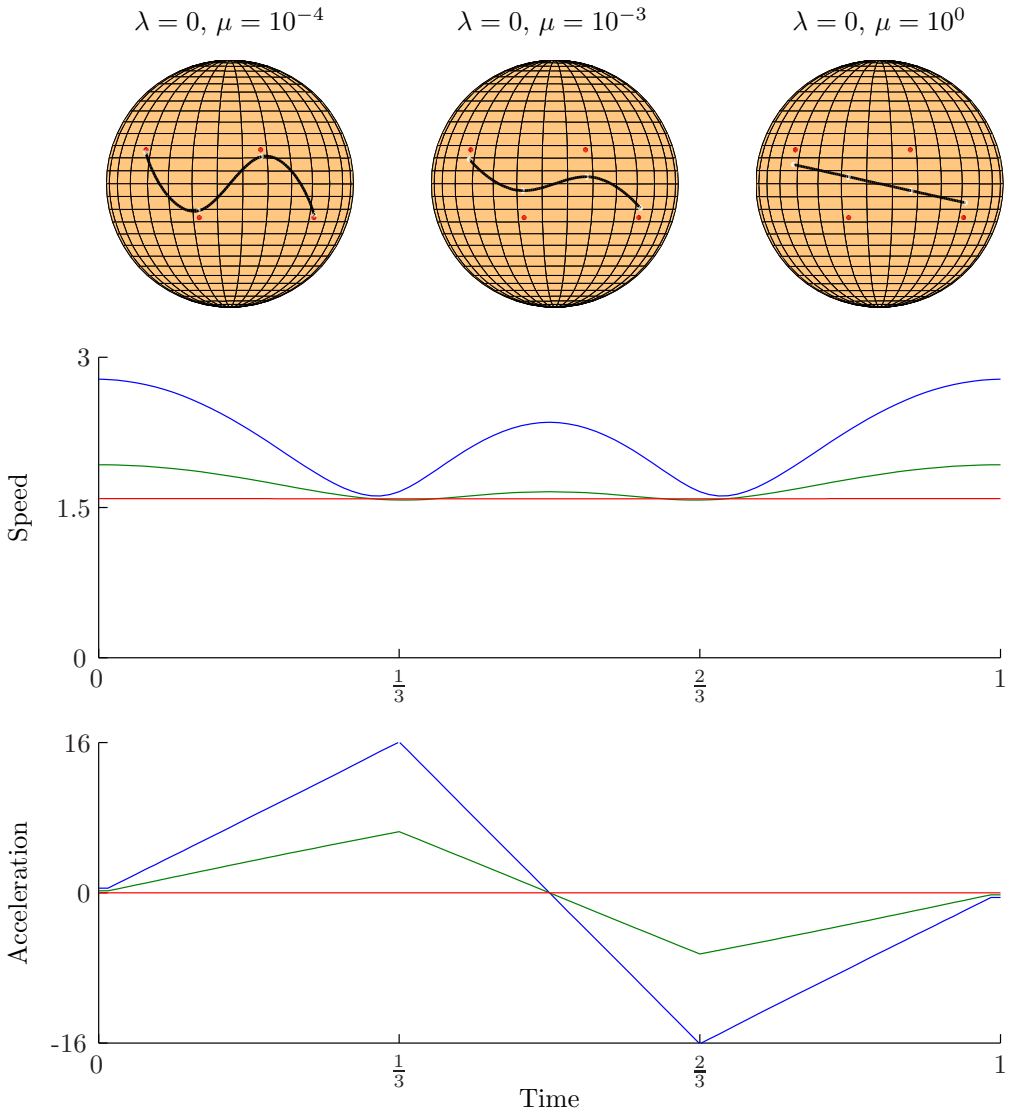

Figure 4.5: $\lambda = 0, \mu > 0$. *The speed and acceleration profiles are computed with geometric finite differences using the exact logarithm. For the spheres from left to right, the profile colors are respectively blue, green and red. The acceleration profiles, as displayed with corrected sign on the second half, are piecewise affine (up to numerical precision). This is consistent with cubic splines in* $\mathbb{R}^n$*, but is atypical on manifolds. The near-geodesic solution is the hardest to compute. See figure 4.7 for more details. The flat segments at the end points* ($t = 0$ *and* $t = 1$) *are caused by our implementation of* E *which artificially sets the integration weights* β_1 *and* β_{N_d} *to zero to spare us the burden of computing the gradient of unilateral differences. This only slightly affects the solution.*

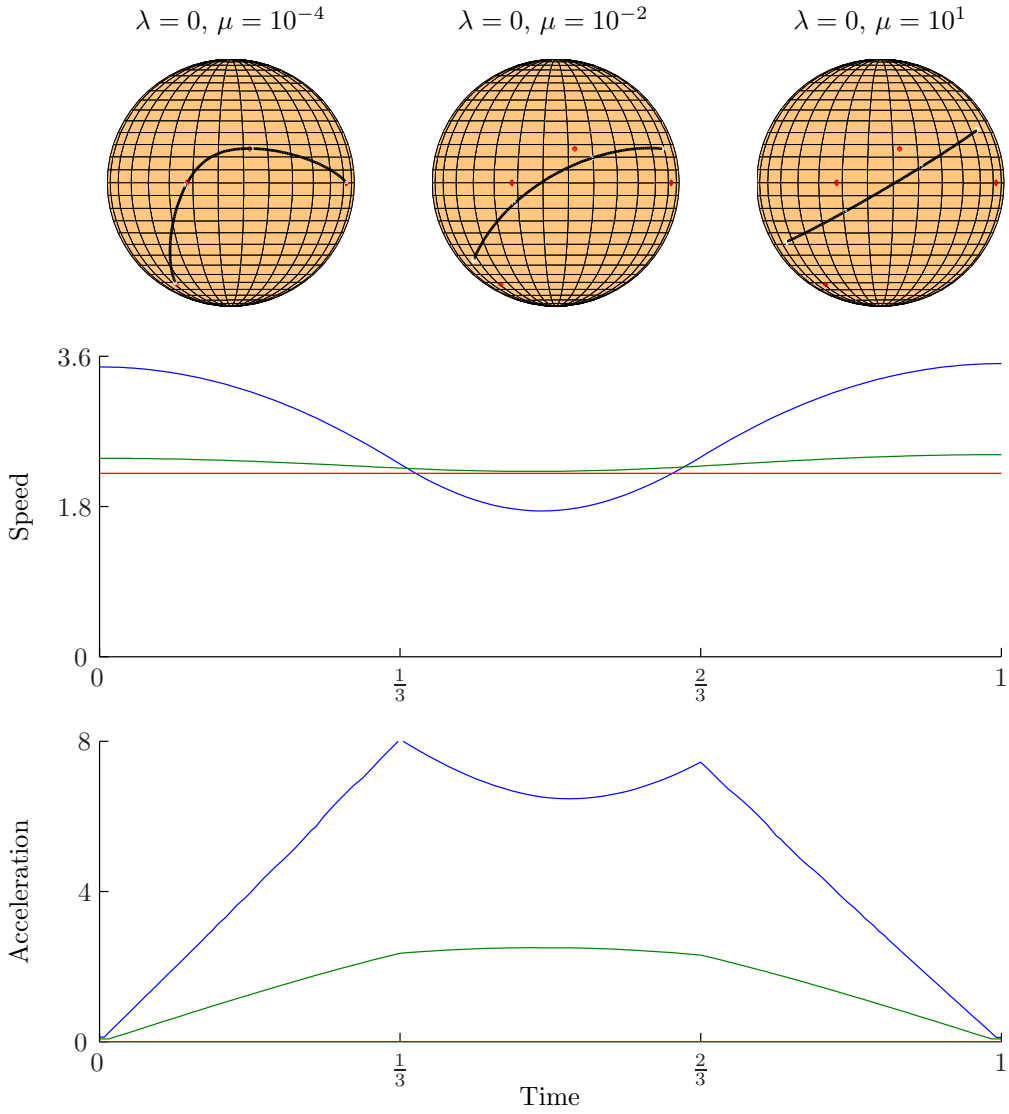

Figure 4.6: *This figure is to be compared with figure 4.5. Notice how the acceleration profiles are not piecewise affine anymore. Previous work by Machado and Silva Leite with continuous curves, see [MS06], predicted this departure from what can be observed with cubic splines in $\mathbb{R}^n$. This numerical experiment shows the existence of a similar effect for discretized curves.*

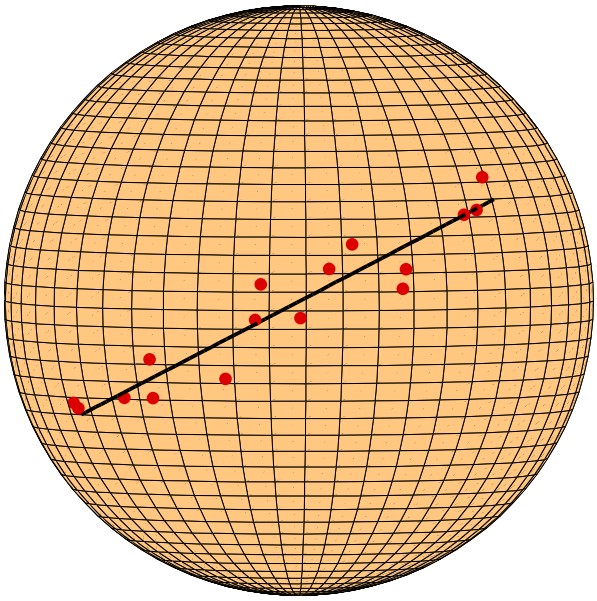

Figure 4.7: *When the acceleration on $\gamma \in \Gamma$ is evaluated using geometric finite differences with the exact logarithm on $\mathbb{S}^2$, it is zero if and only if there is a geodesic $\gamma(t)$ such that $\gamma(\tau_i) = \gamma_i$ for each i, see section 3.1. Hence, when setting $\lambda = 0$ and some large value for μ, it is sufficient to find (optimize) the position of $\gamma(t_i)$ for each distinct data time t_i. This observation yields an effective, reliable manner of computing the geodesic regression. Unfortunately, when we use the simplified logarithm and the $\Delta\tau_i$'s are not all equal, the claim is not true anymore. The above procedure still gives a nice initial guess for the optimization algorithm, but we further need to refine the curve (increasing N_d hence also the computational cost) to reduce the impact of the generalized logarithm on the final accuracy. With large N_d, the optimization algorithm tends to "be satisfied" with any geodesic not too far from the data. To avoid this, an efficient workaround is to optimize with $N_\mathrm{d} = N$ first (place the initial guess on the data), then successively re-optimize while increasing N_d by curve refinement, algorithm 6. The curve shown in this figure has maximum acceleration of 1.5×10^{-4} for a total length of 1.79 over 1 unit of time. The parameter values are: $\lambda = 0$, $\mu = 10$.*

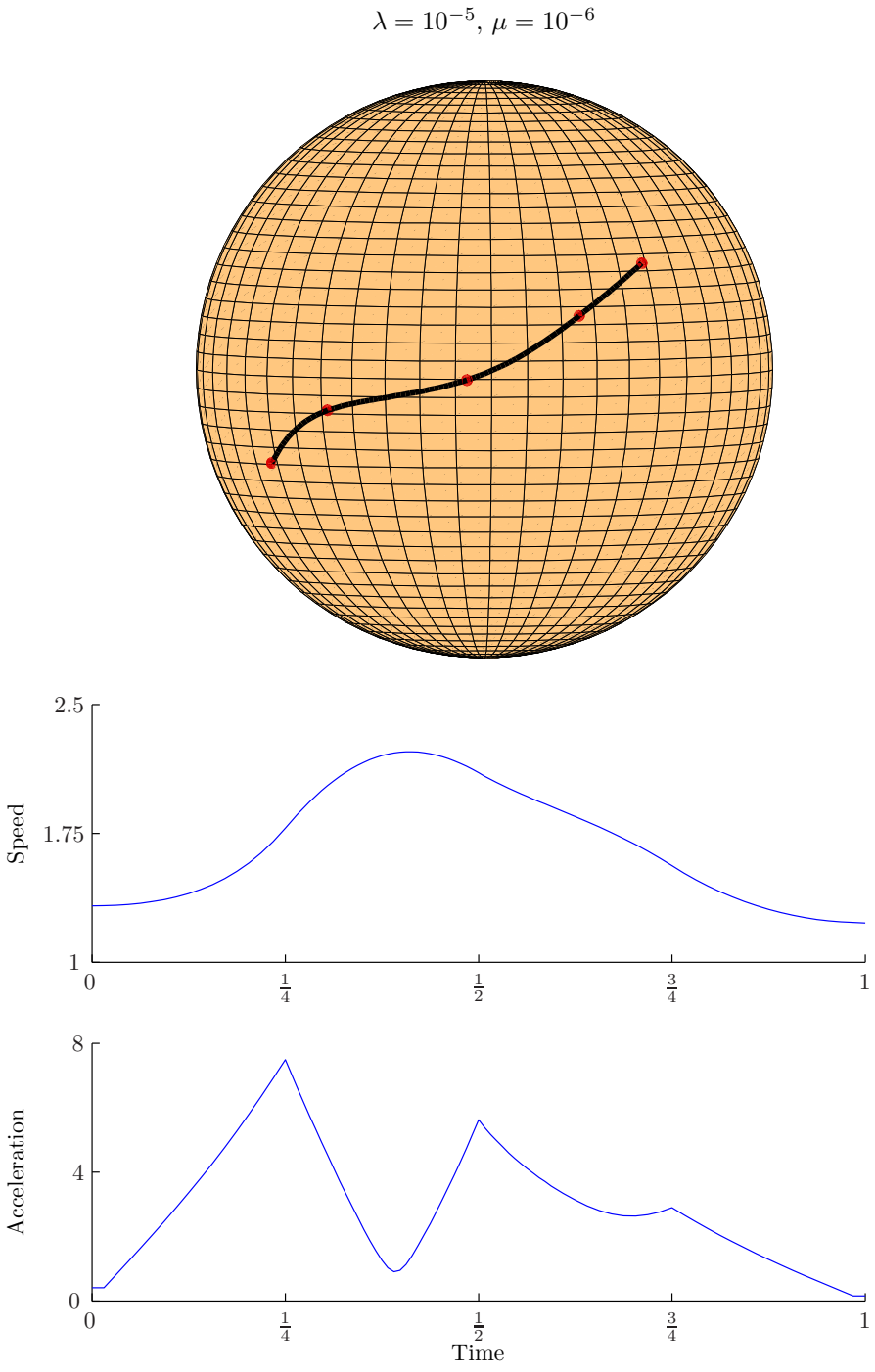

Figure 4.8: *When $\lambda > 0$ and $\mu > 0$, the solution lives in a large family of curves. The acceleration profile is still continuous and looks piecewise differentiable, but we do not have an easy characterization of the family anymore (not even in the Euclidean case). The flat pieces at the end points on the acceleration plot were explained in the comments of figure 4.5. Notice how picking small positive values for both tuning parameters yields a near-interpolatory yet smooth-looking curve.*

4.4.2 Numerical performances

Aside from the data volume, i.e., N, the computational cost depends on a number of parameters. Namely:

- The core descent algorithm used,
- The step choosing algorithm,
- The quality of the initial guess,
- The number of discretization points, and
- The parameters λ and μ.

The relationship between numerical performance and factors such as N_d, λ and μ is complex to describe. For the other factors, we give a few guidelines hereafter.

The core descent algorithms we introduced in chapter 3 can be anything from a steepest descent method to a Polak-Ribière or Fletcher-Reeves flavored CG. The difference between Fletcher-Reeves and Polak-Ribière is not, in general, significant. They usually both perform better than a steepest descent. Detailed algorithms to apply second order descent methods on manifolds, including trust region methods, are described in [AMS08]. We would like to try these in future work.

We select algorithm 5 as our step length choosing method. It is particularly well suited to our problem since, for small steps, the line-search function $\phi(\alpha) = E(\mathsf{R}_\gamma(\alpha p))$ can be (amazingly) well approximated by a quadratic. This comes from the fact that, in $\mathbb{R}^n \times \ldots \times \mathbb{R}^n$, our objective function E is quadratic and Γ locally resembles a Euclidean space. Some alternative step choosing algorithms require the computation of $\phi'(\alpha)$, the derivative of the line search function. For the sake of completeness, we provide an explicit formula for it in appendix A.

Picking a smart initial guess is important to ensure convergence. Fortunately, numerical experiments tend to show that the attraction basin of the global minimum is surprisingly large (but not equal to Γ). Even picking γ^0 at random often works. Here are a few choices we recommend depending on the scenario:

- For the data mean, choose $\gamma^0 = \frac{\overline{p}}{\|\overline{p}\|}$ where $\overline{p}$ is the arithmetic mean of the data in $\mathbb{R}^3$;
- For piecewise geodesic regression, choose $\gamma_i^0 = p_i$;
- For geodesic regression, choose $\gamma_i^0 = p_i$, optimize, then successively refine the curve and re-optimize;
- For mixed order regression, solve the problem in $\mathbb{R}^3$ (chapter 1) then project the solution back on the sphere, or do the same as for geodesic regression.

As a rule of thumb, it is a good idea to pre-solve the problem with low N_d. Typically, pre-solving with one discretization point γ_i for each distinct data time label t_i yields good performances. Placing the initial curve γ^0 on the data for this pre-solving step has never failed in our numerical experiments, regardless of the scenario. The refinement step we refer to is highlighted in algorithm 6. It requires the possibility to compute the mean of two points on the manifold[2]. The main advantage of this algorithm is that it lets us set a tolerance on the spacing between two discretization points without oversampling. In other words, it is fine to work with curves that alternate slow and fast segments. The resulting time sampling $\tau_1, \ldots, \tau_{N_\mathrm{d}}$ will, in general, be non-homogeneous. The added advantage is that most of the optimization is done on fewer points than the final discretization quality.

[2]If that is overly complicated, one can always use γ_i or γ_{i+1} as an approximation for the mean of these same points. It is then up to the optimization process to move the new points to the right places.

Algorithm 6: Iterative curve refinement strategy

input : An initial curve γ; tol, a refinement tolerance.
output: An optimized curve γ such that two successive points are separated by a distance less than tol.

```
continue := true
while continue do
    γ := optimize(γ)
    continue := false
    foreach i such that dist(γ_i, γ_{i+1}) > tol do
        Insert mean(γ_i, γ_{i+1}) between them at time (τ_i + τ_{i+1})/2
        continue := true
    end
end
```

The CG algorithm performed really well on the piecewise geodesic regression (order 1) problems. Figure 4.9 shows the evolution of step length, gradient norm and objective value over the 15 descent iterations. Figure 4.10 shows the same thing with a steepest descent algorithm. CG clearly beats SD. Both exhibit a nice behavior.

Second order problems need more iterations. Furthermore, when μ has a high value (which is desirable when near geodesic regression is needed), the raw algorithm tends to get stuck with a close-to geodesic curve not too far away from the data. This stems from the fact that μE_{a} overshadows E_{d}. To circumvent this, we use algorithm 6. The convergence behavior for the second case exposed in figure 4.5 is shown in figure 4.11. Only the pre-optimization step is shown, i.e., the placement of the γ_i's corresponding to the p_i's. Subsequent steps (refinement + optimization) can be rapidly carried out (around 30 iterations). Fletcher-Reeves and Polak-Ribière perform about the same. The steepest descent method does as good a job as CG on the two first instances, and takes as much as five times more iterations on the close-to geodesic example for comparable precision.

Computing geodesic regressions is an interesting problem. As we showed before, it turns out to be the most challenging one for our algorithms. Figure 4.12 shows step length and gradient norm evolution over a huge number of iterations (1250) of the CG algorithm for the geodesic regression shown in figure 4.7. We have other algorithms in mind for geodesic regression. We will explore them in future work.

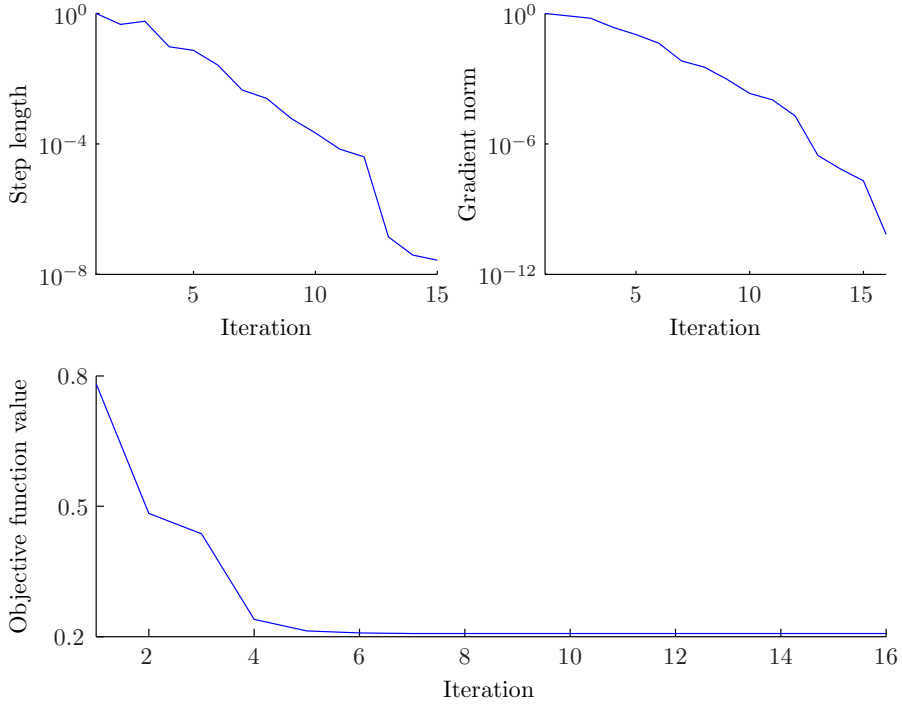

Figure 4.9: *Convergence of the Fletcher-Reeves CG algorithm on the most difficult case of figure 4.4, i.e., $\lambda = 10^{-1}$. The behavior of the algorithm is excellent. It is slightly better than with Polak-Ribière. The evolution of the norm of the gradient is typical for superlinear convergence. Notice how the objective value quickly reaches a plateau. To assess convergence, it is best to look at the gradient norm evolution.*

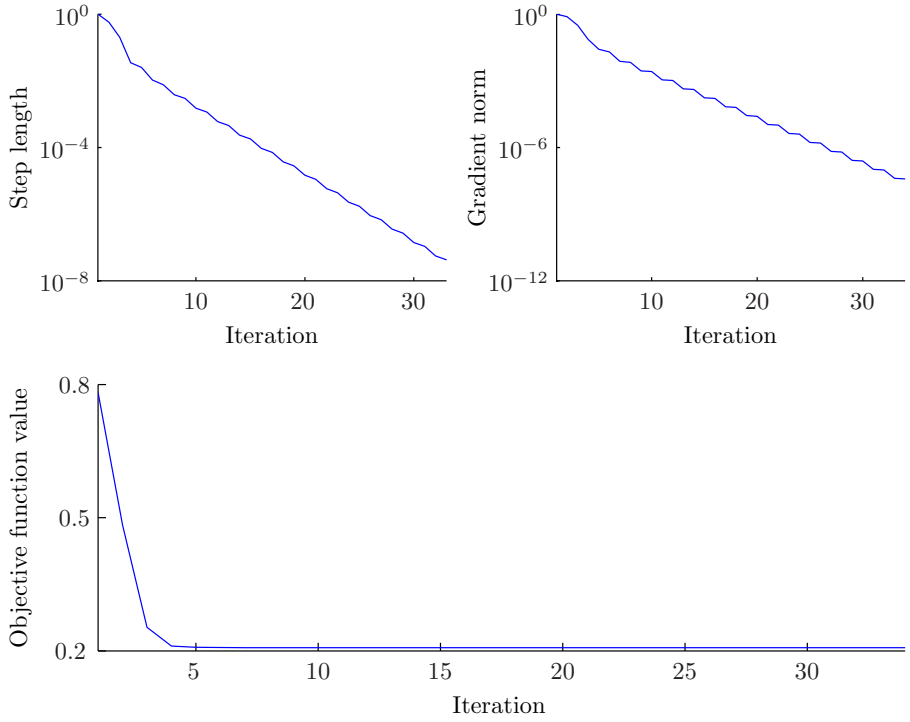

Figure 4.10: *Convergence of the steepest descent algorithm on the same problem as figure 4.9. The behavior of the algorithm is good, but it needs more than twice as many iterations as its CG counterpart. The evolution of the norm of the gradient is typical for linear convergence.*

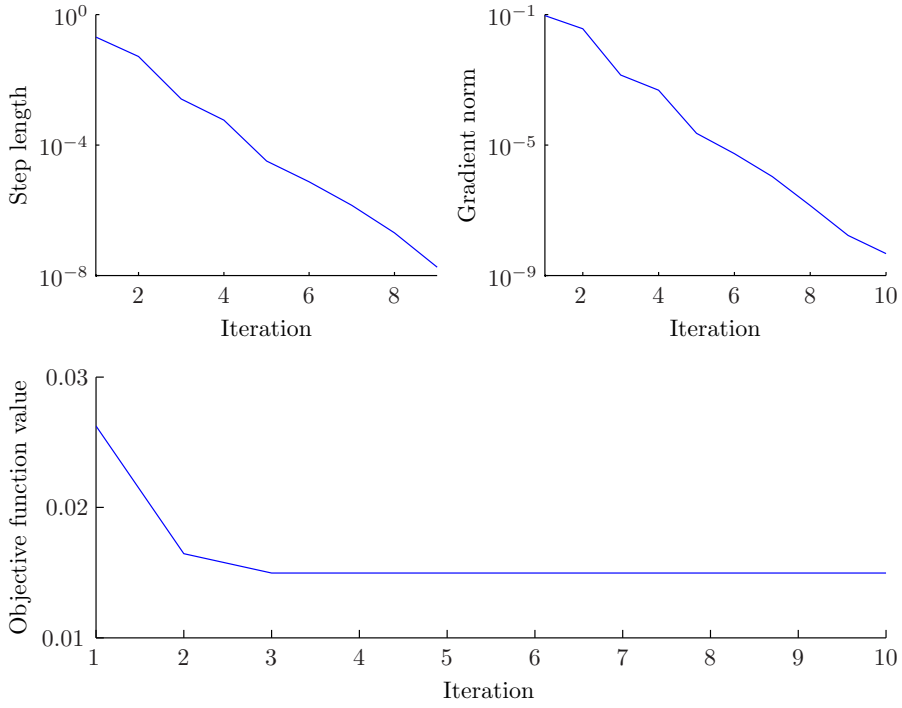

Figure 4.11: *Optimization of the γ_i's corresponding to the p_i's for the second case displayed in figure 4.5. Convergence is achieved in about ten iterations. All our algorithms perform about the same except for the close-to geodesic regression example, for which the CG algorithm performs much better. In subsequent refinement steps, the points computed in this step will be slightly displaced.*

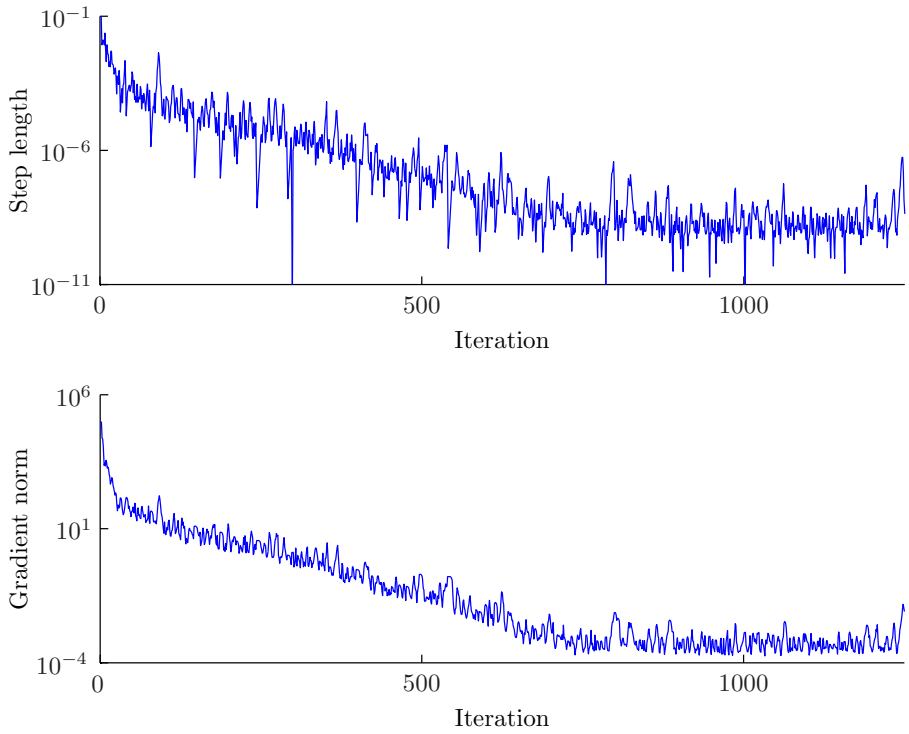

Figure 4.12: *Optimization with CG for the problem shown in figure 4.7, with initial guess for γ equal to the data. The parameter μ is set to 10 and the optimization is carried out by successive refinements. After about 750 iterations, the algorithm reached a threshold. The gradient norm was reduced by 8 orders of magnitude, and numerical precision becomes an issue. We refine the curve by adding points on the geodesics joining the γ_i's and re-optimizing in a few steps.*

Chapter 5

Discrete curve fitting on positive-definite matrices

In the previous chapter, we built and analyzed a method to fit curves to data on the sphere. One of the main advantages of that manifold is that it is very easy to visualize the data and how the algorithms behave. In that setting, intuition can get you a long way and one might argue that the differential geometry background is expendable. $\mathbb{S}^2$ was an ideal toy example to test our ideas, but we need to work on more complicated manifolds to justify the complexity of our method.

In this chapter, we propose to study curve fitting on a manifold that is both difficult to picture and of practical interest: the set of positive-definite matrices $\mathbb{P}_+^n$, endowed with a Riemannian metric. As is, $\mathbb{P}_+^n$ is an open convex cone. With the usual metric, $\mathbb{P}_+^n$ is thus not complete[1]. Completeness is a desirable property because our optimization algorithms produce sequences of points on $\mathbb{P}_+^n$ and we certainly want these sequences to converge in $\mathbb{P}_+^n$. A possible workaround is to use a metric that will deform the space so as to stretch the cone limits to infinity. One of the metrics that does that, later referred to as the affine-invariant metric, is used in this chapter and the methodology developed in this document is applied to the resulting manifold. As we will see, computations are more involved, mainly because of the need for derivatives of matrix functions.

We then exhibit other ways of solving similar problems. We first observe that, $\mathbb{P}_+^n$ being a convex set, we can interpolate between points by using piecewise linear interpolation. This does not give us the freedom to tune between fitting and smoothness, but it is nevertheless a very simple method that we need to compare our algorithms against. Yet another alternative is to exploit the convex nature of our problem. Modern solvers can readily deal with the usual matrix norms. Next, we investigate an interesting metric introduced by Arsigny et al. in [AFPA08], called the Log-Euclidean metric. It endows $\mathbb{P}_+^n$ with a vector space structure, effectively enabling us to use the simple mathematics developed in the first chapter to solve our problem rapidly.

We conclude this chapter by testing all described methods and comparing them against each other.

[1] Quoting [Wei10], a complete metric space is a metric space in which every Cauchy sequence is convergent. In this definition, the *metric* refers to the Riemannian distance induced by the Riemannian metric.

5.1 $\mathbb{P}_+^n$ geometric toolbox

We note $\mathbb{H}^n$ the set of symmetric matrices of size n and $\mathbb{P}_+^n \subset \mathbb{H}^n$ the set of positive-definite matrices of size n. We use the notation $A \in \mathbb{P}_+^n \Leftrightarrow A \succ 0$. The embedding space $\mathbb{H}^n$ is endowed with the usual metric

$$\langle \cdot, \cdot \rangle : \mathbb{H}^n \times \mathbb{H}^n \to \mathbb{R} : (H_1, H_2) \mapsto \langle H_1, H_2 \rangle = \operatorname{trace}(H_1 H_2) = \operatorname{trace}(H_2 H_1).$$

The norm associated to this metric is the Frobenius norm

$$\|H\| = \sqrt{\langle H, H \rangle} = \sqrt{\operatorname{trace}(H^2)} = \sqrt{\sum_{i,j} H_{ij}^2}.$$

The set of positive-definite matrices is an open convex subset of $\mathbb{H}^n$, hence it is not complete. Giving $\mathbb{P}_+^n$ a Riemannian manifold structure by restricting the above metric to $\mathbb{P}_+^n$ thus would yield a non-complete manifold. We can endow $\mathbb{P}_+^n$ with a different Riemannian metric in order to make the resulting manifold complete. One such metric is given in table 5.1, see [AFPA08, Bha07]. Following Arsigny et al. in [AFPA08], we call it the *affine-invariant metric*. Since the affine-invariant metric is *not* the restriction of the metric on $\mathbb{H}^n$ to $\mathbb{P}_+^n$, with this metric, $\mathbb{P}_+^n$ is not a Riemannian submanifold of $\mathbb{H}^n$. Regardless of the metric, $\mathbb{P}_+^n$ being an open subset of $\mathbb{H}^n$, the tangent space $T_A\mathbb{P}_+^n$ at any positive matrix A corresponds to the set $\mathbb{H}^n$. We illustrate this on figure 5.1. From now on, $\mathbb{P}_+^n$ refers to the set of positive-definite matrices endowed with the geometric toolbox described in table 5.1.

Set:	$\mathbb{P}_+^n = \{A \in \mathbb{R}^{n\times n} \ : \ A = A^T \text{ and } x^T A x > 0 \ \forall x \in \mathbb{R}^n, x \neq 0\}$
Tangent spaces:	$T_A\mathbb{P}_+^n \equiv \mathbb{H}^n = \{H \in \mathbb{R}^{n\times n} \ : \ H = H^T\}$
Inner product:	$\langle H_1, H_2 \rangle_A = \left\langle A^{-1/2} H_1 A^{-1/2}, A^{-1/2} H_2 A^{-1/2} \right\rangle$
Vector norm:	$\|H\|_A = \sqrt{\langle H, H \rangle_A}$
Distance:	$\operatorname{dist}(A, B) = \left\| \log\left(A^{-1/2} B A^{-1/2}\right) \right\|$
Exponential:	$\operatorname{Exp}_A(H) = A^{1/2} \exp\left(A^{-1/2} H A^{-1/2}\right) A^{1/2}$
Logarithm:	$\operatorname{Log}_A(B) = A^{1/2} \log\left(A^{-1/2} B A^{-1/2}\right) A^{1/2}$
Mean:	$\operatorname{mean}(A, B) = A^{1/2} (A^{-1/2} B A^{-1/2})^{1/2} A^{1/2}$

Table 5.1: *Geometric toolbox for the Riemannian manifold $\mathbb{P}_+^n$ endowed with the affine-invariant metric*

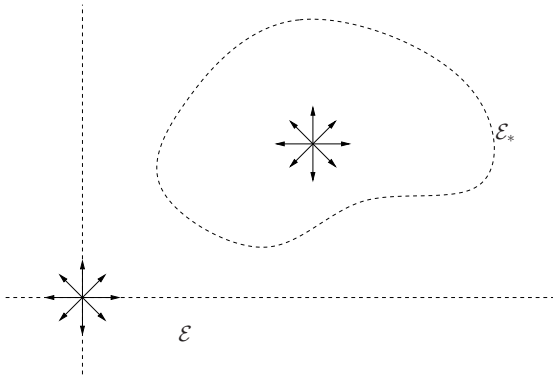

Figure 5.1: *Tangent vectors to an open subset $\mathcal{E}_*$ of a vector space $\mathcal{E}$. Figure courtesy of Absil et al., [AMS08].*

In loose terms, the metric described in table 5.1 stretches the limits of $\mathbb{P}_+^n$ to infinity. To visualize this, it is instructive to particularize the exponential map to positive numbers, i.e.,

positive-definite matrices of size 1. Starting at $A = 1$ and moving along the tangent vector $H = -1$, we follow the curve $\mathrm{Exp}_A(tH) = \exp(-t)$. We never reach non-positive numbers for finite t. The space deformation is stronger close to singular matrices.

We use the matrix exponentials and logarithms. These are defined as series derived from the Taylor expansions of their scalar equivalents:

$$\exp(A) = I + A + \frac{1}{2}A^2 + \ldots = \sum_{k=0}^{\infty} \frac{1}{k!}A^k \tag{5.1}$$

$$\log(A) = (A-I) - \frac{1}{2}(A-I)^2 + \frac{1}{3}(A-I)^3 - \ldots = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}(A-I)^k \tag{5.2}$$

The series for exp is convergent for any square matrix A. The series for log is derived from the real series $\log(1+x) = x - x^2/2 + x^3/3 - \ldots$, which we know is convergent if $|x| < 1$ (and divergent if $|x| > 1$). We thus expect the matrix counterpart to be convergent when $\rho(A-I) < 1$, where $\rho(X)$ is the spectral radius of X:

$$\rho(X) = \max\left(|\lambda_1|, \ldots, |\lambda_n|\right), \text{ with } \lambda_1, \ldots, \lambda_n \text{ the eigenvalues of } X.$$

This is indeed the case. On the other hand, we know that $\log(x)$ is well defined for $x > 0$. According to Higham, see [Hig08, problem 11.1], a Taylor series can be used to define the principal logarithm of any matrix having no eigenvalue on $\mathbb{R}^-$. A hint to this is that, using the identity

$$\log(A) = s\log(A^{1/s}),$$

it is always possible, with s big enough, to bring the eigenvalues of $A^{1/s}$ into a circle centered at 1 (in the complex plane) and with radius strictly smaller than 1. Consequently, the matrix $A^{1/s} - I$ has spectral radius less than 1 and the Taylor series (5.2) can be used. In particular, the matrix logarithm is well defined as such over $\mathbb{P}_+^n$. Furthermore, Bhatia and Arsigny et al., see [Bha07, AFPA08], show that the restriction

$$\exp : \mathbb{H}^n \to \mathbb{P}_+^n : H \mapsto \exp(H)$$

is a diffeomorphism between the metric spaces $\mathbb{H}^n$ and $\mathbb{P}_+^n$. The inverse mapping is the restriction $\log : \mathbb{P}_+^n \to \mathbb{H}^n$.

Every matrix $H \in \mathbb{H}^n$ can be diagonalized as $H = UDU^T$ with $U^TU = I$, the identity matrix, and D a diagonal matrix composed with the real eigenvalues λ_i of H. This yields easier formulas for the exponential:

$$\exp(H) = \sum_{k=0}^{\infty} \frac{1}{k!}\left(UDU^T\right)^k = U\left[\sum_{k=0}^{\infty}\frac{1}{k!}D^k\right]U^T = U\exp(D)U^T,$$

where $\exp(D)$ is the diagonal matrix whose entries are $\exp(\lambda_i)$. Same goes for the logarithm of a positive matrix $A = UDU^T$:

$$\log(A) = U\log(D)U^T.$$

5.2 Curve space and objective function

The curve space on $\mathbb{P}_+^n$ is given by

$$\Gamma = \mathbb{P}_+^n \times \ldots \times \mathbb{P}_+^n = (\mathbb{P}_+^n)^{N_\mathrm{d}}.$$

Each of the N_d points γ_i composing a curve γ is a positive matrix. The tools shown in table 5.1 can be used on Γ by component wise extension. The objective function E is defined as:

$$E : \Gamma \to \mathbb{R} : \gamma \mapsto E(\gamma) = E_\mathrm{d}(\gamma) + \lambda E_\mathrm{v}(\gamma) + \mu E_\mathrm{a}(\gamma) \tag{5.3}$$

$$
\begin{aligned}
E_{\mathrm{d}}(\gamma) &= \frac{1}{2}\sum_{i=1}^{N} w_i \operatorname{dist}^2(p_i, \gamma_{s_i}) \\
E_{\mathrm{v}}(\gamma) &= \frac{1}{2}\sum_{i=1}^{N_{\mathrm{d}}-1} \frac{1}{\Delta\tau_i} \operatorname{dist}^2(\gamma_i, \gamma_{i+1}) \\
E_{\mathrm{a}}(\gamma) &= \frac{1}{2}\left[\frac{\Delta\tau_1}{2}\|a_1\|^2_{\gamma_1} + \sum_{i=2}^{N_{\mathrm{d}}-1} \frac{\Delta\tau_{i-1}+\Delta\tau_i}{2}\|a_i\|^2_{\gamma_i} + \frac{\Delta\tau_{N_{\mathrm{d}}-1}}{2}\|a_{N_{\mathrm{d}}}\|^2_{\gamma_{N_{\mathrm{d}}}}\right]
\end{aligned} \tag{5.4}
$$

The p_i's are positive-definite matrices. This is to be compared to the definition given in section 4.2. The acceleration vectors $a_i \in T_{\gamma_i}\mathbb{P}^n_+ \equiv \mathbb{H}^n$ are symmetric matrices defined by

$$
\begin{aligned}
a_1 &= \frac{-2}{\Delta\tau_1\Delta\tau_2}\operatorname{Log}_{\gamma_1}(\gamma_2) + \frac{2}{\Delta\tau_2(\Delta\tau_1+\Delta\tau_2)}\operatorname{Log}_{\gamma_1}(\gamma_3), \\
a_i &= \frac{2}{\Delta\tau_{i-1}+\Delta\tau_i}\left[\frac{1}{\Delta\tau_i}\operatorname{Log}_{\gamma_i}(\gamma_{i+1}) + \frac{1}{\Delta\tau_{i-1}}\operatorname{Log}_{\gamma_i}(\gamma_{i-1})\right], \quad \forall i \in 2 \dots N_{\mathrm{d}}-1, \\
a_{N_{\mathrm{d}}} &= \frac{-2}{\Delta\tau_{N_{\mathrm{d}}-1}\Delta\tau_{N_{\mathrm{d}}-2}}\operatorname{Log}_{\gamma_{N_{\mathrm{d}}}}(\gamma_{N_{\mathrm{d}}-1}) + \frac{2}{\Delta\tau_{N_{\mathrm{d}}-2}(\Delta\tau_{N_{\mathrm{d}}-1}+\Delta\tau_{N_{\mathrm{d}}-2})}\operatorname{Log}_{\gamma_{N_{\mathrm{d}}}}(\gamma_{N_{\mathrm{d}}-2}).
\end{aligned}
$$

5.3 Gradient of the objective

We now compute the gradient of E as defined in the previous section. This requires formulas for the derivatives of the matrix logarithm, appearing in the distance function and the logarithmic map defined in table 5.1. We derive a generic way of computing derivatives of functions of symmetric matrices. The way we derive the material in this section is inspired by [FPAA07, appendix] and [Bha07].

Let $A \in \mathbb{H}^n$, U an orthogonal matrix and $D = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$, such that $A = UDU^T$. Let f be a smooth, real function defined by a Taylor series

$$
f(x) = \sum_{k=0}^{\infty} a_k x^k.
$$

Generalized to matrices, this yields

$$
f(A) = \sum_{k=0}^{\infty} a_k A^k = U \operatorname{diag}(f(\lambda_1), \dots, f(\lambda_n))\, U^T.
$$

We would like to compute $\mathrm{D}f(A)[H]$, the directional derivative of f at A in the direction $H \in \mathbb{H}^n$. We differentiate the series term by term. To this end, we need the following result:

$$
\mathrm{D}(X \mapsto X^k)(A)[H] = \lim_{h\to 0} \frac{(A+hH)^k - A^k}{h} = \sum_{l=1}^{k} A^{l-1} H A^{k-l}
$$

Then,

$$
\begin{aligned}
\mathrm{D}f(A)[H] &= \sum_{k=1}^{\infty} a_k \sum_{l=1}^{k} A^{l-1} H A^{k-l} \\
&= U\left[\sum_{k=1}^{\infty} a_k \sum_{l=1}^{k} D^{l-1} U^T H U D^{k-l}\right] U^T \\
&= U \cdot \mathrm{D}f(D)[U^T H U] \cdot U^T
\end{aligned}
$$

which is a simpler object to compute because D is diagonal. Let us write $\tilde{H} = U^T H U$ and

$M = \mathrm{D}f(D)[\tilde{H}]$. Entry-wise, we get:

$$\begin{aligned} M_{ij} &= \sum_{k=1}^{\infty} a_k \sum_{l=1}^{k} (D^{l-1}\tilde{H}D^{k-l})_{ij} \\ &= \sum_{k=1}^{\infty} a_k \sum_{l=1}^{k} \lambda_i^{l-1}\lambda_j^{k-l}\tilde{H}_{ij} \\ &= \tilde{H}_{ij} \sum_{k=1}^{\infty} a_k \frac{\lambda_j^k}{\lambda_i} \sum_{l=1}^{k} \left(\frac{\lambda_i}{\lambda_j}\right)^l. \end{aligned}$$

Using the identity $\sum_{l=1}^{k} x^k = x\frac{1-x^k}{1-x}$, valid for $x \neq 1$, we distinguish two cases:

$$\frac{\lambda_j^k}{\lambda_i} \sum_{l=1}^{k} \left(\frac{\lambda_i}{\lambda_j}\right)^l = \begin{cases} \frac{\lambda_i^k - \lambda_j^k}{\lambda_i - \lambda_j}, & \text{if } \lambda_i \neq \lambda_j \\ k\lambda_i^{k-1}, & \text{if } \lambda_i = \lambda_j \end{cases}$$

Hence:

$$M_{ij} = \tilde{H}_{ij}\tilde{f}(\lambda_i, \lambda_j),$$

with:

$$\tilde{f}(\lambda_i, \lambda_j) = \begin{cases} \frac{f(\lambda_i) - f(\lambda_j)}{\lambda_i - \lambda_j}, & \text{if } \lambda_i \neq \lambda_j \\ f'(\lambda_i), & \text{if } \lambda_i = \lambda_j \end{cases}$$

The coefficients $\tilde{f}(\lambda_i, \lambda_j)$ are termed the *first divided differences* by Bhatia, see [Bha07, p. 60]. We build the matrix $\tilde{F}$ such that $\tilde{F}_{ij} = \tilde{f}(\lambda_i, \lambda_j)$. Then,

$$M = \tilde{H} \odot \tilde{F},$$

where $\odot$ stands for entry-wise multiplication (Hadamard's product, also known as Schur's product). Setting $f = \log$, we explicitly compute $\mathrm{D}\log(A)[H]$, $A \succ 0, H = H^T$, as follows:

1. Diagonalize A: $A = UDU^T$, U orthogonal, $D = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_n)$,
2. Compute $\tilde{H} = U^T H U$,
3. Compute $\tilde{F}$ (first divided differences of log based on the λ_i's),
4. Compute $M = \tilde{H} \odot \tilde{F}$,
5. Compute $\mathrm{D}\log(A)[H] = UMU^T$

By construction, $\mathrm{D}\log(A)[H]$ is symmetric.

Remark 5.3.1. *As one can see from this algorithm, it is not critical that H be symmetric nor that A be positive-definite. As long as A does not have eigenvalues on $\mathbb{R}^-$, diagonalizability is sufficient. We then use U^{-1} instead of U^T.*

The objective function components E_{d} and E_{v} are linear combinations of squared distances. To compute their gradients, we need only derive an expression for $\mathrm{grad}\, f_A(X)$, with

$$f_A(X) = \frac{1}{2}\,\mathrm{dist}^2\,(A, X) = \frac{1}{2}\left\langle \log\left(A^{-1/2}XA^{-1/2}\right), \log\left(A^{-1/2}XA^{-1/2}\right)\right\rangle.$$

By using these identities:

$$\begin{aligned} \mathrm{D}(f \circ g)(X)[H] &= \mathrm{D}f(g(X))[\mathrm{D}g(X)[H]] \text{ (composition rule)}, \\ \mathrm{D}(X \mapsto \langle f(X), g(X)\rangle)(X)[H] &= \langle \mathrm{D}f(X)[H], g(X)\rangle + \langle f(X), \mathrm{D}g(X)[H]\rangle \text{ (product rule), and} \\ \mathrm{D}(X \mapsto AXB)(X)[H] &= \lim_{h\to 0} \frac{A(X+hH)B - AXB}{h} = AHB, \end{aligned}$$

we work out the following formula:

$$\mathrm{D}f_A(X)[H] = \left\langle \mathrm{D}\log(A^{-1/2}XA^{-1/2})[A^{-1/2}HA^{-1/2}], \log(A^{-1/2}XA^{-1/2}) \right\rangle.$$

By definition 2.4.2, the gradient $\operatorname{grad} f_A(X)$ is the unique symmetric matrix verifying

$$\forall H \in \mathbb{H}^n, \quad \mathrm{D}f_A(X)[H] = \langle \operatorname{grad} f_A(X), H \rangle_X .$$

Considering $\left(\overline{H}_1, \ldots, \overline{H}_{\frac{n(n+1)}{2}}\right)$, an orthonormal basis of $\mathbb{H}^n$ endowed with the inner product $\langle \cdot, \cdot \rangle$, we can compute an orthonormal basis $\left(H_1, \ldots, H_{\frac{n(n+1)}{2}}\right)$ of $\mathbb{H}^n$ endowed with the inner product $\langle \cdot, \cdot \rangle_X$ as $H_i = X^{1/2}\overline{H}_i X^{1/2}$, since

$$\langle H_i, H_j \rangle_X = \left\langle X^{-1/2}H_iX^{-1/2}, X^{-1/2}H_jX^{-1/2} \right\rangle = \langle \overline{H}_i, \overline{H}_j \rangle = \delta_{ij},$$

where δ_{ij} is the Kronecker delta. We thus compute the gradient of f_A at X as follows:

$$\begin{aligned} \operatorname{grad} f_A(X) &= \sum_{k=1}^{\frac{n(n+1)}{2}} \mathrm{D}f_A(X)[H_k]H_k \\ &= \sum_{k=1}^{\frac{n(n+1)}{2}} \mathrm{D}f_A(X)[X^{1/2}\overline{H}_kX^{1/2}]X^{1/2}\overline{H}_kX^{1/2}. \end{aligned}$$

Collecting the relevant equations yields an explicit algorithm to compute $\operatorname{grad} f_A(X)$. Regretfully, this method requires $\frac{n(n+1)}{2}$ computations of $\mathrm{D}\log$. We can do better.

Remembering that for square matrices $\operatorname{trace}(AB) = \operatorname{trace}(BA)$, and with $X = UDU^T$:

$$\begin{aligned} \mathrm{D}\left(X \mapsto \frac{1}{2}\|\log(X)\|^2\right)(X)[H] &= \langle \mathrm{D}\log(X)[H], \log(X) \rangle \\ &= \operatorname{trace}\left(U(\tilde{H} \odot \tilde{F})U^T \cdot U\log(D)U^T\right) \\ &= \operatorname{trace}\left((\tilde{H} \odot \tilde{F})\log(D)\right) \\ &= \sum_{i=1}^{n} \log(\lambda_i)\tilde{H}_{ii}\tilde{F}_{ii} \\ &= \sum_{i=1}^{n} \log(\lambda_i)\lambda_i^{-1}\tilde{H}_{ii} \\ &= \operatorname{trace}\left(D^{-1}\log(D)\tilde{H}\right) \\ &= \operatorname{trace}\left(U^TUD^{-1}U^TU\log(D)U^TU\tilde{H}U^TU\right) \\ &= \operatorname{trace}\left(X^{-1}\log(X)H\right) \\ &= \left\langle X^{-1}\log(X), H \right\rangle, \end{aligned}$$

$$\begin{aligned} \mathrm{D}f_A(X)[H] &= \left\langle A^{1/2}X^{-1}A^{1/2}\log(A^{-1/2}XA^{-1/2}), A^{-1/2}HA^{-1/2} \right\rangle \\ &= \left\langle X^{-1}\log(XA^{-1}), H \right\rangle \\ &= \left\langle \log(XA^{-1})X, H \right\rangle_X, \end{aligned}$$

where we used $A\log(B)A^{-1} = \log(ABA^{-1})$. Hence,

$$\operatorname{grad} f_A(X) = \log(XA^{-1})X.$$

This is a symmetric matrix. Indeed,

$$\log(XA^{-1})X = \log\left(X^{1/2}X^{1/2}A^{-1}X^{1/2}X^{-1/2}\right)X = X^{1/2}\log\left(X^{1/2}A^{-1}X^{1/2}\right)X^{1/2}.$$

From this equation, we also see that $\operatorname{grad} f_A(B) = -\operatorname{Log}_B(A)$, which makes perfect sense: to increase the distance between A and B as fast as possible with an infinitesimal change, B ought to move away from A. By the symmetry $f_A(B) = f_B(A) = \frac{1}{2}\operatorname{dist}^2(A,B)$,

$$\operatorname{grad}\left(A \mapsto \frac{1}{2}\operatorname{dist}^2(A,B)\right)(A) = \operatorname{grad} f_B(A) = \log(AB^{-1})A,$$
$$\operatorname{grad}\left(B \mapsto \frac{1}{2}\operatorname{dist}^2(A,B)\right)(B) = \operatorname{grad} f_A(B) = \log(BA^{-1})B.$$

Figure 5.2 compares $\langle \operatorname{grad} f_A(X), H\rangle_X$ to a centered difference for some fixed A, X, H. This figure provides further evidence that our formula for $\operatorname{grad} f_A(X)$ is correct.

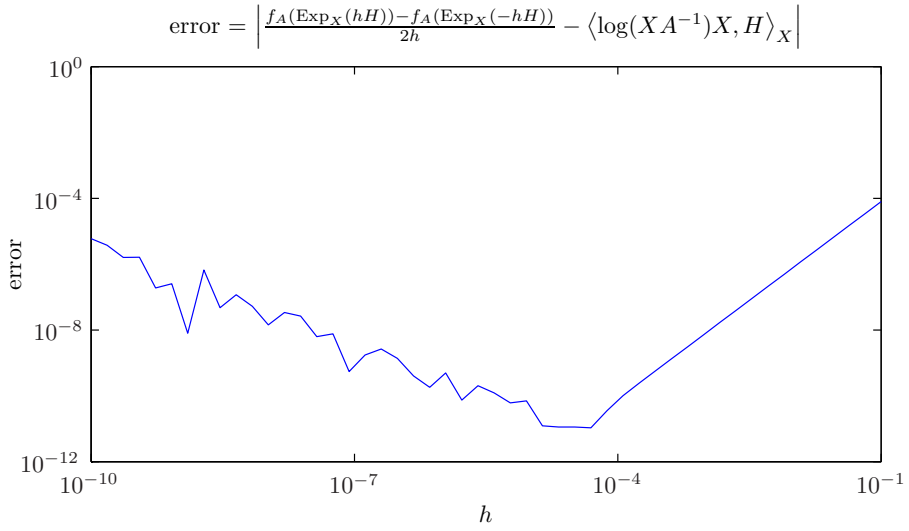

Figure 5.2: *Comparison of* $\frac{f_A(\operatorname{Exp}_X(hH)) - f_A(\operatorname{Exp}_X(-hH))}{2h}$ *with* $\langle \operatorname{grad} f_A(X), H\rangle_X$ *(order of magnitude: 1) for some fixed* A, X, H*. The slope of the straight segment is 2. Typically, the breaking point occurs for* h *close to* 10^{-5}*, due to numerical accuracy. This plot gives us confidence in the formula derived for* $\operatorname{grad} f_A(X)$*. Similar graphs have been generated to check every gradient formula given in this document, including* $\operatorname{grad} E$ *itself.*

Deriving the gradient for E_a is more cumbersome. We construct an explicit formula for it in appendix B. We still highlight the important points here. First, we needed to prove the following identity, for three square matrices A, B, C of size n:

$$\begin{aligned}\langle A \odot B, C\rangle &= \operatorname{trace}((A \odot B)C) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}B_{ij}C_{ji} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}(B^T)_{ji}(C^T)_{ij} \\ &= \left\langle A \odot C^T, B^T\right\rangle \end{aligned} \tag{5.5}$$

We also needed to prove the following proposition.

Proposition 5.3.2. *Let* $A, B \in \mathbb{P}_+^n$*. Then* AB^{-1} *has real, positive eigenvalues and is diagonalizable.*

Proof. We note $\Lambda(A)$ the spectrum of A, i.e., the set of eigenvalues of A. We know that, for any invertible matrix S, $\Lambda(SAS^{-1}) = \Lambda(A)$. Indeed:

$$1 = \det(I) = \det(SS^{-1}) = \det(S)\det(S^{-1}),$$
$$\det(SAS^{-1} - \lambda I) = \det(S(A - \lambda I)S^{-1}) = \det(S)\det(A - \lambda I)\det(S^{-1}) = \det(A - \lambda I).$$

Then,

$$\Lambda(AB^{-1}) = \Lambda(A^{1/2}A^{1/2}B^{-1}A^{1/2}A^{-1/2}) = \Lambda(A^{1/2}B^{-1/2}B^{-1/2}A^{1/2}) \\ = \Lambda((B^{-1/2}A^{1/2})^T(B^{-1/2}A^{1/2}))$$

Furthermore, any symmetric matrix of the form X^TX with $\ker X = \{0\}$ is positive-definite, since

$$\forall x \in \mathbb{R}^n, x \neq 0,\ x^TX^TXx = \|Xx\|^2 > 0.$$

Setting $X = B^{-1/2}A^{1/2}$, we see that AB^{-1} has the spectrum of a positive-definite matrix. We diagonalize the symmetric matrix $A^{1/2}B^{-1}A^{1/2} = UDU^T$, U orthogonal and D diagonal. Define $V = U^TA^{-1/2}$. $VAB^{-1}V^{-1} = D$ is diagonal, hence AB^{-1} is diagonalizable. □

The joint material of this section and of appendix B is sufficient to implement direct algorithms to compute the gradient of E. We use these algorithms in our descent methods from chapter 3. In section 5.7, we illustrate our results.

5.4 Alternative I: linear interpolation

$\mathbb{P}_+^n$ is a convex set. Indeed, let $A, B \in \mathbb{P}_+^n$. Then, $tA + (1-t)B \in \mathbb{P}_+^n$ for all $t \in [0,1]$, since

$$\forall x \in \mathbb{R}^n, x \neq 0,\ x^T(tA + (1-t)B)x = tx^TAx + (1-t)x^TBx > 0.$$

Hence, to interpolate between two points $p_1, p_2 \in \mathbb{P}_+^n$, we can simply interpolate between the (real) entries of p_1 and p_2, linearly. This alternative to our geometric approach is so simple that we have to consider it. In section 5.7, we compare our main results for piecewise geodesic interpolation to this alternative. Of course, we cannot use linear interpolation to solve the generic problem of smooth regression with more than two data points.

5.5 Alternative II: convex programming

When we endowed $\mathbb{P}_+^n$ with the affine-invariant metric, we "stretched" the limits of the (open) set $\mathbb{P}_+^n$ to infinity, making the resulting metric space complete. This ensures that solutions of the regression problem are made of matrices $\gamma_i \succ 0$. The Log-Euclidean metric, which we introduce in the next section, has a similar stretching effect, needed to map the convex set $\mathbb{P}_+^n$ onto the vector space $\mathbb{H}^n$.

We can sometimes solve the problem without such deformations. Using the metric $\langle \cdot, \cdot \rangle$ defined on $\mathbb{H}^n$ and the associated norm $\| \cdot \|$, $\mathbb{P}_+^n$ is now seen as a Riemannian submanifold of $\mathbb{H}^n$. A natural objective for the regression problem is given by equations (1.3) through (1.6). The acceleration vectors are defined using finite differences in the ambient space $\mathbb{H}^n$.

This objective is convex. The set Γ is convex too. Standard solvers for semidefinite programming, like SeDuMi or SDPT3 [Stu98, TTT99], can solve a relaxed version of our problem: minimize $E(\gamma)$ such that each γ_i is positive semidefinite. The usage of such solvers is made easy via modeling languages such as Yalmip and CVX [LÖ4, GB10]. We use the latter with the Frobenius norm for the figures shown in section 5.7. If, at optimality, none of the $\gamma_i \succeq 0$ constraints is active (i.e., $\gamma_i \succ 0, \forall i$) the optimal solution corresponds to regression in $\mathbb{H}^n$. This can be done with the simple tools from chapter 1. If, however, one of the constraints is active, there is, a priori, no way to convert the solution of the relaxed problem into a solution for our original problem. The original problem might even not have a solution, specifically because Γ is not complete. We exhibit this in figure 5.5.

Among the advantages of convex programming, we note that:

- efficient, robust algorithms are available,
- and provide optimality certificates;

- we can use 1, 2, ∞ and Frobenius norms over $\mathbb{H}^n$;
- we need only provide the problem description (no gradients);
- it is easy to add convex constraints.

The affine-invariant and Log-Euclidean distances, seen as functions from $\mathbb{P}_+^n \times \mathbb{P}_+^n$ to $\mathbb{R}^+$ where $\mathbb{P}_+^n$ is understood as a convex subset of the vector space $\mathbb{H}^n$, are not convex. Indeed, for $n = 1$, $\mathbb{P}_+^n$ is the set of positive real numbers. The two metrics are equal, such that $\text{dist}^2(x, y) = (\log(y) - \log(x))^2$. The Hessian of this function at $(x, y) = (1, 2)$ has a negative eigenvalue, hence it is not convex. Consequently, convex programming cannot be used to solve the regression problem with the affine-invariant and Log-Euclidean metrics.

5.6 Alternative III: vector space structure

We mentioned in section 5.1 that the restricted matrix exponential

$$\exp : \mathbb{H}^n \to \mathbb{P}_+^n : H \mapsto \exp(H)$$

is a smooth diffeomorphism. Following [AFPA08], this can be used to endow $\mathbb{P}_+^n$ with a vector space structure. We first introduce new summation and scaling operators:

$$\begin{aligned} \oplus : \mathbb{P}_+^n \times \mathbb{P}_+^n \to \mathbb{P}_+^n : (A, B) \to A \oplus B = \exp(\log(A) + \log(B)), \\ \otimes : \mathbb{R} \times \mathbb{P}_+^n \to \mathbb{P}_+^n : (\alpha, A) \to \alpha \otimes A = \exp(\alpha \log(A)) = A^\alpha. \end{aligned}$$

By construction,

$$\exp : (\mathbb{H}^n, +, .) \to (\mathbb{P}_+^n, \oplus, \otimes)$$

is a vector space isomorphism. The inverse mapping is the (principal) matrix logarithm. Arsigny et al. use this to endow $\mathbb{P}_+^n$ with the metric described in table 5.2, termed the Log-Euclidean metric, see [AFPA08]. The vector space structure simplifies a number of problems. For example, the geodesic between A and B is $\gamma(t) = \exp(t \log(A) + (1 - t) \log(B))$.

Set:	$\mathbb{P}_+^n = \{A \in \mathbb{R}^{n \times n} \ : \ A = A^T \text{ and } x^T A x > 0 \ \forall x \in \mathbb{R}^n, x \neq 0\}$
Tangent spaces:	$T_A \mathbb{P}_+^n \equiv \mathbb{H}^n = \{H \in \mathbb{R}^{n \times n} \ : \ H = H^T\}$
Inner product:	$\langle H_1, H_2 \rangle_A = \langle \mathrm{D} \log(A)[H_1], \mathrm{D} \log(A)[H_2] \rangle$
Vector norm:	$\|H\|_A = \sqrt{\langle H, H \rangle_A}$
Distance:	$\text{dist}(A, B) = \|\log(B) - \log(A)\|$
Exponential:	$\text{Exp}_A(H) = \exp(\log(A) + \mathrm{D} \log(A)[H])$
Logarithm:	$\text{Log}_A(B) = \mathrm{D} \exp(\log(A))[\log(B) - \log(A)]$
Mean:	$\text{mean}(A, B) = \exp(.5(\log(A) + \log(B)))$

Table 5.2: *Geometric toolbox for the Riemannian manifold $\mathbb{P}_+^n$ endowed with the Log-Euclidean metric.*

Let $A, B \in \mathbb{P}_+^n$ be commuting matrices, i.e., $AB = BA$. Then, there exists U, orthogonal, such that $A = U D_A U^T$ and $B = U D_B U^T$, with D_A and D_B diagonal, see [Bha07, p. 23][2]. Using commutativity of diagonal matrices:

$$\begin{aligned} \| \log(B) - \log(A) \|^2 &= \| U(\log(D_B) - \log(D_A)) U^T \|^2 \\ &= \| U \log(D_B D_A^{-1}) U^T \|^2 \\ &= \| U \log({D_A}^{-1/2} D_B {D_A}^{-1/2}) U^T \|^2 \\ &= \| U \log(U^T A^{-1/2} U U^T B U U^T A^{-1/2} U) U^T \|^2 \\ &= \| \log(A^{-1/2} B A^{-1/2} \|^2 \end{aligned}$$

[2] In [Bha07], Bhatia states that it is sufficient to have $A, B \in \mathbb{H}^n$ for this to be true

This proves that, if A and B commute, the affine-invariant and the Log-Euclidean metric are identical. In particular, this is true when A and B are diagonal. This is linked to the fact that, for commuting A and B, $\log(AB) = \log(A) + \log(B) = \log(BA)$.

We solve the smooth regression problem on $\mathbb{P}_+^n$ endowed with the Log-Euclidean metric as follows:

1. Compute the new data points $\tilde{p}_i = \log(p_i) \in \mathbb{H}^n$ for each original data point $p_i \in \mathbb{P}_+^n$;
2. Compute $\tilde{\gamma} \in \mathbb{H}^n \times \ldots \times \mathbb{H}^n$, the solution of the regression problem in $\mathbb{H}^n$ with the new data points;
3. Compute γ, the solution of the original problem, as $\gamma_i = \exp(\tilde{\gamma}_i)$.

Step 2 can be carried out by solving a regression problem in $\mathbb{R}^{\frac{n(n+1)}{2}}$ as described in chapter 1. Thanks to the vector space structure, we achieve guaranteed minimization of the objective by solving sparse, structured linear systems. Furthermore, we need not go through the hassle of computing gradients like we did in section 5.3. Using modern QP solvers, it is also easy to add constraints to the original problem in the form of linear equalities and inequalities. Linear matrix inequalities can be added using semidefinite programming solvers, see section 5.5.

5.7 Results and comments

The solutions to our regression problem on $\mathbb{P}_+^n$ can be categorized in the same way we did in section 4.4 for the sphere. In this section, we show plots for different scenarios. We compare linear interpolation, convex programming, the Log-Euclidean metric and the affine-invariant metric. Positive-definite matrices of size $n = 2$ are represented as ellipses. The axis orientations of these ellipses correspond to the eigenvector directions and the axis lengths to the corresponding eigenvalues.

In many special cases, some of these methods yield the same solutions. For example, we proved that when the data points p_i are commuting matrices, the affine-invariant and the Log-Euclidean metrics are equivalent. Of course, it is then computationally advantageous to solve the problem with the LE metric. Because of this, we always use the LE metric solution as our initial guess for the affine-invariant metric optimization. Additionally, since the convex programming approach uses the Frobenius norm, the shortest path between two matrices corresponds to linear interpolation in $\mathbb{H}^n$.

We use the CG algorithm with the following naive vector transport:

$$\mathsf{T}_\eta(\xi) = \xi$$

This means that we simply compare tangent vectors belonging to different tangent spaces as is. We expect this to work because our descent algorithms usually make small steps. Because of the smoothness of the inner product on Riemannian manifolds, close tangent spaces from $\mathbb{P}_+^n$ "resemble each other". In practice, we observe that the CG algorithm with this vector transport performs about as well as the steepest descent method, and sometimes much better.

We propose an alternative vector transport, theoretically more attractive, based on the following vector transport on $\mathbb{P}_+^n$, with $H \in T_A\mathbb{P}_+^n$:

$$\mathsf{T}_{\mathrm{Log}_A(B)}(H) = B^{1/2}A^{-1/2}HA^{-1/2}B^{1/2} \in T_B\mathbb{P}_+^n$$

It has the nice property that, for any two tangent vectors $H_1, H_2 \in T_A\mathbb{P}_+^n$, noting $\overline{H}_i = \mathsf{T}_{\mathrm{Log}_A(B)}(H_i) \in T_B\mathbb{P}_+^n$, we have:

$$\begin{aligned}\left\langle \overline{H}_1, \overline{H}_2 \right\rangle_B &= \left\langle B^{-1/2}\overline{H}_1B^{-1/2}, B^{-1/2}\overline{H}_2B^{-1/2} \right\rangle \\ &= \left\langle A^{-1/2}H_1A^{-1/2}, A^{-1/2}H_2A^{-1/2} \right\rangle = \langle H_1, H_2 \rangle_A .\end{aligned}$$

Surprisingly, the CG method with this vector transport extended to Γ showed significantly poorer performances on the tests we ran. Consequently, in the following, we use $\mathsf{T}_\eta(\xi) = \xi$.

Zeroth order

The weighted mean $\overline{p}$ of positive-definite matrices can be defined as

$$\overline{p} = \arg\min_{X \in \mathbb{P}_+^n} \frac{\sum_{i=1}^N w_i f_X(p_i)}{\sum_{i=1}^N w_i}. \tag{5.6}$$

Recall that $f_X(A) = \frac{1}{2}\operatorname{dist}^2(X, A)$. For the Log-Euclidean metric, we simply have

$$\overline{p} = \exp\left(\frac{\sum_{i=1}^N w_i \log(p_i)}{\sum_{i=1}^N w_i}\right).$$

This is a generalization of the geometric mean for positive reals. For the affine-invariant metric, we use the Log-Euclidean mean as an initial guess and apply one of our minimization algorithms to the expression (5.6). When the p_i's commute, the initial guess is optimal. Otherwise, convergence (i.e., $\|\operatorname{grad} E\| < 10^{-9}$) with 250 random matrices of size 5 is achieved with any of our algorithms in, typically, three steps or less. In [Bha07], Bhatia proves that the optimization problem (5.6) is strictly convex in some sense.

First order

Setting $\lambda > 0$ and $\mu = 0$, again, produces piecewise geodesic regressions. Figure 5.3 compares solutions obtained with our main method and the three mentioned alternatives. Figure 5.4 shows the behavior of the CG and the steepest descent methods. Both converge rapidly. Many of the remarks we made in section 4.4 for the sphere still hold. More specifically, notice how the curves obtained with the deforming metrics pass through thinner matrices than the others. This effect has already been observed by Arsigny et al. for the geodesic joining two data points, see [AFPA08]. The geodesic between A and B has a closed-form expression for the affine-invariant metric:

$$\begin{aligned}\gamma_{A,B} : [0,1] \to \mathbb{P}_+^n : t \mapsto \gamma_{A,B}(t) &= \operatorname{Exp}_A\left(t \operatorname{Log}_A(B)\right) \\ &= A^{1/2} \exp\left(t \log\left(A^{-1/2} B A^{-1/2}\right)\right) A^{1/2}\end{aligned} \tag{5.7}$$

Second order

For geodesic regression (high μ, zero λ), we simply compute the γ_i's at the data time labels t_i, because we use the exact logarithmic map on $\mathbb{P}_+^n$ (see our discussion in subsection 4.4.1). This drastically decreases the computational cost. We show an example in figure 5.6. Geodesics built with the deforming metrics can be indefinitely extended on both ends, whereas the regression built with convex programming would, eventually, reach a singular matrix.

Setting $\lambda = 0$ and $\mu > 0$ yields spline-like curves. Figure 5.7 shows an example for the four solving techniques we discussed. Computing the solution for the affine-invariant metric is computationally more intensive, as figure 5.8 demonstrates. The CG algorithm, despite our poor choice of vector transport, performs much better than the steepest descent method. SD needed 3.5 times as many iterations for equal quality (judged based on the gradient norm). The refinement technique, algorithm 6, cut down the number of needed iterations by about 20% for the example on figure 5.7, compared to using the Log-Euclidean solution as initial guess directly.

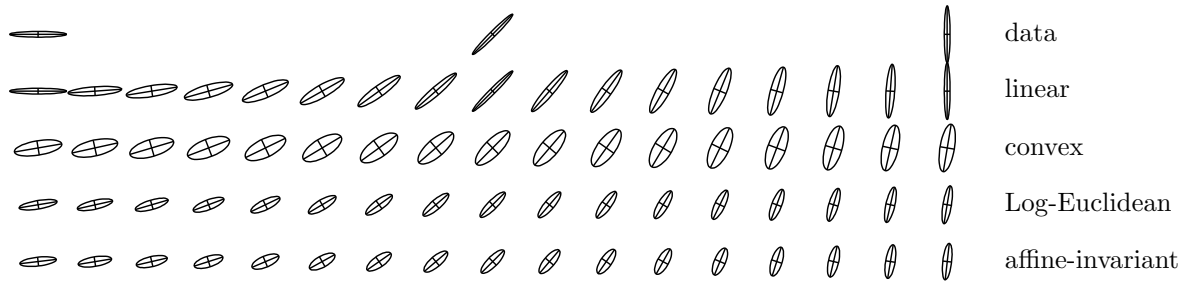

Figure 5.3: $\lambda = 10^{-1}, \mu = 0$. *The data consists in three positive-definite, non-commuting matrices shown on the first line. The second line shows linear interpolation between the data. The third line is obtained by solving the problem with SDPT3, using CVX. When λ goes to zero, this goes to linear interpolation. The fourth line is computed by solving a linear least-squares problem on the* $\log(p_i)'s$ *and going back on* $\mathbb{P}_+^n$ *via the matrix exponential. For the fifth line, we use the Log-Euclidean solution at the three data times t_i as initial guess for the breaking points. We apply a minimization algorithm starting with them. The additional intermediate points are computed with equation* (5.7)*. This is still optimal. Computation times for the three last lines are comparable.*

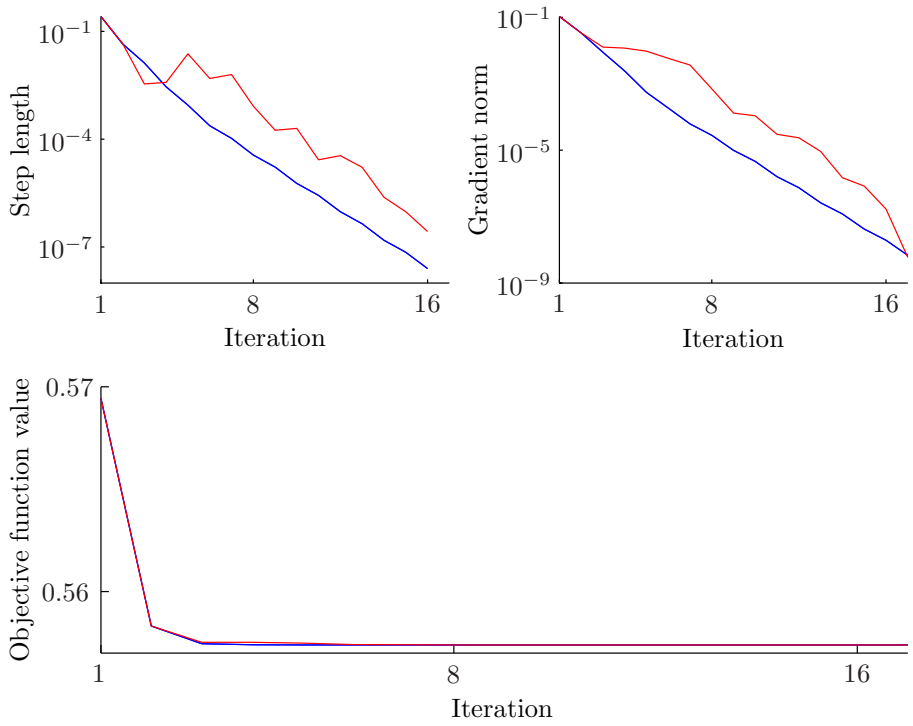

Figure 5.4: *These plots show the behavior of the minimization process for the affine-invariant metric solution shown in figure 5.3. Both the geometric steepest descent method (blue) and the non-linear geometric CG method (red) perform really well. The important point here is that we need only compute the γ_i's at the data times (the breaking points), i.e., we search for just three positive-definite matrices.*

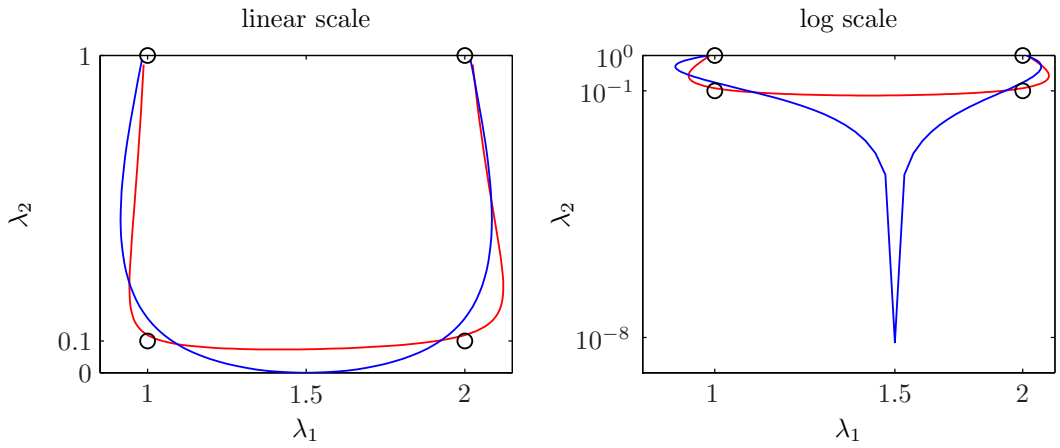

Figure 5.5: $\lambda = 0, \mu = 10^{-4}$. *The four data points are* 2×2 *positive-definite diagonal matrices. They are pictured as black circles, placed in the eigenvalue plane in linear scale (left) and logarithmic scale (right). Since the data matrices commute, the solution with the affine-invariant and the Log-Euclidean metrics are the same. The red curve shows regression with these metrics. The blue curve shows regression with convex programming (metric inherited from* $\mathbb{H}^n$*). SDPT3, our semidefinite programming solver, claims optimality (according to the default tolerances). Notice how the blue curve passes through almost singular matrices. The actual solution most likely passes through singular matrices. SDPT3 uses an interior point method, hence constraints are never tight. The fact that the optimal curve leaves* $\mathbb{P}_+^n$ *is a main argument against convex programming in this setting.*

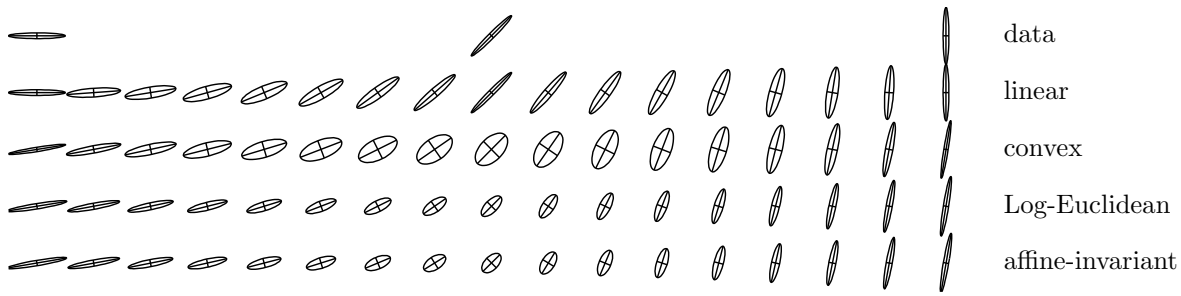

Figure 5.6: $\lambda = 0, \mu = 10^3$. *An example of geodesic regression across three data points with our three solving techniques, and linear interpolation (second line). Convex and Log-Euclidean metric solutions (third and fourth lines) are exact (up to numerical accuracy) and quick to compute. The affine-invariant metric solution (bottom line) is reasonably quick to compute. On this plot, the curve displayed has maximum acceleration of* 10^{-3} *for a total length of* 3.3 *traveled in one time unit.*

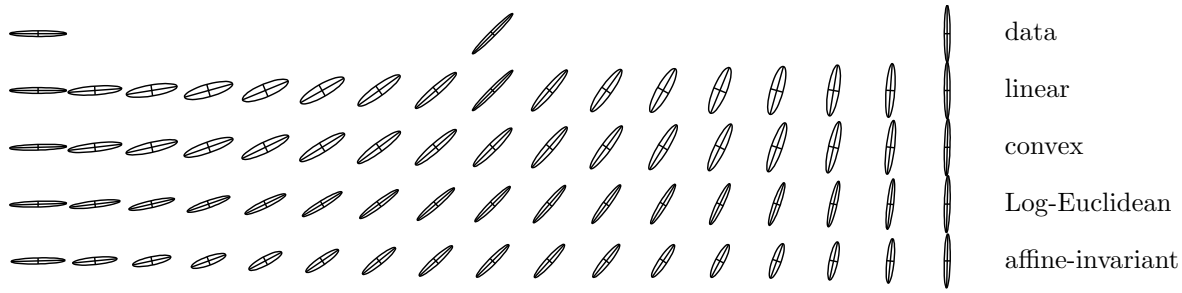

Figure 5.7: $\lambda = 0, \mu = 10^{-3}$. *The three methods of interest (bottom lines) give sensible results. Which one should use in practice is application-dependent. Of course, piecewise linear interpolation is not acceptable when spline-like curves are required.*

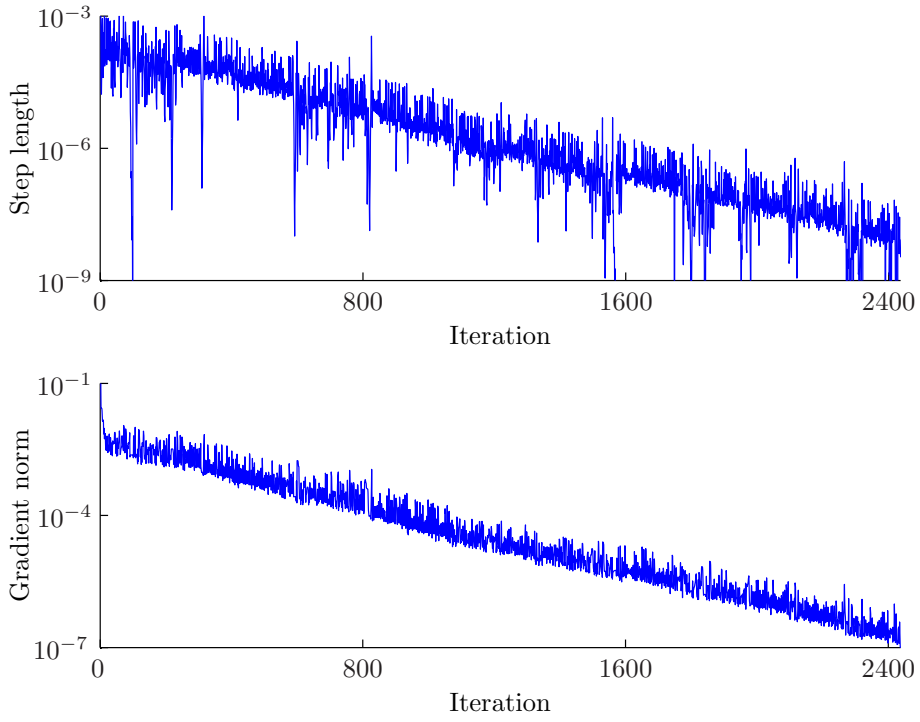

Figure 5.8: *These plots show the convergence behavior for figure 5.7 with the CG algorithm and the successive refinement technique. Notice how the CG algorithm exhibits a linear convergence rate. Convergence is excessively slow compared to alternatives II and III which gave almost instantaneous results.*

Conclusions and perspectives

In the present work, we discretized the problem of fitting smooth curves to data on manifolds (1.2) and we designed and implemented algorithms to solve the discrete problem (3.6). To this end, we introduced the concept of geometric finite differences, which fill the classic role of finite differences on manifolds, and computed explicit formulas for the objective (3.6) and its gradient on $\mathbb{R}^n, \mathbb{S}^2, \mathbb{P}^n_+$ and SO(3). The optimization schemes we used are various flavors of a geometric version of the non-linear conjugate gradient scheme taken from [AMS08]. To alleviate the computational burden, we exploited the concepts of retraction and vector transport, borrowed from [AMS08], as proxies for the expensive exponential map and parallel transport. We also introduced a similar concept for the logarithmic map, which we here call generalized logarithms.

Performances were shown to be excellent for first order problems ($\mu = 0$): our optimization algorithms reach small values of the gradient in a few iterations. The more interesting case of second order regression ($\mu > 0$), however, is solved significantly more slowly. The iterative refinement technique we introduced, although it has proven valuable, only partly helps. Furthermore, the algebra needed to obtain the gradient of (3.6) on matrix manifolds turned out to be quite involved. This adds to the intricacy of implementing the algorithms we designed. The attractive alternatives we proposed on $\mathbb{P}^n_+$ are further arguments pointing toward the need for faster, simpler algorithms. Two options we intend to pursue in future work to address these concerns are

1. the usage of automatic differentiation techniques or finite difference approximations of the derivatives of the objective, and
2. the implementation of higher-order optimization schemes like, e.g., the geometric trust-region method exposed in [AMS08].

Let us mention a few research directions we would like to investigate in future work.

- We believe that, as $\max_i |\tau_{i+1} - \tau_i| \to 0$, the solution of the discrete problem may tend toward a sampled version of the solution of the continuous problem. We intend to work on a proof of this statement. On a related note, we would like to verify that the breaking points appearing in optimal curves for $\lambda > 0, \mu = 0$ in the continuous case coincide with the breaking points computed with our discrete approach.
- Geodesic regression is a special case of interest in applications, but is slow to achieve with our present algorithms. A geodesic on $\mathcal{M}$ is completely specified by an element of the tangent bundle $T\mathcal{M}$, that is: a point $p \in \mathcal{M}$ and a tangent vector $\xi \in T_p\mathcal{M}$. We would like to, generically, equip the manifold $T\mathcal{M}$ with a toolbox based on the toolbox for $\mathcal{M}$ so we could apply the algorithms described in this document to efficiently compute geodesic regressions.
- To apply our techniques to a manifold $\mathcal{M}$, one needs a geometric toolbox for $\mathcal{M}$, possibly a vector transport and the gradients of the functions $f(A, B) = \operatorname{dist}^2(A, B)$ and $g(A, B, C) = \langle \operatorname{Log}_A(B), \operatorname{Log}_A(C) \rangle_A$ with respect to all variables. At some point, it would be interesting to build software that only asks the user for these elements. This would enable fast prototyping and usefulness assessment. For all manifolds treated in this document, we found that $\operatorname{grad}(A \mapsto f(A, B))(A) = -2 \operatorname{Log}_A(B)$, which makes sense geometrically.

This is a general property, see [SASK09, Theorem 3.1]. This observation partly alleviates the implementation burden for second order problems ($\mu > 0$). For first order problems ($\mu = 0$), no gradient formulas have to be computed at all.

- A typical strategy in (discrete-time) control is to compute the optimal commands to apply to the system up to some distant horizon in the future, then only apply the first computed command. The criterion is usually expressed in terms of the predicted future states of the system, based on the proposed command. It is customary to also include a regularization term on the command in the objective function. A well-established strategy to tackle this problem is, e.g., Model Predictive Control. In practical applications, the system is often subject to constraints. When the admissible set for the command is a manifold $\mathcal{M}$, we really are looking for an optimal smooth curve on $\mathcal{M}$. Investigating these applications is a natural extension of our work.

- The software written for this work[3] is only a proof of concept. The rich structure of the problems we treat gives plenty of opportunities for efficient organization of the computations, which should drastically improve the performances. When our algorithms reach maturity, it will be useful to design and distribute efficient, usable software.

- Other manifolds of practical interest are, e.g., the Grassmann (quotient) manifold and the (infinite-dimensional) shape space. Smooth interpolation in the shape space corresponds to smooth morphing between given shapes, with applications in computer graphics, object tracking...

The positive results we obtained in the present work and the appealing questions left to answer strongly encourage us to further our investigations.

[3]Downloadable here: `http://www.inma.ucl.ac.be/~absil/Boumal/`

Appendix A

Line search derivative on $\mathbb{S}^2$

We give an explicit formula for the derivative of the line search function $\phi : \mathbb{R} \to \mathbb{R}$ appearing in the descent methods applied on the sphere $\mathbb{S}^2$. The equivalent formula for the line search function on $\Gamma = \mathbb{S}^2 \times \ldots \times \mathbb{S}^2$ is a component wise extension of the one given below.

With $x \in \mathbb{S}^2$, $d \in T_x\mathbb{S}^2$ and $\|d\| = 1$:

$$
\begin{aligned}
\mathsf{R}_x(\alpha d) &= \frac{x + \alpha d}{\sqrt{1+\alpha^2}} \\
\frac{\mathrm{d}}{\mathrm{d}\alpha}\mathsf{R}_x(\alpha d) &= \frac{1}{(1+\alpha^2)^{\frac{3}{2}}}(d - \alpha x) \in T_{\mathsf{R}_x(\alpha d)}\mathbb{S}^2 \\
\phi(\alpha) &= E(\mathsf{R}_x(\alpha d)) \\
&= \overline{E}(\mathsf{R}_x(\alpha d)) \\
\phi'(\alpha) &= \left\langle \nabla\overline{E}(\mathsf{R}_x(\alpha d)), \frac{\mathrm{d}}{\mathrm{d}\alpha}\mathsf{R}_x(\alpha d) \right\rangle \\
&= \frac{1}{(1+\alpha^2)^{\frac{3}{2}}} \left\langle \nabla\overline{E}(\mathsf{R}_x(\alpha d)), d - \alpha x \right\rangle \\
&= \frac{1}{(1+\alpha^2)^{\frac{3}{2}}} \left\langle \operatorname{grad} E(\mathsf{R}_x(\alpha d)), d - \alpha x \right\rangle_{\mathsf{R}_x(\alpha d)} \\
\phi'(0) &= \left\langle \operatorname{grad} E(x), d \right\rangle_x
\end{aligned}
$$

Appendix B

Gradient of E_{a} for $\mathbb{P}_+^n$

In this appendix, we derive an explicit formula for the gradient of E_{a} from chapter 5, equation (5.4). For $i \in 2, \ldots, N_{\mathrm{d}} - 1$, with the appropriate constants r_1, r_2:

$$\begin{aligned}
\frac{1}{2}\|a_i\|_{\gamma_i}^2 &= \frac{1}{2}\left\|r_1 \operatorname{Log}_{\gamma_i}(\gamma_{i+1}) + r_2 \operatorname{Log}_{\gamma_i}(\gamma_{i-1})\right\|_{\gamma_i}^2 \\
&= \frac{1}{2}\left\|{\gamma_i}^{-1/2}\left(r_1 \operatorname{Log}_{\gamma_i}(\gamma_{i+1}) + r_2 \operatorname{Log}_{\gamma_i}(\gamma_{i-1})\right){\gamma_i}^{-1/2}\right\|^2 \\
&= \frac{1}{2}\left\|r_1 \log\left({\gamma_i}^{-1/2}\gamma_{i+1}{\gamma_i}^{-1/2}\right) + r_2 \log\left({\gamma_i}^{-1/2}\gamma_{i-1}{\gamma_i}^{-1/2}\right)\right\|^2 \\
&= r_1^2 f_{\gamma_i}(\gamma_{i+1}) + r_2^2 f_{\gamma_i}(\gamma_{i-1}) + r_1 r_2 \left\langle \log\left({\gamma_i}^{-1/2}\gamma_{i+1}{\gamma_i}^{-1/2}\right), \log\left({\gamma_i}^{-1/2}\gamma_{i-1}{\gamma_i}^{-1/2}\right)\right\rangle.
\end{aligned}$$

We need to provide formulas for the gradient of the third term with respect to γ_i and γ_{i+1} (which is symmetric with γ_{i-1}). We introduce the real function g,

$$g : (\mathbb{P}_+^n)^3 \to \mathbb{R} : (A, B, C) \mapsto g(A, B, C) = \left\langle \log\left(A^{-1/2}BA^{-1/2}\right), \log\left(A^{-1/2}CA^{-1/2}\right)\right\rangle.$$

Using the identity $\langle A \odot B, C\rangle = \langle A \odot C^T, B^T\rangle$ we proved in section 5.3, we compute:

$$\begin{aligned}
\mathrm{D}\left(B \mapsto g(A, B, C)\right)(B)[H] &= \left\langle \mathrm{D}\log(A^{-1/2}BA^{-1/2})[A^{-1/2}HA^{-1/2}], \log(A^{-1/2}CA^{-1/2})\right\rangle \\
&\quad \text{Introducing } A^{-1/2}BA^{-1/2} = UDU^T, \tilde{H} = U^T A^{-1/2}HA^{-1/2}U \text{ and} \\
&\quad \tilde{F}, \text{ the first divided differences of } \log \text{ w.r.t. the } \lambda_i = D_{ii} \\
&= \left\langle U(\tilde{H} \odot \tilde{F})U^T, \log(A^{-1/2}CA^{-1/2})\right\rangle \\
&= \left\langle \tilde{H} \odot \tilde{F}, U^T \log(A^{-1/2}CA^{-1/2})U\right\rangle \\
&\quad \text{Using the identity (5.5):} \\
&= \left\langle [U^T \log(A^{-1/2}CA^{-1/2})U] \odot \tilde{F}, \tilde{H}\right\rangle \\
&= \left\langle A^{-1/2}U([U^T \log(A^{-1/2}CA^{-1/2})U] \odot \tilde{F})U^T A^{-1/2}, H\right\rangle
\end{aligned}$$

$$\operatorname{grad}\left(B \mapsto g(A, B, C)\right)(B) = BA^{-1/2}U\left([U^T \log(A^{-1/2}CA^{-1/2})U] \odot \tilde{F}\right)U^T A^{-1/2}B$$

We still need the gradient with respect to A. For this, we observe that:

$$\begin{aligned}
g(A, B, C) &= \left\langle \log\left(A^{-1/2}BA^{-1/2}\right), \log\left(A^{-1/2}CA^{-1/2}\right)\right\rangle \\
&= \left\langle A^{-1/2}\log\left(BA^{-1}\right)A^{1/2}, A^{-1/2}\log\left(CA^{-1}\right)A^{1/2}\right\rangle \\
&= \left\langle \log\left(BA^{-1}\right), \log\left(CA^{-1}\right)\right\rangle \\
&\quad \text{Using } \log(X) = -\log(X^{-1}): \\
&= \left\langle \log\left(AB^{-1}\right), \log\left(AC^{-1}\right)\right\rangle
\end{aligned}$$

The matrices AB^{-1} and AC^{-1} are not, in general, symmetric. They still have real, positive eigenvalues and are diagonalizable. We proved this statement in proposition 5.3.2. We introduce:

$$\begin{aligned}
AB^{-1} &= U_B D_B U_B^{-1} \\
AC^{-1} &= U_C D_C U_C^{-1} \\
\tilde{F}_B &= \text{first divided differences of log for } \lambda_i = (D_B)_{ii} \\
\tilde{F}_C &= \text{first divided differences of log for } \lambda_i = (D_C)_{ii} \\
Y_B &= U_B^{-1} \log(AC^{-1}) U_B = U_B^{-1} U_C \log(D_C) U_C^{-1} U_B \\
Y_C &= U_C^{-1} \log(AB^{-1}) U_C = U_C^{-1} U_B \log(D_B) U_B^{-1} U_C
\end{aligned}$$

Then, similar algebra yields

$$\operatorname{grad}\left(A \mapsto g(A,B,C)\right)(A) = U_B D_B (Y_B \odot \tilde{F}_B) U_B^{-1} A + U_C D_C (Y_C \odot \tilde{F}_C) U_C^{-1} A.$$

Although it is not obvious from it's expression, this matrix is, by construction, symmetric.

The equations presented in section 5.3 and in this appendix are sufficient to compute the gradient $\operatorname{grad} E(\gamma)$.

Appendix C

SO(3) geometric toolbox

Our algorithms and problem formulation from chapter 3 are well defined for any smooth, finite-dimensional Riemannian manifold. In practice, they can be applied to such manifolds only if the expressions for, notably, the logarithmic map and the geodesic distance are tractable. SO(3) (the special orthogonal group) is such a manifold of great practical interest. SO(3) has dimension 3. Elements of SO(3) are characterized by an axis of rotation in $\mathbb{R}^3$ and an angle of rotation.

In this appendix, we give a full toolbox of formulas needed to apply the techniques described in this document to SO(3). The material in this appendix has been tested and works very well. Practical applications include computer graphics for example, where one could require a camera to show some 3D scene under different rotations at predetermined times, with smooth transitions between rotations. Typically, one would apply our algorithms with zero λ and low μ to achieve smooth interpolation. The proposed approach yields considerably smoother looking transitions than could be obtained with piecewise geodesic interpolation. Another area of potential applications could be related to control of mechanical systems. Our reference for this appendix is [Bak00], an introduction to matrix Lie groups.

The linear group is defined as

$$\mathrm{GL}(3) = \{A \in \mathbb{R}^{3\times 3} \,:\, \det(A) \neq 0\}.$$

A particular subgroup of it is the special linear group:

$$\mathrm{SL}(3) = \{A \in \mathrm{GL}(3) \,:\, \det(A) = 1\}.$$

Considering only the orthogonal matrices in SL(3) yields the special orthogonal group:

$$\mathrm{SO}(3) = \{A \in \mathrm{SL}(3) \,:\, A^T A = I\}.$$

SO(3) is a Lie group, i.e., it is a group and a manifold. Applying theorem 2.2.2 gives a simple expression for the tangent spaces, see table C.1. The tangent space at the origin,

$$\mathrm{so}(3) = T_I \,\mathrm{SO}(3) = \{H \in \mathbb{R}^{3\times 3} \;:\; H + H^T = 0\}$$

is called the Lie Algebra of SO(3) and corresponds to the set of skew-symmetric matrices. Note that $T_A \,\mathrm{SO}(3) = A\,\mathrm{so}(3) = \{H = A\tilde{H} \,:\, \tilde{H} \in \mathrm{so}(3)\}$. Endowing so(3) with the inner product from GL(3) gives a natural metric on SO(3):

$$\langle H_1, H_2\rangle_A = \left\langle A^T H_1, A^T H_2\right\rangle_I = \mathrm{trace}\left((A^T H_1)^T A^T H_2\right) = \mathrm{trace}\left(H_1^T H_2\right) = \langle H_1, H_2\rangle .$$

Hence, SO(3) is a submanifold of GL(3). It can be proven that, at the identity matrix I, the Exp and Log maps reduce to the matrix exponential and logarithm:

$$\mathrm{Exp}_I : \mathrm{so}(3) \to \mathrm{SO}(3) : H \mapsto \mathrm{Exp}_I(H) = \exp(H),$$
$$\mathrm{Log}_I : \mathrm{SO}(3) \to \mathrm{so}(3) : A \mapsto \mathrm{Log}_I(A) = \log(A).$$

Set:	$\mathrm{SO}(3) = \{A \in \mathbb{R}^{3\times 3} \ : \ A^T A = I \text{ and } \det(A) = 1\}$
Tangent spaces:	$T_A \mathrm{SO}(3) = \{H \in \mathbb{R}^{3\times 3} \ : \ A^T H + H^T A = 0\}$
Inner product:	$\langle H_1, H_2 \rangle_A = \mathrm{trace}\left(H_1^T H_2\right)$
Vector norm:	$\|H\|_A = \sqrt{\langle H, H \rangle_A}$
Distance:	$\mathrm{dist}\,(A, B) = \|\mathrm{Log}_A\,(B)\| = \|\log(A^T B)\|$
Exponential:	$\mathrm{Exp}_A\,(H) = A\,\mathrm{Exp}_I\left(A^T H\right) = A\exp(A^T H)$
Logarithm:	$\mathrm{Log}_A\,(B) = A\log(A^T B)$
Mean:	$\mathrm{mean}\,(A, B) = A\exp(.5\log(A^T B))$

Table C.1: *Toolbox for the special orthogonal group,* SO(3).

From there, we can define the Exp and Log maps everywhere on SO(3), see table C.1.

We introduce the two next functions like we did in section 5.3 for positive matrices:

$$f_A : \mathrm{SO}(3) \to \mathbb{R}^+ : B \mapsto f_A(B) = \frac{1}{2}\,\mathrm{dist}^2\,(A, B)\,,$$
$$g : (\mathrm{SO}(3))^3 \to \mathbb{R}^+ : (A, B, C) \mapsto g(A, B, C) = \left\langle A\log(A^T B), A\log(A^T C)\right\rangle.$$

Similar algebra to what was done in section 5.3 and appendix B yields formulas for the gradients. We used remark 5.3.1 and the fact that orthogonal matrices are diagonalizable.

$$\mathrm{grad}\, f_A(B) = B\log(A^T B) = -\,\mathrm{Log}_B\,(A) \in T_B\,\mathrm{SO}(3)$$

And:

$$\begin{aligned}
A^T B &= U_B D_B U_B^{-1} \quad \text{(diagonalize)}\\
A^T C &= U_C D_C U_C^{-1} \quad \text{(diagonalize)}\\
\tilde{F}_B &= \text{first divided differences of log for } \lambda_i = (D_B)_{ii}\\
\tilde{F}_C &= \text{first divided differences of log for } \lambda_i = (D_C)_{ii}\\
\mathrm{grad}\,(A \mapsto g(A, B, C))\,(A) &= BU_B\left(\tilde{F}_B \odot \left(U_B^{-1}\log(C^T A)U_B\right)\right)U_B^{-1}\\
&\quad + CU_C\left(\tilde{F}_C \odot \left(U_C^{-1}\log(B^T A)U_C\right)\right)U_C^{-1} \in T_A\,\mathrm{SO}(3)\\
\mathrm{grad}\,(B \mapsto g(A, B, C))\,(B) &= AU_B^{-T}\left(\tilde{F}_B \odot \left(U_B^T\log(A^T C)U_B^{-T}\right)\right)U_B^T \in T_B\,\mathrm{SO}(3)\\
\mathrm{grad}\,(C \mapsto g(A, B, C))\,(C) &= AU_C^{-T}\left(\tilde{F}_C \odot \left(U_C^T\log(A^T B)U_C^{-T}\right)\right)U_C^T \in T_C\,\mathrm{SO}(3)
\end{aligned}$$

This is sufficient to compute the gradient of the objective function (3.6) applied to $\Gamma = \mathrm{SO}(3) \times \ldots \times \mathrm{SO}(3)$. The descent algorithms from chapter 3 apply directly, with the following vector transport on SO(3) with a component wise extension to Γ:

$$\mathsf{T}_\xi\,(\eta) = BA^T\eta, \qquad \text{with } \xi, \eta \in T_A\,\mathrm{SO}(3) \text{ and } \xi = \mathrm{Log}_A\,(B)\,.$$

This vector transport preserves inner products (it is an isometric transport), i.e.,

$$\langle \eta_1, \eta_2 \rangle_A = \langle \mathsf{T}_\xi\,(\eta_1)\,, \mathsf{T}_\xi\,(\eta_2)\rangle_B\,.$$

Bibliography

[AFPA08] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):328, 2008.

[Alt00] C. Altafini. The de Casteljau algorithm on $SE(3)$. In *Book chapter, Nonlinear control in the Year 2000*, pages 23–34, 2000.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[Bak00] Andrew Baker. An introduction to matrix groups and their applications. `http://www.maths.gla.ac.uk/~ajb/dvi-ps/lie-bern.pdf`, 2000.

[Bha07] R. Bhatia. *Positive definite matrices*. Princeton University Press, 2007.

[Boo86] W.M. Boothby. *An introduction to differentiable manifolds and Riemannian geometry*. Academic Press Inc, 1986.

[CKS99] P. Crouch, G. Kun, and F. Silva Leite. The de Casteljau algorithm on the Lie group and spheres. In *Journal of Dynamical and Control Systems*, volume 5, pages 397–429, 1999.

[CS91] P. Crouch and F. Silva Leite. Geometry and the dynamic interpolation problem. In *Proc. Am. Control Conf., Boston, 26–29 July, 1991*, pages 1131–1136, 1991.

[CS95] P. Crouch and F. Silva Leite. The dynamic interpolation problem: on Riemannian manifolds, Lie groups, and symmetric spaces. *J. Dynam. Control Systems*, 1(2):177–202, 1995.

[CSC95] M. Camarinha, F. Silva Leite, and P. Crouch. Splines of class C^k on non-Euclidean spaces. *IMA J. Math. Control Inform.*, 12(4):399–410, 1995.

[dC92] Manfredo Perdigão do Carmo. *Riemannian geometry*. Mathematics: Theory & Applications. Birkhäuser Boston Inc., Boston, MA, 1992. Translated from the second Portuguese edition by Francis Flaherty.

[FPAA07] P. Fillard, X. Pennec, V. Arsigny, and N. Ayache. Clinical DT-MRI estimation, smoothing, and fiber tracking with log-Euclidean metrics. *IEEE Transactions on Medical Imaging*, 26(11):1472–1482, 2007.

[GB10] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. `http://cvxr.com/cvx`, May 2010.

[Hai09] Luc Haine. *Eléments de géométrie différentielle (MAT2110 course notes)*. Mathematics Department, Université catholique de Louvain (UCL), 2009.

[Hig08] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[HS07] K. Hüper and F. Silva Leite. On the geometry of rolling and interpolation curves on S^n, SO_n, and Grassmann manifolds. *J. Dyn. Control Syst.*, 13(4):467–502, 2007.

[JSR06] Janusz Jakubiak, Fátima Silva Leite, and Rui C. Rodrigues. A two-step algorithm of smooth spline generation on Riemannian manifolds. *J. Comput. Appl. Math.*, 194(2):177–191, 2006.

[KDL07] Alfred Kume, Ian L. Dryden, and Huiling Le. Shape-space smoothing splines for planar landmark data. *Biometrika*, 94(3):513–528, 2007.

[LÖ4] J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[MS06] Luís Machado and F. Silva Leite. Fitting smooth paths on Riemannian manifolds. *Int. J. Appl. Math. Stat.*, 4(J06):25–53, 2006.

[MSH06] Luís Machado, F. Silva Leite, and Knut Hüper. Riemannian means as solutions of variational problems. *LMS J. Comput. Math.*, 9:86–103 (electronic), 2006.

[NHP89] Lyle Noakes, Greg Heinzinger, and Brad Paden. Cubic splines on curved spaces. *IMA J. Math. Control Inform.*, 6(4):465–473, 1989.

[NW99] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer Verlag, 1999.

[PN07] Tomasz Popiel and Lyle Noakes. Bézier curves and C^2 interpolation in Riemannian manifolds. *J. Approx. Theory*, 148(2):111–127, 2007.

[SASK09] Chafik Samir, P.-A. Absil, Anuj Srivastava, and Eric Klassen. A gradient-descent method for curve fitting on riemannian manifolds. *Technical Report UCL-INMA-2009.023 (submitted), Universite catholique de Louvain*, 2009.

[Stu98] Jos F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones, 1998.

[TTT99] K. C. Toh, M.J. Todd, and R. H. Tutuncu. Sdpt3 - a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[VD08] Paul Van Dooren. *Algorithmique numérique (INMA2710 course notes)*. Ecole Polytechnique de Louvain, Université catholique de Louvain (UCL), 2008.

[Wei10] Eric W. Weisstein. Complete metric space. From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/CompleteMetricSpace.html`, 2010.