



Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Interpolation and fitting on Riemannian manifolds

Pierre-Yves Gousenbourger

Thesis submitted in partial fulfillment
of the requirements for the degree of
Docteur en sciences de l'ingénieur

Dissertation committee:

Prof. Pierre-Antoine Absil (UCLouvain, Belgium) – advisor
Prof. Laurent Jacques (UCLouvain, Belgium) – advisor
Prof. Gabriel Peyré (École normale supérieure, Paris, France)
Prof. Jean-Pierre Raskin (UCLouvain, Belgium) – chair
Prof. Jean-François Remacle (UCLouvain, Belgium)
Prof. Benedikt Wirth (Universität Münster, Germany)

Version of September 15, 2020.

All unpublished material in this document is © 2020 of the author, all rights reserved.

To my Mom, to my Dad
and to my wonderful wife.

“Sur les nombres réels repose l’analyse.”

– Beni citant Michel Willem

“Mathématiques et poésie ont beaucoup en
commun : (...) si c’est beau, simple,
frappant, c’est que vous êtes
probablement sur la bonne voie.”

– Cédric Villani

“*Omnis scientia requirit mathematicam:*
toute science requiert les mathématiques.”

– Roger Bacon [1214-1294]

Abstract

THE access to constantly increasing computational capacities has revolutionized the way engineering is seen. We are now able to produce a large quantity of data, thanks to cheap sensors. However, processing such data remains costly both in computational time and in energy. One of the reasons is that the structure of the data is often omitted or unknown. The search space becomes so large that finding the solution to simple problems often turns out to be finding a needle in a haystack.

A classical problem in data processing is called the “fitting problem”. It consists in fitting a d -dimensional curve to a set of data points associated to d parameters. The curve must pass sufficiently close to the data points while being regular enough. When the underlying structure of the data points (*i.e.*, the manifold) is known, one can impose to the curve to preserve this structure (*i.e.*, to remain on the manifold), such that the search space is drastically reduced.

The goal of this thesis is to develop methods to (approximately) solve this fitting problem; the bet is to require “less” (less computational capabilities, power, storage, time) by leveraging “more” knowledge on the search space. The objective is the following: provide a toolbox that produces a differentiable fitting curve to data points on manifolds, based on very few and simple geometric tools, at low computational cost and storage capacity, all this while maintaining an acceptable quality of the solutions.

The algorithms are applied to different illustrative problems. In 3D shape reconstruction, the data points are organs contours acquired via MRI, and the parameter is the acquisition depth; in wind fields estimation, the data points belong to the manifold of positive semi-definite matrices of given rank, and the parameters are the prevalent wind amplitudes and angles. We also show the performances of our algorithms in applications for parametric model order reduction.

Acknowledgments

As Henry Ford said: *“Coming together is a beginning; keeping together is progress; working together is success.”* With this little note, I would like to share the success of this thesis with the people who made it happen by closely collaborating, or just by being a helpful hand. My sincere thanks go:

- to Prof. Pierre-Antoine Absil and Prof. Laurent Jacques, my advisors, for their wise advice, their patience and their strong support in all the steps of my thesis; a special tribute should be given to Pierre-Antoine for his stunning skills in explaining hard concepts in a very clear manner; similarly, I deeply thank Laurent for his humanity and his unconditional support in the two hard moments I had to go through;
- to Prof. Benedikt Wirth, from Universität Münster who is not only an amazing researcher with impressive mathematical knowledge, but who is also a kind and human host: my three visits to Münster have been fruitful, challenging and exciting; they all paved the way to publications or seeded ideas that made my thesis grow;
- to Dr. Estelle Massart, my closest collaborator at UCLouvain, for the numerous and stimulating chats we had in her office (which was also a gossip hub, by the way), the productive work we did together and her open-mindedness; her human qualities and her deep mathematical knowledge make her a very valuable colleague;
- to Dr. Ronny Bergmann, from Technische Universität Chemnitz, for his hospitality in both Kaiserslautern and Chemnitz, as well as his trust at the beginning of my PhD. I’ll remember and cherish two legendary

★ | Acknowledgments

discussions we had: the kick-off of our joint work in Louvain-la-Neuve on the board of the cafeteria, and the other one during the GSI in Paris, where we finally set the notations for our paper. That was hard work, but we made it;

to Prof. Jean-Pierre Raskin for his presidency in my jury, but also for the freedom and the trust he left me while rethinking the way we handle internships; thanks also to Prof. Jean-François Remacle and Prof. Gabriel Peyré for their precious comments on my manuscript;

to Dr. François Rottenberg for being an amazing office mate since the very beginning, for his (very peculiar) sense of humor, for his support and his numerous pieces of advice, not only about research;

to the *Dolphins* (Dr. Amir, Anne-Sophie, Antoine, Dr. Cédric, Damien, Dr. François, Gabriel, Dr. Kévin, Maxime, Niels, Simon, Stéphanie, Thomas, Dr. Valerio, Victor, Vincent... and Beni), for sharing moments of joy and sadness, supporting each other but also providing the most important thing during breaks: taking our minds off things!

to Beni, specifically, for his theories, his smile, his happiness, his energy (positive or negative), his quotes, his “oh! I just read an article about [insert here a random topic]”... and who never got fired, after all;

to Anne, Astrid, Benoit, Brigitte, David, Esther, Isabelle, François, Jean, Ludzzie, Marie, Marie-Hélène, Nathalie, Pascale, Patricia, Souley and Viviane, our incredible administrative and technical staff, for their patience, their smile and their availability; special thanks to Marie for knowing everything I needed to know when I needed to know it, and to Isabelle for the beard;

to the other members of ICTEAM and of EPL, who gave me the possibility to do “something else” than “just some research”, who trusted me when there was an opportunity to change something, to rise ACSEP from the ashes, or just to give an idea;

to Prof. Michel Verleysen, our Dean, for his trust, his resilience and his constant willingness to listen to everyone’s word;

to Myriam Banai and Delphine Ducarme for their smile, their energy in setting new pedagogical frameworks, their open-mindedness, their

caring ear, their constant support and... the way they have to let people feel that every single person is special;

to my students (all of them) who gave me the privilege to know what it feels to serve a real cause and who made me a better assistant, hence a better researcher;

to my wife, for her support “no matter what”, her love, her courage when I lost mine, and for never losing faith in me;

and of course, to my Mom, my Dad, my brother, my sister, and my friends who were there to share the brightest times but also the darkest moments... or just because they sometimes politely let me explain what I did, without understanding a single word of the problem ;-). For this resilience and because we all need such friends and family, you have my infinite gratitude.

This thesis is dedicated to my Mom and Dad.

This work, the related travels and conference expenses were mostly supported by (i) the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160, (ii) “Communauté française de Belgique - Actions de Recherche Concertées” (contract ARC 14/19-060), and of course (iii) UCLouvain.

Contents

1	Introduction	1
1.1	What is the current research about manifolds?	2
1.2	What are the motivations of data fitting?	3
1.3	How to tackle the data fitting problem?	4
1.4	Why is this work different?	7
1.5	How to read (and what to find in) this thesis?	10
2	Concerning manifolds	15
2.1	A (not too) short introduction for the impatient	16
2.2	From a chart to a manifold	20
2.3	A Euclidean space on manifolds: the tangent space	26
2.4	Riemannian structure of a manifold	30
2.5	Remarkable mappings on Riemannian manifolds	33
2.6	Additional numerical tools	41
3	Interpolation with Bézier curves	47
3.1	Euclidean Bézier curves	48
3.2	Bézier curves on manifolds	52
3.3	Interpolation with velocity-imposed curves	56
3.4	Interpolation with composite Bézier curves	61
3.5	An application to transvaginal ultrasound	67
4	Fitting with Bézier, blended, and Bézier-like curves	77
4.1	Fitting with composite Bézier curves	79
4.2	Fitting with blended Bézier curves	90
4.3	Fitting with composite Bézier-like curves	98
4.4	Numerical examples	107

4.5	An application to wind fields estimation	112
4.6	Another application to parametric model order reduction . . .	120
5	Optimality of the Bézier fitting curve	127
5.1	Some additional mathematical elements	128
5.2	Gradient of the discretized mean squared acceleration	130
5.3	Numerical considerations	141
5.4	Validation of the fitting methods	146
6	Interpolation with Bézier surfaces	157
6.1	Euclidean Bézier surfaces	158
6.2	Bézier surfaces on manifolds	161
6.3	Composite Bézier surfaces on manifolds	165
6.4	Control points generation for surface interpolation	178
6.5	Accelerated generation of control points	187
6.6	Numerical examples	191
7	Fitting with blended surfaces	199
7.1	Euclidean thin plate splines	200
7.2	Fitting with blended thin plate splines	203
7.3	Illustrative examples	217
8	Summary and perspectives	223
8.1	What was it about?	223
8.2	How well were the objectives achieved?	225
8.3	What to do next?	229
	Appendices	233
A	Coefficients for interpolation with a composite cubic curve . .	233
B	Coefficients for interpolation with a hybrid composite curve . .	235
C	Coefficients for fitting with a composite cubic curve	236
D	Proof of equation (6.33)	237
E	Examples of geometric elements on manifolds	238
	List of publications	239
	Bibliography	243
	List of symbols	261
	Index	265

1

Introduction

ALWAYS MORE. This is the current fashion of our society. It began after the Second World War, when the belongings became a new sign of wealth [Des06]. Sometimes, more makes sense: more rights, more justice, more equity, more respect, more freedom,... Sometimes, it is more questionable [MMRB72]: more money, more assets, more power, more broadcasting, more Netflix, more connectivity,...

The sciences, and more specifically the computer sciences, make no exception to this rule. With the discovery of the new computational capabilities of computers, some new “mores” emerged: more data, more computational power, more complexity. This has permitted tremendous discoveries and revolutionized the way engineering was thought. Machine learning, and now deep learning, permits to mimic neural networks allowing to understand and solve problems we wouldn’t have dreamed of several years ago; finite elements were made more popular. Furthermore, the amelioration of CPUs provided the necessary devices for those methods. It also came with the drawback that this machinery requires a high computational cost and, as a consequence, energy; it cannot be integrated in small devices where the computational capacity is low.

Let us take a quick example of a complicated optimization problem $\min_{x \in S} f(x)$, constrained with many rules $C(x)$, driven by a highly non-linear objective function f on a large space S . A brute-force approach to find the minimum value of f would be to generate a solution for a lot

of values of $x \in S$, check if they satisfy $C(x)$, and then choose among them the best solution. With enough time and energy, this approach might work. However, if it was possible to search only on the space $S_C := \{x \in S \text{ such that } C(x)\}$, one could drastically reduce the time of search.

In this thesis, we are interested in such a nonlinear optimization problem. We consider the problem of fitting a (d -dimensional) curve to a set of data points on a Riemannian manifold (see Chapter 2 for an introduction to manifolds). More formally, the problem is expressed as follows.

Given data points d_i on a Riemannian manifold \mathcal{M} , $i \in \mathbb{N}^d$, associated with time instants $t_i \in \mathbb{R}^d$, we seek a d -dimensional curve $\gamma: \mathbb{R}^d \rightarrow \mathcal{M}$ that strikes a balance between the conflicting goals of being “sufficiently straight” while passing “sufficiently close” to the data points at the given times.

The goal of this work (that builds on previous theses at UCLouvain [Ren13, Bou14, Mas19]) is to develop methods to (approximately) solve this fitting problem; the bet with those methods is to require “less” (less computational capabilities, power, storage, time) by using “more” knowledge on the search space. The objective will be the following: provide a toolbox that requires as little knowledge as possible, as little computational cost, and as little storage capacity, but meanwhile furnishes acceptable solutions.

1.1 What is the current research about manifolds?

During recent years, it has become more and more common and important to process data from non-Euclidean spaces, in particular from Riemannian manifolds. To cite a few applications, such data appear in computer vision [Fle13] (shape space), to interpret colors in images [BLSW14] (sphere), or to represent fixed rank matrices as submanifold of all matrices [Van13].

Among all applications, optimization on manifolds has gained a lot of interest this last decade. The textbook of Absil *et al.* [AMS08] is probably a key element in this popularization. It summarizes several optimization methods on matrix manifolds; it also served as basis for the development of toolboxes (Manopt [BMAS14] in 2014, and MVIRT [Ber17] in

2017) providing easy access to such optimization methods. Manopt, in particular, received positive return in many different topics of research, with applications in low-rank modelling in image analysis [ZYZY15], dimensionality reduction [CG15], phase retrieval [SQW17] or even 5G-like MIMO systems [YSZL16]. Research is still conducted in this field, enriching the toolbox month after month (for instance with proximal gradient method [HW19]). The toolbox is currently translated to Python [TKW16] and to Julia (manopt.jl). MVIRT stems from recent interest in manifold-valued image and data processing: total variation regularization of phase-valued data [SC11, SC13] or manifold-valued data [LSKC13, WDS14, HW20], second order methods [BLSW14, BBSW16], and total generalized variation algorithms for manifold-valued data [BFPS18, BHSW18] are some of the applications of this toolbox. See also [BLPS19] for a review of the recent literature in the field of manifold-valued image processing.

Very recently, a rich variety of research has been conducted in this direction, like image morphing with time-discrete geodesics [EKPR19], manifold regression to predict MEG/EEG brain signals without source modeling [SAV⁺19], dictionary learning [HHS⁺15], Riemannian optimization tools [LB19, MKJS19], or even human behavior understanding [KDB⁺20].

Among all those applications, one in particular will be treated in this manuscript: manifold-valued data fitting.

1.2 What are the motivations of data fitting?

The data fitting problem is motivated by various applications that require to denoise or resample parameter-dependent data on a Riemannian manifold \mathcal{M} .

For example, a crucial task in computational anatomy is to denoise and resample the evolution of the shape of an organ, yielding a curve fitting problem ($d = 1$) on a shape manifold. In Arnould *et al.* [AGS⁺15], for instance, different images of an organ are acquired in the space of closed 2D-shapes \mathcal{S} at different body-depths, using MRI. The final goal is to reconstruct the 3D volume $\gamma_{\text{MRI}}(z): \mathbb{R} \rightarrow \mathcal{S}$ of the imaged organ, where z is the depth inside the human body of each MRI slice. In the same work, orientations of a probe are also registered at different times by 2D transvaginal ultrasound (TVUS) [BRP12]. Hence, another curve can be fitted to recover the probe navigation path $\gamma_{\text{TVUS}}(t): \mathbb{R} \rightarrow \text{SE}(3)$. Here, the manifold \mathcal{M} is

the set of rigid-body motions in \mathbb{R}^3 .

Rigid motion has actually many other applications, like interpolation or fitting of rotations of 3D objects lying on the special orthogonal group $\mathcal{M} = \text{SO}(3)$. This problem arises in robotics for motion planning of rigid bodies or trajectory optimization [WLS⁺20], or in computer graphics, to track 3D objects [Par10]. In Cosserat rods [San10], we have $d = 1$ and $\mathcal{M} = \text{SE}(3)$, while Cosserat shells [SNB16] require $d = 2$ and $\mathcal{M} = \text{SE}(3)$.

Curve fitting is also useful in projection-based model reduction of dynamical systems depending on parameters. In Pyta *et al.* [PA16], the dynamical system is the Navier-Stokes equations and depends only on the Reynolds parameter ($d = 1$). The reduced model is obtained by computing several suitable projectors. Those projectors are elements of the Grassmann manifold $\mathcal{M} = \text{Gr}(n, r)$. Finding a projector is however a time- and memory-consuming task. Based on projectors pre-computed offline for a small set of parameter values, fitting is used to approximate the projector associated with a new parameter value. A new algorithm was recently proposed by Mosquera *et al.* [MHF19] for PMOR.

A similar approach is used in the wind field modeling problem (see Section 4.5 for details). Here, a wind field is characterized by a mean field and a covariance matrix C belonging to the manifold $\mathcal{M} = \mathcal{S}_+(n, r)$, the set of positive semidefinite matrices of size n and rank r [MHA19].

In diffusion tensor imaging, a diffusion tensor is an element of the manifold $\mathbb{P}(3)$ of 3×3 symmetric positive definite matrices. Each tensor is acquired for each voxel of a volume of interest (thus $d = 3$), and interpolation can be used to infer more finely sampled data [PFA06].

Let us finally mention fitting of sphere-valued data ($\mathcal{M} = \mathbb{S}^{n-1}$) used in many applications of data analysis, for storm tracking and prediction, or the study of bird migration [SKKS14]; and also liquid crystals, which can be described by a function from \mathbb{R}^3 into the projective space $\mathbb{RP}(2)$ [Muc12].

1.3 How to tackle the data fitting problem?

The problem of fitting a d -dimensional curve to data points (with or without the two above-mentioned conflicting goals) has been widely considered in the literature for two specific cases.

The first one is when the manifold \mathcal{M} is a Euclidean space. In this case, a preferred way to handle fitting problems is to resort to splines, *i.e.*, to

piecewise polynomials. Those functions are convenient to manipulate and evaluate, while the interpolation error can be kept small. As the pieces are usually of low degree, such splines avoid the Runge’s phenomenon that plagues high-degree polynomial interpolation.

The piecewise-polynomial approach also permits a large range of variations of the fitting problem, depending on the desired degree of smoothness, the admissible classes or type of polynomial pieces (like Bézier forms, for instance), optimality criteria, etc. The book of Farin [Far02] is a convenient point of entry to this vast literature.

Another approach are methods called *subdivision schemes*, quite popular in some communities of computer-aided geometric design. These schemes allow to efficiently draw curves and surfaces in Euclidean spaces: a discrete set of points is recursively refined, resulting in a limit curve or surface. Depending on the chosen refinement scheme, the curve may interpolate or just approximate the initial points. For instance, Dyn *et al.* [DLG87] proposed a four-point scheme whose limit case leads to C^1 -interpolatory curves; she extended it to a C^2 -(nearly interpolatory) scheme in [DFH05]. Deng and Ma considered more general polynomial-reproducing schemes [DM16]. The monograph of Peters and Reif [PR08] gives a larger introduction into the topic.

The second case, more recent, is the case of manifold-valued fitting for $d = 1$, *i.e.*, curve fitting. Two schools coexist in the literature: the one that relies on optimization of splines, and the one that relies more on constructive algorithms, like subdivision schemes generalized to non-Euclidean spaces. Such schemes and their convergence and smoothness analysis can be found, *e.g.*, in [Dyn09, Gro08, NYY11, WD05, WP06, WNYG07, Wei10, Wei12]. In complicated spaces, however, the many necessary recursions to evaluate a subdivision curve may become computationally prohibitive. Also, subdivision schemes are typically local so that for instance curve interpolation with minimal global curvature cannot be achieved. Hence, this approach comes in contradiction with at least two goals of this work (see Section 1.4): being computationally tractable and constructing a “straight enough” curve.

The spline-based approach permits to better handle the two conflicting goals mentioned in the beginning of this chapter (*i.e.*, data fitting and smoothness). It consists in encapsulating them in an optimization problem

$$\min_{\gamma \in \Gamma} E_\lambda(\gamma) := \min_{\gamma \in \Gamma} \int_{t_0}^{t_n} \left\| \frac{D^2 \gamma(t)}{dt^2} \right\|_{\gamma(t)}^2 dt + \lambda \sum_{i=0}^n d^2(\gamma(t_i), d_i), \quad (1.1)$$

where the set Γ is an admissible set of curves on \mathcal{M} , $\frac{D^2}{dt^2}$ denotes the (Levi-Civita) second covariant derivative, $\|\cdot\|_{\gamma(t)}$ is the Riemannian metric at $\gamma(t)$, and $d(\cdot, \cdot)$ is the Riemannian distance. The parameter λ (controlled by the user or by the algorithm itself, *e.g.*, through a cross-validation procedure) sets the balance between the *regularizer* term $\int_{t_0}^{t_n} \|\frac{D^2\gamma(t)}{dt^2}\|_{\gamma(t)}^2 dt$ (that permits to the curve to be “sufficiently straight”) and the *goodness of fit* term $\sum_{i=0}^n d^2(\gamma(t_i), d_i)$ (that permits to the curve to pass “sufficiently close” to the data).

This approach is convenient because it generalizes well from the Euclidean space. Indeed, when the Riemannian manifold \mathcal{M} reduces to a Euclidean space and Γ is chosen to be the Sobolev space $H^2(t_0, t_n)$, a classical result (see, *e.g.*, [GS93, Theorem 2.4]) states that the solution of (1.1) is a natural cubic spline. Specifically, it is the interpolating natural cubic spline when $\lambda \rightarrow \infty$ (see, *e.g.*, [SM03] for its definition) and the least-squares linear regression as $\lambda \rightarrow 0$ (see [MS06, Proposition 4.5 and 4.6] for a result on manifolds).

Several methods exist to solve problem (1.1). In Samir *et al.* [SASK12], the problem is addressed nearly as is. They consider an infinite dimensional Sobolev space equipped with the Palais-metric, and (1.1) is minimized with a steepest-descent approach; that approach was applied to image processing by Su *et al.* [SDK⁺12].

However, problem (1.1) is often too complicated to be tackled in its full generality, so most authors make some approximations or restrictions to transform it to a simpler problem. In Boumal *et al.* [BA11], the curve is discretized with K points, reducing the search space to a product manifold \mathcal{M}^K , and the covariant derivative is replaced by manifold-valued finite differences. The algorithm is then applied to $SO(3)$. Machado and Monteiro [MM17] handle the specific case of the sphere. Kim *et al.* [KDL18] or Hüper *et al.* [HS07] generalized to several spaces a technique from Jupp and Kent [JK87] called “unwrapping and unrolling”; this technique has the advantage to “transform” the manifold-valued problem to an equivalent Euclidean problem, and solves it with classical Euclidean techniques.

The limit case where $\lambda = 0$, *i.e.*, geodesic regression, was studied in the work of Rentmeesters [Ren11] and Fletcher [Fle13]. In the former, the problem is solved with a gradient descent technique for Riemannian symmetric spaces, while in the latter, the least-squares problem is studied intrinsically as a minimization of the sum of squared geodesic distances on \mathcal{M} . An extension of geodesic regression is polynomial regression, for which Riemannian

nian techniques were proposed by Hinkle *et al.* [HFJ14] (intrinsic method) and Lin *et al.* [LSZD17] (extrinsic methods).

The other limit case where $\lambda \rightarrow \infty$, however, was poorly documented before the beginning of this work. In the last two years, however, it has gained some interest in the field. We mention the work of Bogfjellmo *et al.* [BMV18] on interpolation with C^2 composite cubic Bézier curves on Riemannian symmetric spaces, or the very recent Hermite’s algorithm developed by Zimmermann [Zim19], as well as the Neville-Aitken approach of Mosquera *et al.* [MHFF19]. Recently, Efland applied techniques using Bézier curves to the space of images [ERS⁺15].

Besides these two special—but important—cases (the Euclidean case, and the manifold-valued case for $d = 1$), multivariate manifold-valued fitting does not appear to have been much researched. Some work can nevertheless be found. Steinke *et al.* [SHPS08] use a technique based on thin plate splines to produce an interpolation map between two Riemannian manifolds. The approach is generalized in [SHS10] where, given a set of training pairs (X_i, Y_i) with the X_i ’s and Y_i ’s on two manifolds, a mapping is sought between these two manifolds. The mapping has to minimize a regularized empirical risk. We also mention a related technique for volumetric registration presented in [JSTL07].

To be fair, the data fitting problem can also be approached in other different ways, but they depart strongly from the work presented in this document. Some examples can be found in [ZN19, Shi08], in the work of Machado [MSLK10, MSM18] or Sander [San16].

1.4 Why is this work different?

In this work, we tackle the fitting problem on manifolds for curves ($d = 1$) and surfaces ($d = 2$). The search space Γ is chosen to be a space of C^1 composite Bézier curves (Chapters 3, 4 and 5), Bézier surfaces (Chapter 6), or thin plate splines (Chapter 7). The optimality of the returned curves (resp. surfaces) is guaranteed only when \mathcal{M} is Euclidean (on manifolds, there is no known closed form for optimal Bézier curves or thin plate splines, so one would have to approximate them).

Euclidean Bézier curves have been extensively used in the CAGD (Computer Aided Geometric Design) community in the past (see Farin [Far02] for a comprehensive textbook) to model smooth curves and surfaces for

real- and vector-valued data points. They can also be used to model smooth curves and surfaces on manifolds [PN07] (see Figure 1.1 for a Bézier surface computed on the Riemannian space of shells with the methods proposed in Chapter 6).



Fig. 1.1 Differentiable composite Bézier surface on the Riemannian space of shells (Section 6.6.4) interpolating the red shapes. The gray shapes are points on the Bézier surface driven by the control points in green. Their location indicates where in the \mathbb{R}^2 domain they are achieved. Dataset courtesy to Yeh *et al.* [YLSL11].

The advantages to work with Bézier functions are numerous. First, compared to [SASK12, BA11], the search space is drastically reduced to the so-called control points of the Bézier functions. Second, it is very simple to impose differentiability conditions for the fitting curves or surfaces (this is appreciated in several applications mentioned in Section 1.2). Finally, an advantage compared to [SASK12, KDL18, Fle13, Ren11] consists in the simplicity of the methods that can be produced in that framework. Indeed, only two objects on the manifold are required: the Riemannian exponential map and the Riemannian logarithm, while most of the techniques mentioned in Section 1.3 require a gradient or heavy computations for parallel transportation.

The present work performs fitting with the following philosophy. Instead of solving the highly nonlinear problem (1.1) on the manifold, one chooses to follow a suboptimal route. As the manifold \mathcal{M} can be locally

approximated by a (linear) tangent space $T_x\mathcal{M}$ around a point x , we solve instead a local approximation of (1.1) on a set of such spaces. The solution obtained there is then either generalized to \mathcal{M} by interpreting the closed form as exponentials and logarithms, either by blending together solutions obtained in various tangent spaces.

With this approach, four cases are studied: interpolation and fitting, respectively with $d = 1$ and $d = 2$. Higher dimensions are not considered in this thesis but a generalization to $d > 2$ does not seem to be too cumbersome.

Among the several techniques developed here, the goal is to combine most of the following desirable properties:

- (i) As $\lambda \rightarrow \infty$, the data points are interpolated at the given times;
- (ii) The curve (resp. surface) is of class C^1 ;
- (iii) If the manifold \mathcal{M} reduces to a Euclidean space, then the produced curve minimizes (1.1), *i.e.*, it is the natural smoothing spline ($d = 1$) or the fitting thin plate spline ($d = 2$);
- (iv) The only knowledge that the methods require from the manifold is the Riemannian exponential and the Riemannian logarithm;
- (v) The produced curve (resp. surface) is represented by $\mathcal{O}(n)$ tangent vectors to the manifold (or simply points on the manifold), where n is the number of data points;
- (vi) Computing $\gamma(t)$ for any given t requires $\mathcal{O}(1)$ exp and log operations once the representation by $\mathcal{O}(n)$ tangent vectors is available.

Those properties cannot be verified without assuming that the fitting problem instances (*i.e.*, the data points and associated times) are such that the considered algorithms evaluate the Riemannian exponentials and logarithms only where they are well defined (so that the resulting curve is well defined as well) and C^1 (so that property (ii) holds). This standing assumption always holds when \mathcal{M} is a Hadamard manifold, and is in general not a concern when \mathcal{M} is complete and the cut loci on \mathcal{M} have codimension greater than one.

This work departs thus from the others by those six properties. The ambition is to propose techniques that are close to optimize (1.1) while being computationally tractable, and that compute differentiable, easy-to-store results.

A Hadamard manifold is a complete, simply connected manifold, with everywhere a non-positive sectional curvature.

1.5 How to read (and what to find in) this thesis?

There are eight chapters in this thesis, including this introduction and a concluding chapter. The six remaining ones are organized as follows: Chapter 2 introduces the principal elements of differential geometry used in this thesis. Chapters 3 to 7 constitute the core of the thesis and the main contributions: Chapters 3 to 5 concern the case $d = 1$ while Chapters 6 and 7 concern the case $d = 2$. Each time, interpolation and fitting are treated separately. Figure 1.2 gives a “reading route” of the thesis.

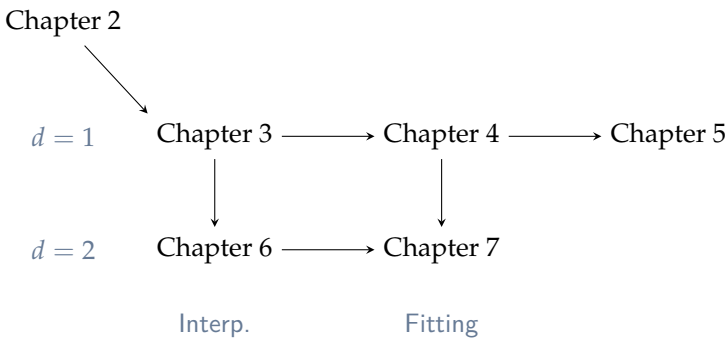


Fig. 1.2 Reading route of the thesis. $A \rightarrow B$ suggests to read A before B .

Ch. 2 An introduction to manifolds

This chapter introduces formally the elements of differential geometry used in the thesis (like the exponential map, the logarithm, the tangent space, etc.). As the mathematical details might sometimes be tedious to manipulate, a first section is provided to give an intuitive idea of the key objects. This first section should be sufficient to understand the vast majority of the thesis, but it might be useful to have a look at the next sections to fully appreciate the proofs provided in the document.

Ch. 3 Interpolation with Bézier curves

This chapter concerns interpolation of n manifold-valued data points associated to time-parameters, by means of piecewise-cubic Bézier splines. The

resulting composite cubic Bézier curve minimizes its mean squared acceleration when the manifold \mathcal{M} is a linear space. First, a reminder on Bézier curves on the Euclidean space and on manifolds is provided.

As first contribution, we propose a method to optimize the control points of the Bézier curve as well as a reconstruction algorithm in a way that is consistent with the properties given in Section 1.4, inspired from the methods published in [AGS⁺15]. The algorithm is afterwards applied to a medical problem where segmented MRI slices must be interpolated in order to reconstruct a 3D-volume of tumoral tissue. This comparison was published in [SGJ15] in collaboration with Dr. Chafik Samir (Université de Clermont, France).

The related publications are listed hereunder.

[AGS⁺15] Antoine Arnould, Pierre-Yves Gousenbourger, Chafik Samir, P.-A. Absil, and Michel Canis. Fitting smooth paths on Riemannian manifolds: Endometrial surface reconstruction and preoperative MRI-based navigation. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, volume 9389 of *Lecture Notes in Computer Sciences*, pages 491–498, Berlin, Heidelberg, 2015. Springer. doi:10.1007/978-3-319-25040-3_53

[SGJ15] Chafik Samir, Pierre-Yves Gousenbourger, and Shantanu H. Joshi. Cylindrical surface reconstruction by fitting paths on shape space. In H. Driira, S. Kurtek, and P. Turaga, editor, *Proceedings of the 1st International Workshop on DIFFerential Geometry in Computer Vision for Analysis of Shapes, Images and Trajectories (DIFF-CV 2015)*, pages 11.1–11.10. BMVA Press, 2015. doi:10.5244/C.29.DIFFCV.11

Ch. 4 Fitting with blended curves

After interpolation, the technique from Chapter 3 is extended to composite cubic *fitting* Bézier curves, to approach the complete problem (1.1). However, a naive generalization comes along with a problem of interpolation when $\lambda \rightarrow \infty$. This problem is related to the local injectivity radius of the manifold and to the generalization of differentiability conditions of Bézier curves. It justifies the creation of the *blended fitting curves*, presented in this chapter as main contribution. Those fitting curves combine the six properties of Section 1.4 and are subject to a proper publication [GMA18c]:

[GMA18c] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Data fitting on manifolds with composite Bézier-like curves and blended cubic splines. *Journal of Mathematical Imaging and Vision*, 61(5):645–671, 2018. doi:10.1007/s10851-018-0865-2

Moreover, the fitting methods (the naive generalization and the blending approach) are applied to two real-life applications, *i.e.*, the wind field problem (Section 4.5) and parametric model order reduction (PMOR, Section 4.6). Each application was the subject of publications in collaboration with Dr. Estelle Massart (UCLouvain, Belgium, and Oxford, UK), for both, and Dr. Nguyen Thanh Son (UCLouvain, Belgium) for PMOR:

[GMM⁺17] Pierre-Yves Gousenbourger, Estelle Massart, Antoni Musolas, P.-A. Absil, Laurent Jacques, Julien M Hendrickx, and Youssef Marzouk. Piecewise-Bézier C^1 smoothing on manifolds with application to wind field estimation. In *ESANN2017*, pages 305–310. Springer, 2017

[MGS⁺19] Estelle Massart, Pierre-Yves Gousenbourger, Nguyen Thanh Son, Tatjana Stykel, and P.-A. Absil. Interpolation on the manifold of fixed-rank positive-semidefinite matrices for parametric model order reduction: preliminary results. In *ESANN2019*, pages 281–286. Springer, 2019

Ch. 5 Analysis of the quality of Bézier fitting curves

In the two previous chapters, equation (1.1) is only verified when \mathcal{M} is a linear space. When \mathcal{M} is *not* a linear space, the resulting composite cubic Bézier curve is only a suboptimal solution of (1.1). In this chapter, we are interested in a numerical analysis of the quality of the curves computed in Chapters 3 and 4.

To do so, the regularizer of (1.1) is approximated by (squared) second order absolute differences introduced in [BBSW16], the search space is limited to the composite Bézier curves presented in Chapter 3, and (1.1) is solved via a Riemannian gradient descent.

The main contribution of this chapter is the derivation of the gradient of the objective function of (1.1) with respect to the control points of the Bézier spline. The gradient computation is based on the recursive structure of the De Casteljau algorithm [PN07]. Furthermore, we achieved to keep close to the six properties of Section 1.4, as the only required tools of the manifold are the exponential map, the logarithmic map, and a certain Jacobi field along geodesics.

This contribution results from a collaboration with Dr. Ronny Bergmann (Technische Universität Chemnitz, Germany), published in the following paper.

[BG18] Ronny Bergmann and Pierre-Yves Gousenbourger. A variational model for data fitting on manifolds by minimizing the acceleration of a Bézier curve. *Frontiers in Applied Mathematics and Statistics*, 4(59):1–16, 2018. doi:10.3389/fams.2018.00059

Ch. 6 Interpolation with Bézier surfaces

Chapters 6 and 7 reproduce the approaches of Chapters 3 and 4, but in the bidimensional case.

In this chapter, we consider the interpolation problem. The first task, and the first contribution, is to extend the notion of manifold-valued Bézier curves (see Chapter 3) to their manifold-valued bivariate expression. Three different approaches are proposed, leading to different results. The second contribution comes with a discussion on the differentiability of the piecewise surfaces of types I, II or III, and with a procedure and constraints to patch Bézier surfaces in a differentiable way. Finally, we design two algorithms to compute the unconstrained control points in such a way that the resulting interpolating spline has minimal mean squared second derivative when \mathcal{M} is a Euclidean space. The first one involves parallel transport (this is a costly operation) while the second one solves does not.

Both algorithms result from collaborations with Prof. Benedikt Wirth and Dr. Paul Striewski (Münster Universität, Germany). They are published in two articles:

[AGSW16b] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striewski, and Benedikt Wirth. Differentiable piecewise-Bézier surfaces on Riemannian manifolds. *SIAM Journal on Imaging Sciences*, 9(4):1788–1828, 2016. doi: 10.1137/16M1057978

[AGSW16a] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striewski, and Benedikt Wirth. Differentiable piecewise-Bézier interpolation on Riemannian manifolds. In *ESANN2016*, pages 95–100. Springer, 2016

Ch. 7 Fitting with blended thin plate splines

Last but not least, the loop is closed with a final chapter on bidimensional fitting. This chapter follows the ideas of Chapter 4, but gives up on Bézier

1 | Introduction

surfaces to consider thin plate splines that are a more conventional generalization of smoothing splines for $d = 2$.

The main contribution here is the definition and the analysis of *blended surfaces* that satisfy the properties of Section 1.4. It is also a collaboration with Prof. Benedikt Wirth (Universität Münster, Germany). A publication is in preparation with extended results and analysis.

[AGW20] P.-A. Absil, Pierre-Yves Gouenbourger, and Benedikt Wirth. Smooth surface fitting in Riemannian manifolds using patch-wise linearization. In progress, 2020

Disclaimer: large parts of this thesis rely verbatim on the publications cited above, specifically [GSA14, AGS⁺15, SGJ15, AGSW16b, AGSW16a, GMM⁺17, GMA18c, BG18, MHA19, AGW20]. A complete list of the articles published during this thesis is provided at the end of the manuscript.

2

Concerning manifolds

RIEMANNIAN MANIFOLDS have a rich structure. They offer a natural way to represent (constrained) non-linear spaces. This preliminary chapter gives an overview of the main differential geometric tools used in this document.

Geometric elements on Riemannian manifolds can be expressed or explained in a natural way, using analogies to the Euclidean space. However, defining properly those notions requires to define the concept of smooth manifold, and itself requires to come back to the notions of chart and atlas.

The definitions considered in this chapter might be tedious to manipulate and the intuition behind each of them is sometimes difficult to imagine. This is why the chapter is separated in two parts. The first one (Section 2.1) is a rough and (hopefully) intuitive presentation of the main concepts used in the thesis. We recommend this section to every reader, especially those who discover the field. Note that the definitions given there are of course not rigorous. More formal definitions are given in the following sections (Sections 2.2, 2.3, 2.4, 2.5). They can be skipped, but they constitute a solid ground for the next chapters.

The exposition adopted here is mainly inspired from the books of Ab-sil *et al.* [AMS08], do Carmo [dC92], and recently Boumal [Bou20]. If the reader wishes to read a more complete and strong introduction to Riemannian manifolds, the following textbooks are also advised [Car46, O'N66, O'N83, Boo86, Lee97, Lee13].

2.1 A (not too) short introduction for the impatient

The point, the line, the square, the circle, the cube are basic elements of our daily life. These are also the first geometric tools that we encounter in the first lessons of geometry in the elementary school. We know how to measure the length of a line segment and how to evaluate the distance between two points, just as we can draw a *straight line* between these two points. These operations are well defined in what we call a *Euclidean space* E . Our three-dimensional space \mathbb{R}^3 is one of them.

More than 150 years ago, Riemann [Rie54] paved the way to a novel branch of differential geometry that is now named after him. This geometry is not Euclidean: the straight line is now curved which has a direct impact on the measure of the distance between two points. In a first intuition, one could see those spaces as *smooth surfaces* S embedded in E , with no holes, no kink, no boundaries (the sphere, the plane or the torus are classical examples of such surfaces embedded in \mathbb{R}^3).

One could think that every operation performed on those kind of spaces is *constrained* to S . Indeed, if the points belonging to S are seen in the *ambient* space E , they are not free to move wherever they want. They are limited to stick to the smooth surface S . This is the point of view that the almighty God would take as he is seeing us, moving on our spherical Earth, constrained to remain on its curved surface, stuck to the ground by the gravity.

However, one could prefer another viewpoint where there is no constraint at all, because no other world than the surface S would exist. This reminds us of the world we are living on. From our viewpoint of teeny tiny humans compared to the large Mother Earth, we all can consider (in a first approximation) that we move freely, unaware that we are restricted to the surface of a sphere in the universe. One could also think about the two-dimensional inhabitants of Flatland [Abb84] or the characters of Paper Mario, who can barely imagine that there might be a third dimension and live happily like cows in a field in their own subspace [Bou20].

Now let us take the viewpoint of the tiny humans on the surface of the Earth. The smooth surfaces S will be called a *smooth manifold* \mathcal{M} (see Section 2.2 for a formal, yet abstract, definition). While we are traveling from our front door to the neighbor's, one can say that we are moving straight. We can even represent the trip on a map, by just drawing a straight line.

At this level, the distance we have made is very small compared to the curvature of the Earth. This comes with the notion that Earth can be locally approximated as a flat space. Imagine that you just have little flat pieces of LEGO[®] but that you want nevertheless to build a sphere: you can do it, as soon as the sphere you build is large enough compared to the pieces of LEGO[®].

Similarly, smooth manifolds \mathcal{M} can be linearized locally around a given point x (Figure 2.2). This linearization is called *tangent space at x of the manifold \mathcal{M}* (or just *tangent space*) and is noted $T_x\mathcal{M}$. As suggested by its name, this space is *tangent* at x . The point of tangency is called the *root* of the tangent space. When the tangent spaces to \mathcal{M} are endowed with a so-called *metric*, i.e., an inner product for each tangent space, smoothly varying from one tangent space to another, the manifold \mathcal{M} is said to be equipped with a *Riemannian structure*. The smooth manifold \mathcal{M} is thus called *Riemannian manifold*. This notion of inner product is a key to be able to define a distance on manifolds. In this thesis, every manifold will be equipped with a Riemannian structure.

This leads us to a question: if manifolds can be seen locally as (flat) vector spaces, what happens when two points are not “local enough” to belong to a same tangent space? Indeed, the approximation of the manifold by a tangent space does not stand when the distances become more and more important: one has to take a broader point of view. When one of our politician is traveling by plane from Brussels to New York City, it become less and less accurate to consider his trip as a *straight line*. Of course, the plane was only *going straight*, but it actually followed the curvature of Earth. In the sense of the space “Earth”, the plane followed the simplest path, the one which costed the least energy: a *straight sphere-line*. In geometric language, this line is called a *geodesic*. Metric, geodesic and distance are extremely related. For instance, on the sphere (which represents the Earth), the geodesic is the *great arc* along the sphere between two points. The distance is given by the length of this great arc, not by the length of the straight line drawn *through* the sphere.

Both approaches have their advantages. Working on tangent spaces is evidently most practical: the tangent space is a Euclidean space for which rules of calculus are well defined. It is easier to work on in. Working directly on the manifold \mathcal{M} , on the other hand, cannot be ignored: the (non-linear) manifold is the space where points actually belong. In many applications, tangent spaces are exploited as much as possible. This means that there must be a way to switch from a tangent space to the manifold

tangent space

metric

Riemannian
manifold

geodesic



Fig. 2.1 Illustration of a geodesic connecting Brussels to New York city on the surface of Earth. The initial velocity ζ can be computed with the logarithmic map as $\log_x(y)$; the destination from Brussels in the direction of the initial velocity can be reached with the exponential map as $y = \exp_x(\zeta)$.

(and reversely). This way is called the *retraction*, and a remarkable one is the Riemannian *exponential map* $\exp_x(\zeta) = y$ (not to be confounded with the matrix exponential). It can be seen as the application that maps a tangent vector ζ from the tangent space $T_x\mathcal{M}$ to another point y onto the manifold \mathcal{M} ; the other way around is accomplished by the Riemannian *logarithm* $\log_x(y) = \zeta$, that allows to represent a point y from \mathcal{M} on $T_x\mathcal{M}$. This representation, here, is the tangent vector ζ . Those mappings are core concepts in this thesis, because they exist on every Riemannian manifold.

Another advantage of those maps is that they are obtained by following the geodesic starting with a given initial velocity. Let us take again this example of the politician flying from Brussels to New York City. In a first approximation, one could say that the plane has to follow a certain initial direction (ζ , for instance) such that, if it follows this direction as the crow flies, it will reach New York City. The exponential map is equivalent to the geodesics starting from Brussels (x), in the given direction (ζ), and reaching New York (y). The logarithm map is the mapping that evaluates the direction the plane must follow in order to link Brussels to New York with a geodesic. Then, under certain conditions, computing a geodesic between two points $x, y \in \mathcal{M}$ can be done using only those maps. In that case, the geodesic can be written $g(t; x, y) = \exp_x(t \log_x(y))$, see Figure 2.1

In general, the exponential map is a “many-to-one” mapping, *i.e.*, dif-

retraction
Riemannian
exponential

Riemannian
logarithm

Exp-Log

ferent tangent values can be given to \exp_x for the same result on \mathcal{M} . From Brussels, one can define several tangent vectors pointing in a given direction such that we reach Paris. The length of the tangent vector can be increased in such a way that we first turn around the Earth and then finish the trip at Paris. Two different vectors lead to the same destination. On the other hand, the logarithmic map is always “one-to-one” where it is defined. They will be an inverse of each other only if we limit the length of the vectors we provide to \exp_x . This maximal length is encapsulated in the notion of injectivity radius that is properly defined later in Definition 2.51. That definition had a serious impact on this thesis: it justifies by itself the methods developed in Chapter 4.

A last important element to introduce is the *parallel transport*. Tangent spaces approximate well the reality of the manifold for points “not too far away from the root”. Therefore, it makes little sense to try to represent, in the tangent space at Brussels, a point in the neighborhood of New York. However, comparing tangent vectors from two tangent spaces is sometimes useful. The parallel transport $P_{x \rightarrow y}(\xi)$ enables to transfer a vector ξ from the tangent space at x to another one at y , following a geodesic.

All the concepts presented here are summarized in Figure 2.2, where

parallel
transport

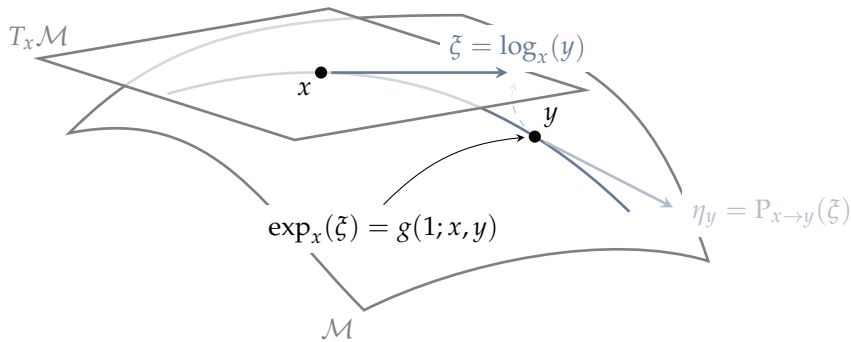


Fig. 2.2 Illustration of the core concepts used in this thesis: the tangent space $T_x\mathcal{M}$ to a manifold \mathcal{M} is a local approximation of \mathcal{M} by a vector space $T_x\mathcal{M}$; the geodesic $g(t; x, y)$ generalizes the idea of “straight line” from the Euclidean space, as it corresponds to a curve with “zero acceleration”; the exponential map $\exp_x(\xi)$ and the logarithmic map $\log_x(y)$, allow to navigate from the tangent space to the manifold (and inversely); the parallel transport $P_{x \rightarrow y}(\xi)$ permits to move a tangent vector from one tangent space to another, following a geodesic.

the Earth is here replaced by a more abstract representation of a manifold. This representation will be used throughout this thesis, even if some examples will be given on the sphere. Nonetheless, keeping this “Earth”-representation in mind will help the reader to embrace the concepts along the chapters.

The next sections are dedicated to more formal definitions. They can be skipped and are not necessary to understand the rest of the thesis, but they constitute a solid mathematical ground to all the presented concepts.

2.2 From a chart to a manifold

Roughly speaking, manifolds can be seen as sets that *look like* the Euclidean space in a close neighborhood. Even if we will focus later on *matrix manifolds* (i.e., manifolds whose elements are represented by matrices), we give here a general definition.

2.2.1 General definition of a manifold

Let us consider a certain d -dimensional set M , with no particular structure yet. As this set has to be interpreted locally as a Euclidean space, it is important to model M after \mathbb{R}^d in a one-to-one correspondence. This correspondence is called *chart*.

Definition 2.1. A d -dimensional *chart* on a set M is a pair (\mathcal{U}, φ) where

1. $\mathcal{U} \subseteq M$ is the *domain* of the chart, and
2. $\varphi : \mathcal{U} \rightarrow \mathbb{R}^d$ is a bijective map between \mathcal{U} and \mathbb{R}^d , with $\varphi(\mathcal{U})$ open on \mathbb{R}^d .

The inverse map $\varphi^{-1} : \varphi(\mathcal{U}) \rightarrow \mathcal{U}$ is called a *local parameterization* of M . Given $x \in \mathcal{U}$, the elements of $\varphi(x) \in \mathbb{R}^d$ are called the *coordinates* of x in M .

The notion of chart (\mathcal{U}, φ) allows objects expressed in \mathcal{U} to be transferred to \mathbb{R}^d . For instance, if $f : \mathcal{U} \rightarrow \mathbb{R}$, then $f \circ \varphi^{-1} : \mathbb{R}^d \rightarrow \mathbb{R}$, where the methods of real analysis exist. Therefore, it makes sense to cover the whole set M with a collection of charts such that every point $x \in M$ can be transferred to a Euclidean space. However, caution must be taken if a

point belongs to the domain of two different charts, and we must impose compatibility conditions.

Definition 2.2. Two charts (\mathcal{U}, φ) and (\mathcal{V}, ψ) of M are said C^∞ -compatible (or “smooth”-compatible) if they have the same dimension d and, if they are overlapping, they satisfy the following conditions:

1. $\varphi(\mathcal{U} \cap \mathcal{V})$ is open in \mathbb{R}^d ;
2. $\psi(\mathcal{U} \cap \mathcal{V})$ is open in \mathbb{R}^d ;
3. $\psi \circ \varphi^{-1} : \varphi(\mathcal{U}) \subset \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a diffeomorphism (i.e., it is a smooth function whose inverse is also smooth).

The function $\psi \circ \varphi^{-1}$ is also called *transition map* from φ to ψ . Compatible charts are illustrated in Figure 2.3

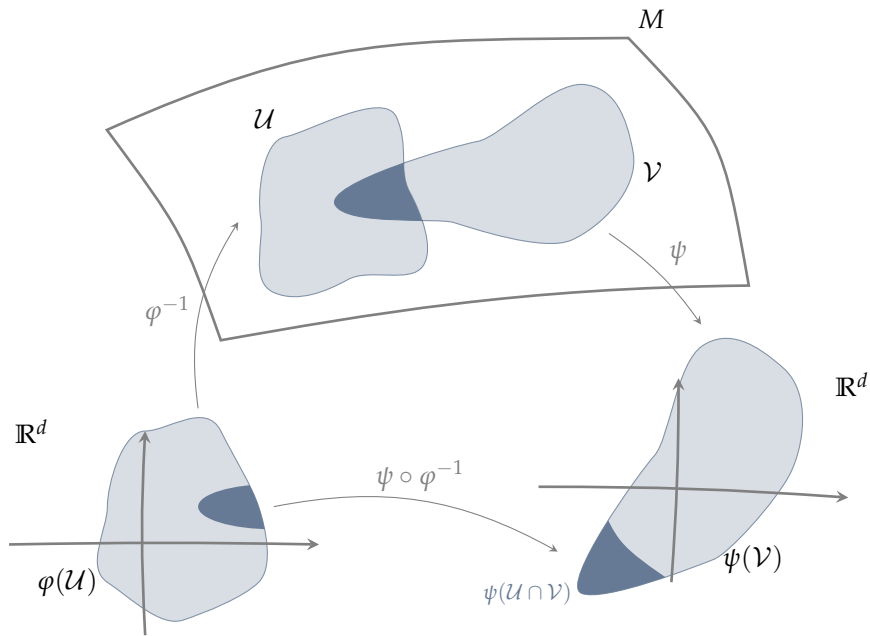


Fig. 2.3 Illustration of charts and compatible charts. The area $\mathcal{U} \subset M$ (resp. $\mathcal{V} \subset M$) is mapped to \mathbb{R}^d via the bijective map φ (resp. ψ). When two areas coincide like $\mathcal{U} \cap \mathcal{V}$ (darker areas), charts are said *compatible* if there exists a diffeomorphism $\psi \circ \varphi^{-1}$ between $\varphi(\mathcal{U} \cap \mathcal{V})$ and $\psi(\mathcal{U} \cap \mathcal{V})$.

It is now possible to define the notion of atlas: a collection of compatible charts that cover the whole set M .

compatible
charts

Definition 2.3. An *atlas* \mathcal{A} of M into \mathbb{R}^d is a collection of smooth-compatible d -dimensional charts on M whose domains cover M . Two atlases \mathcal{A}_1 and \mathcal{A}_2 are said *equivalent* if any pair of charts $(\mathcal{U}_1, \varphi_1) \in \mathcal{A}_1$ and $(\mathcal{U}_2, \varphi_2) \in \mathcal{A}_2$ are compatible (in other words, if $\mathcal{A}_1 \cup \mathcal{A}_2$ is still an atlas of M). Given an atlas \mathcal{A} , the unique set \mathcal{A}^+ of all charts (\mathcal{U}, φ) such that $\mathcal{A} \cup \{(\mathcal{U}, \varphi)\}$ is also an atlas. This set is called *maximal atlas* of M .

We can now give a permissive definition of a manifold.

Definition 2.4. A smooth d -dimensional *manifold* \mathcal{M} is a pair $\mathcal{M} = (M, \mathcal{A}^+)$ where \mathcal{A}^+ is a maximal atlas of the set M .

Note that we will simply name a manifold with \mathcal{M} instead of M when there is no confusion possible. Here is a basic example to illustrate our early definitions.

Example 2.5. The vector space \mathbb{R}^n has obviously a manifold structure $\mathcal{M} = (\mathbb{R}^n, \mathcal{A}^+)$. Indeed, its associated maximal atlas \mathcal{A}^+ is composed of the chart (\mathcal{U}, φ) where $\mathcal{U} = \mathbb{R}^n$ and φ is the identity map $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^n : x \mapsto x$. The same reasoning can be applied to the set $\mathbb{R}^{n \times p}$. Its manifold structure is of dimension np . A possible chart is $\varphi : \mathbb{R}^{n \times p} \rightarrow \mathbb{R}^{np} : X \mapsto \text{vec}(X)$, with $\text{vec}(X)$ stacking the columns of X on each other.

Remark 2.6. This definition is actually only permissive and is sufficient in most cases. However, it does not verify that the topology of the atlas \mathcal{A}^+ is (i) Hausdorff and (ii) second-countable, *i.e.*, [Lee13, Chap. 1]

- (i) for any pair of points $x, y \in M$, there exists disjoint open subsets $U, V \subset M$ such that $x \in U$ and $y \in V$,
- (ii) there exists a countable basis for the topology of M .

These last two conditions are necessary to “exclude certain unconventional topologies” [AMS08, §3.1.1]. For a complete presentation, we refer the reader to the textbooks of Absil *et al.* [AMS08, §3.1.2], Brickell and Clark [BC70], the introduction to smooth manifolds of Lee [Lee13], or to the recent and excellent summary of Boumal [Bou20, Chap. 8].

2.2.2 Smooth maps and submanifolds

In this thesis, we will mainly work with smooth manifolds. We define now the notion of *smooth mapping*. In the following, consider two smooth manifolds $\mathcal{M} = (M, \mathcal{A}^+)$ and $\mathcal{N} = (N, \mathcal{B}^+)$, respectively of dimension d_M and d_N .

Definition 2.7. A mapping $f : \mathcal{M} \mapsto \mathcal{N}$ is of class C^k if, for any point $x \in \mathcal{M}$, there exists a chart (\mathcal{U}, φ) of \mathcal{M} and a chart (\mathcal{V}, ψ) of \mathcal{N} such that $x \in \mathcal{U}$, $f(\mathcal{U}) \subset \mathcal{V}$ and

$$\hat{f} = \psi \circ f \circ \varphi^{-1} : \mathbb{R}^{d_M} \rightarrow \mathbb{R}^{d_N} : \varphi(\mathcal{U}) \mapsto \psi(\mathcal{V})$$

is of class C^k . The function \hat{f} is called the *local expression of f* in the charts (\mathcal{U}, φ) and (\mathcal{V}, ψ) . A map of class C^∞ is called *smooth*.

It is very easy to characterize a smooth real function whose domain is \mathcal{M} by generalizing this definition. Let $f : \mathcal{M} \rightarrow \mathbb{R}$. If $f \circ \varphi^{-1}$ is smooth, then f is a smooth function on \mathcal{M} . The same applies to manifold-valued functions $B : \mathbb{R} \rightarrow \mathcal{M}$, and smoothness is assessed if $\varphi \circ B$ is smooth. Functions like B are in the core of this work, as the main goal is to fit C^1 curves to data points on a manifold \mathcal{M} .

The next definitions are worth being stated as most of the manifolds present in this work are *embedded submanifolds* of the vector space $\mathbb{R}^{n \times p}$.

Definition 2.8. We say that \mathcal{N} is an *embedded submanifold*, a *regular submanifold*, or simply a *submanifold* of \mathcal{M} if $N \subseteq M$ and if and only if [AMS08, Prop. 3.3.2] around each point $x \in N$, there exists a chart (\mathcal{U}, φ) of \mathcal{M} such that

$$N \cup \mathcal{U} = \{x \in \mathcal{U} : \varphi(x) \in \mathbb{R}^{d_N} \times \{0\}\}.$$

In other words, $N \cup \mathcal{U}$ is an intersection between a d_N dimensional plane and $\varphi(\mathcal{U})$. The rest of the coordinates are filled with zeros.

Example 2.9. In her thesis, Massart [Mas19] gives the following example of hyperplanes embedded in \mathbb{R}^d . Consider a hyperplane $N := \{x \in \mathbb{R}^d, \text{ such that } x_i = 0\}$ for one certain i taken within $\{0, \dots, d\}$, and the linear manifold $\mathcal{M} = (\mathbb{R}^d, \varphi : x \mapsto x)$ defined in Example 2.5. For all $x \in N$, $\varphi(x) = x \in \mathbb{R}^{d-1} \times \{0\}$. Therefore, we can give a manifold structure to N as $\mathcal{N} = (\mathbb{R}^{d-1}, \varphi)$.

Example 2.10. Another intuitive example of embedded submanifolds is the case of the sphere S^{d-1} , embedded in the Euclidean space \mathbb{R}^d restricted to unit norm vectors.

Example 2.11. Hyperspheres are in fact one-dimensional versions of the Stiefel manifold. This manifold is extensively described in [AMS08, §3.3.2]. It is a submanifold of the linear space $\mathbb{R}^{n \times p}$, $p \leq n$ defined as $\text{St}(n, p) := \{X \in \mathbb{R}^{n \times p} \text{ such that } X^T X = I_p\}$, where I_p is the identity matrix of size p .

The structure inherited by \mathcal{N} from \mathcal{M} is a strong tie between the “mother” manifold and the submanifold. Indeed, a smooth function defined on \mathcal{M} and restricted to \mathcal{N} will remain a smooth function on \mathcal{N} . This is of particular interest for matrix manifolds, as they are all embedded submanifolds of the linear space $\mathbb{R}^{n \times p}$.

2.2.3 Product and quotient manifolds

Instead of extracting a subset from an existing set, one could also join two sets together in order to create a unique manifold called *product manifold*.

Definition 2.12. Consider the charts $(\mathcal{U}, \varphi) \in \mathcal{A}^+$ and $(\mathcal{V}, \psi) \in \mathcal{B}^+$. The structure of the *product manifold* $\mathcal{M} \times \mathcal{N}$ is given by all the charts obtained as $(\mathcal{U} \times \mathcal{V}, \varphi \times \psi)$, with $\varphi \times \psi : (x, y) \mapsto (\varphi(x), \psi(y))$.

Another way to restrict the dimension of a manifold is to characterize a set by means of equivalence relations. For instance, one could be interested in working with subspaces of \mathbb{R}^d . A subspace S can be represented by an infinite number of equivalent matrices whose columns form the basis of S . The interest of *quotient manifolds* is to summarize all those matrices as an equivalence class of points.

Definition 2.13. An *equivalent relation* is an operator \sim on a set M . It satisfies three properties, for all $x, y, z \in M$:

1. $x \sim x$ (by reflexivity);
2. $x \sim y$ if and only if $y \sim x$ (by symmetry);
3. $x \sim y$ and $y \sim z$ implies $x \sim z$ (by transitivity).

Two points satisfying $x \sim y$ are said to be *equivalent*. With such a relation, we can associate to each point $x \in M$ an *equivalence class* containing x , also called *fiber* of x denoted by

$$[x] := \{y \in M \text{ such that } y \sim x\}.$$

The *quotient set* of M (or the set of all equivalence classes of M) is denoted by $M / \sim = \{[x], x \in M\}$. It is composed of subsets of M . This quotient set can take several smooth manifold structures, which is unique under certain conditions defined in [AMS08, §3.4]. We will then talk about a *quotient manifold* M / \sim .

Quotient manifolds are extensively used and defined in the work of Massart to which we refer the reader [MA18, MHA19, Mas19].

We finish this section by presenting some examples of manifolds embedded in the Euclidean space or quotient of the Euclidean space. From the choice of the structure given to the manifold (embedded or quotient) depends the expressions of the different tools expressed on the manifold (geodesics, exponential map, metric, ...), see [MHA19] for an example on the manifold of positive semi-definite matrices of fixed rank (Example 2.18).

Example 2.14. The orthogonal group is an embedded submanifold of $\mathbb{R}^{d \times d}$ defined as

$$\mathcal{O}_d = \{X \in \mathbb{R}^{d \times d} \text{ such that } X^\top X = I_d\}.$$

orthogonal group \mathcal{O}_d

Example 2.15. The special orthogonal group $\text{SO}(n)$ is a subspace of \mathcal{O}_n restricted to all isometries of \mathcal{O}_n that preserve the orientation of the space. In short

$$\text{SO}(n) = \{X \in \mathcal{O}_n \text{ such that } \det(X) = 1\}.$$

special orthogonal group $\text{SO}(n)$

Example 2.16. The Stiefel manifold $\text{St}(n, p)$ defined in Example 2.11 can be seen as a quotient manifold of the orthogonal group. Indeed, the equivalence class is

$$[Q] = \left\{ Q \begin{pmatrix} I_p & 0 \\ 0 & Q_{n-p} \end{pmatrix} \text{ such that } Q_{n-p} \in \mathcal{O}_{n-p} \right\}.$$

Stiefel manifold

We often note it $\text{St}(n, p) = \mathcal{O}_n / \mathcal{O}_{n-p}$.

Example 2.17. The Grassmann manifold is a typical example of a quotient manifold. Indeed, $\text{Gr}(n, p)$ is the set of all p -dimensional subspaces of \mathbb{R}^n . Representing a subspace of \mathbb{R}^n is usually done as the space spanned by the columns of a given matrix $X \in \text{St}(n, p)$, i.e., an orthogonal basis. However, an infinite number of matrices can represent the subspace as soon as they are rotated by a matrix $Q \in \mathcal{O}_p$. We can then say that $XQ \sim X$, where $X \sim Y$ means that there exists a $Q \in \mathcal{O}_p$ such that $X = YQ$ (in other words, $X \sim Y \Leftrightarrow \text{span}(X) = \text{span}(Y)$). The Grassmann manifold can thus be defined as $\text{Gr}(n, p) = \text{St}(n, p) / \mathcal{O}_p$. For an alternative definition of the Grassmann manifold, we refer to [AMS08, §3.4.4].

Grassmann manifold $\text{Gr}(n, p)$

Example 2.18. As a last example, let us consider the manifold of fixed-rank positive-semidefinite matrices (see Chapters 3 and 4 for an application), defined as

$$\mathcal{S}_+(n, p) = \{X \in \mathbb{R}^{n \times n} \text{ such that } X \succeq 0, \text{rank}(X) = p\},$$

$\mathcal{S}_+(n, p)$

2 | Concerning manifolds

for $p \leq n$. Several representations exist for this manifold. They are equivalent, but have all their advantages and disadvantages (for instance, in some representation, there is no easy way to compute an exponential map). To name but a few, the manifold $\mathcal{S}_+(n, p)$ can be identified as

- a quotient $\mathbb{R}_*^{n \times p} / \mathcal{O}_p$ where $\mathbb{R}_*^{n \times p}$ is the manifold of matrices of full column rank [MA18];
- an embedded submanifold in $\mathbb{R}^{n \times n}$ [VAV09];
- a quotient $(\text{St}(n, p) \times \mathcal{P}_p) / \mathcal{O}_p$, where \mathcal{P}_p is the set of positive definite matrices [BS09];
- a homogeneous space [Van13].

We refer the reader to the thesis of Massart [Mas19, Chap. 8] for a complete discussion.

In the next chapters, we will define notions like *tangent spaces*, *geodesics*, *logarithm and exponential maps*. These tools can be derived in general using charts, but this implies intricate calculus which can be avoided by exploiting the structure of the linear space in which the manifold is embedded. Actually, the notion of chart will never be used in practice, even though it was crucial to introduce the rigorous notion of manifolds. This work deals with matrix manifolds where the tools of differential geometry do not depend on the choice of a chart: only the structure of the manifold matters.

2.3 A Euclidean space on manifolds: the tangent space

The notion of tangent space on manifolds relies on the notion of *tangent vector*, a generalization of the directional derivative of a real-valued function $f : \mathcal{M} \rightarrow \mathbb{R}$, in the direction η . When \mathcal{M} is a Euclidean space, the directional derivative, is defined simply as

$$Df(x)[\eta] = \lim_{t \rightarrow 0} \frac{f(x + t\eta) - f(x)}{t}.$$

However, the operation $f(x + t\eta) - f(x)$ requires a vector space structure that does not make sense when \mathcal{M} is a nonlinear manifold.

In [AMS08, §3.5], several approaches are presented. We keep here the one where $x + t\eta$ is interpreted as any curve $\gamma : \mathbb{R} \rightarrow \mathcal{M} : t \mapsto \gamma(t)$ passing

through x , with $\gamma(0) = x$. Therefore, $f(\gamma(t))$ becomes a smooth curve from $\mathbb{R} \rightarrow \mathbb{R}$ where the directional derivative is well-defined in the (tangent) direction given by the curve γ . But before we can define it properly, we must introduce the notion of smooth curve on a manifold.

Definition 2.19. A smooth mapping $\gamma : I \subseteq \mathbb{R} \rightarrow \mathcal{M} : t \mapsto \gamma(t)$ is called a *curve* on \mathcal{M} .

curve

The definition of *tangent vector* is extracted from [AMS08, §3.5.1].

Definition 2.20. Consider x , a point on a manifold \mathcal{M} and $\gamma : \mathbb{R} \rightarrow \mathcal{M}$ a smooth curve such that $\gamma(0) = x$. Consider also the set $\mathcal{F}(\mathcal{M})$ of all smooth real-valued functions on \mathcal{M} , and more specifically the subset $\mathcal{F}_x(\mathcal{M}) \subset \mathcal{F}(\mathcal{M})$ of all smooth real-valued functions defined in a neighborhood around x . A *tangent vector* ξ_x to \mathcal{M} at x is the mapping

$\mathcal{F}(\mathcal{M}),$
 $\mathcal{F}_x(\mathcal{M})$

tangent vector
 ξ_x

$$\dot{\gamma}(0) : \mathcal{F}_x(\mathcal{M}) \rightarrow \mathbb{R} : f \mapsto \dot{\gamma}(0)f := \left. \frac{d}{dt}f(\gamma(t)) \right|_{t=0}.$$

It is also called the *tangent vector* to the curve γ at $t = 0$. The curve γ is said to *realize* the tangent vector ξ_x .

We can now give a constructive definition of the tangent space.

Definition 2.21. The *tangent space* $T_x\mathcal{M}$ to \mathcal{M} at $x \in \mathcal{M}$ is the set of all tangent vectors to \mathcal{M} at x .

tangent space
 $T_x\mathcal{M}$

We call the point x the *root* of the tangent space (in several manuscript, it is also called the *foot*). When it is clear from the context, we will often omit the subscript x of the tangent vectors. However, we will never forget it when defining the tangent space $T_x\mathcal{M}$. A tangent space, its root and some tangent vectors are illustrated in Figure 2.4.

Property 2.22. Given $\xi, \eta \in T_x\mathcal{M}$ and $a, b \in \mathbb{R}$, $T_x\mathcal{M}$ has a vector space structure as $(a\xi + b\eta)f = a(\xi f) + b(\eta f)$.

A proof is given in [AMS08, p.34], where they pose $\xi = \dot{\gamma}_1(0)$, $\eta = \dot{\gamma}_2(0)$ and define a curve $\gamma(t) = \varphi^{-1}(a\varphi(\gamma_1(t)) + b\varphi(\gamma_2(t)))$.

Note 2.23. The aforementioned definition of tangent space is constructive and intuitive, as it gives the (useful) idea that tangent vectors *point* towards a direction from a given point x . However, this definition was initially possible because of the vector space structure of tangent spaces, which is

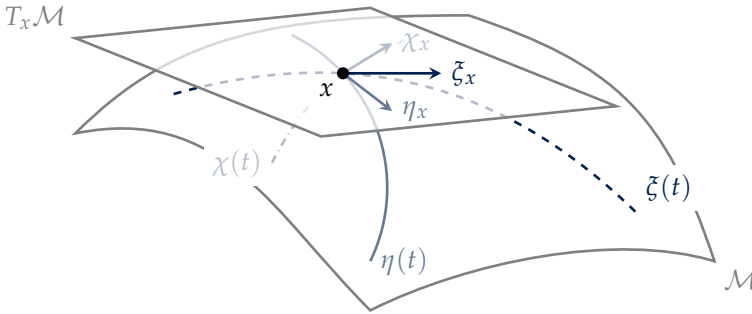


Fig. 2.4 Illustration of a tangent space $T_x\mathcal{M}$ and of several tangent vectors ξ_x, η_x and χ_x as velocities of the manifold-valued curves $\zeta(t), \eta(t)$ and $\chi(t)$. The point x is called the *root* of the tangent space on \mathcal{M} .

easy to imagine when \mathcal{M} is embedded in $\mathbb{R}^{n \times p}$. This is a strong hypothesis. If we abandon it, one has to rely on the notion of chart. Tangent spaces are defined in their full generality as equivalent classes of curves belonging to the set $\Gamma_x = \{\gamma : \mathbb{R} \rightarrow \mathcal{M} \text{ such that } \gamma \in C^1, \gamma(0) = x\}$. Consider a chart (\mathcal{U}, φ) on \mathcal{M} such that $x \in \mathcal{U}$ and $\gamma_1, \gamma_2 \in \Gamma_x$. We define the equivalence \sim on Γ_x as

$$\gamma_1 \sim \gamma_2 \Leftrightarrow \left. \frac{d}{dt} \varphi(\gamma_1)(t) \right|_{t=0} = \left. \frac{d}{dt} \varphi(\gamma_2)(t) \right|_{t=0}$$

i.e., γ_1 and γ_2 are equivalent if $\varphi \circ \gamma_1$ and $\varphi \circ \gamma_2$ have the same derivative for all φ . See [AMS08, Prop. 3.5.2] for a proof of the equivalence. Then, the *tangent space* is defined as the quotient space $T_x\mathcal{M} = \Gamma_x / \sim$ and a *tangent vector* to \mathcal{M} at x is given by the equivalence class $[\gamma]$. Given the chart (\mathcal{U}, φ) , the following bijective map

$$\theta_x^\varphi : T_x\mathcal{M} \rightarrow \mathbb{R}^n : [\gamma] \mapsto \theta_x^\varphi([\gamma]) = \left. \frac{d}{dt} \varphi(\gamma(t)) \right|_{t=0},$$

“induces a linear space structure over $T_x\mathcal{M}$ ” [Bou20]. This notion of tangent vector naturally induces a notion of directional derivative that reconciles the two approaches. Indeed, for $f \in \mathcal{F}_x(\mathcal{M})$ and $\zeta = [\gamma]$, we have

$$Df(x)[\zeta_x] = \left. \frac{d}{dt} f(\gamma(t)) \right|_{t=0} = \zeta_x f,$$

which turns out to correspond to Definition 2.20.

Interestingly, when \mathcal{M} is an embedded manifold represented as the level set of a constant function $g(x)$, then the tangent space is $T_x\mathcal{M} = \ker(Dg(x))$ [AMS08, p.40].

Example 2.24. Let $\mathcal{M} = \mathbb{S}^{n-1} = \{x \in \mathbb{R}^n \text{ such that } g(x) = x^\top x = 1\}$. The tangent space on the sphere \mathbb{S}^{n-1} is given by

$$T_x\mathcal{M} = \{\eta \in \mathbb{R}^n \text{ such that } \eta^\top x = 0\}.$$

Indeed, let $\gamma : t \mapsto \gamma(t)$ be a curve on \mathbb{S}^{n-1} . It follows that $\gamma(t)^\top \gamma(t) = 1$ and that $\dot{\gamma}(t)^\top \gamma(t) + \gamma(t)^\top \dot{\gamma}(t) = 2\dot{\gamma}(t)^\top \gamma(t) = 0$, where $\dot{\gamma}(t) \in T_{\gamma(t)}\mathcal{M}$.

Example 2.25. Consider the special orthogonal group $\mathcal{M} = \text{SO}(n)$ from Example 2.15. The tangent space on this group is

$$T_X\mathcal{M} = \{Z = X\Omega \text{ such that } \Omega \in \text{Skew}_n\}.$$

Indeed, the curve $X : t \mapsto X(t)$ satisfies $X(t)^\top X(t) = I_n$, where I_n is the identity matrix of size n . Therefore, $\dot{X}(t)^\top X(t) + X(t)^\top \dot{X}(t) = 0$. It imposes that $Z = \dot{X}(t) = X\Omega$, where Ω is skew-symmetric, i.e., $\Omega^\top = -\Omega$.

For the sake of completeness, we define the notions of *tangent bundle* and of *vector field*, directly related to tangent spaces.

Definition 2.26. The set of all tangent vectors is called the *tangent bundle*. It is defined as

$$T\mathcal{M} := \bigcup_{x \in \mathcal{M}} T_x\mathcal{M}.$$

The projection $\pi : T\mathcal{M} \rightarrow \mathcal{M} : \xi \mapsto \pi(\xi) := x$ extracts the root of the tangent vector $\xi \in T_x\mathcal{M}$.

The set $T\mathcal{M}$ admits a manifold structure. This is quite easy to build given a chart (\mathcal{U}, φ) of \mathcal{M} . Then, the mapping

$$\psi : \xi \mapsto (\varphi_1(x), \dots, \varphi_d(x), \xi\varphi_1, \dots, \xi\varphi_d),$$

is a chart of $T\mathcal{M}$ with domain $\pi^{-1}(\mathcal{U})$ [AMS08, §3.5.3].

We are now able to define vector fields as smooth mappings from \mathcal{M} to $T\mathcal{M}$. Just as scalar fields assign a scalar to each point $x \in \mathcal{M}$, vector fields assign a tangent vector $\xi \in T_x\mathcal{M}$ to each point $x \in \mathcal{M}$.

2 | Concerning manifolds

Definition 2.27 (*Vector field*). A *vector field* is a smooth mapping $\xi : \mathcal{M} \rightarrow T\mathcal{M} : x \mapsto \xi_x$ that assigns to each point $x \in \mathcal{M}$ a tangent vector $\xi_x \in T_x\mathcal{M}$. The set of all vector fields is noted $\mathcal{X}(\mathcal{M})$. For two vector fields $\xi, \eta \in \mathcal{X}(\mathcal{M})$ and a smooth function $f \in \mathcal{F}(\mathcal{M})$,

1. the *application* of the vector field to f is noted ξf , such that $(\xi f)(x) := \xi_x(f)$;
2. the *addition* of two vector fields is given by $(\eta + \xi)_x := \eta_x + \xi_x$;
3. the *product* of a vector field by f (not to be confounded with the application) is $(f\xi)_x := f(x)\xi_x$.

Many algorithms on manifolds exploit the vector space structure of tangent spaces as local vector space approximation of \mathcal{M} . To do so, the points on \mathcal{M} are mapped to a given tangent space $T_x\mathcal{M}$, where computation is done, and then the result is mapped back to \mathcal{M} thanks to a *retraction*.

Definition 2.28. Let R be a smooth mapping from the tangent bundle $T\mathcal{M}$ to \mathcal{M} and R_x be the restriction of R to $T_x\mathcal{M}$. R is called a *retraction* on \mathcal{M} if

1. for $0 \in T_x\mathcal{M}$, $R_x(0) = x$;
2. for $\xi_x \in T_x\mathcal{M}$ and $\gamma : \mathbb{R} \rightarrow \mathcal{M} : t \mapsto R_x(t\xi_x)$, then $\dot{\gamma}(0) = \xi_x$.

In general, there is an infinity of possible choices of retraction. For instance, on the sphere S^{n-1} , one could simply define

$$R_x : T\mathcal{M} \rightarrow \mathcal{M} : \xi \mapsto R_x(\xi) = \frac{x + t\xi_x}{\|x + t\xi_x\|_F}$$

as an acceptable retraction. In this thesis, we will principally use a remarkable retraction called the *exponential map* that follows a *geodesic* (see for that Definitions 2.48 and 2.46, respectively). These two objects only exist on *Riemannian manifolds*: this is the subject of the next section.

2.4 Riemannian structure of a manifold

In data processing, it is central to measure the similarity between two data points. This comes with the notion of distance. Such distance varies depending on the geometry of the space where the data belong. *Riemannian*

manifolds are manifolds where the tangent space is equipped with an *inner product* that implies a *norm* and thus a *length*.

We continue to consider a manifold \mathcal{M} , a point $x \in \mathcal{M}$ and its tangent space $T_x\mathcal{M}$.

Definition 2.29. An *inner product* $\langle \cdot, \cdot \rangle_x$ on $T_x\mathcal{M}$ is a bilinear, symmetric positive-definite form on $T_x\mathcal{M}$. In other words, let $\xi, \zeta, \eta \in T_x\mathcal{M}$, $a, b \in \mathbb{R}$. The following properties are respected

1. $\langle a\xi + b\zeta, \eta \rangle_x = a\langle \xi, \eta \rangle_x + b\langle \zeta, \eta \rangle_x$;
2. $\langle \xi, \eta \rangle_x = \langle \eta, \xi \rangle_x$;
3. $\langle \xi, \xi \rangle_x \geq 0$, with $\langle \xi, \xi \rangle_x = 0$ if and only if $\xi = 0$.

The inner product induces the *norm* $\|\xi\|_x = \sqrt{\langle \xi, \xi \rangle_x}$ of a tangent vector $\xi \in T_x\mathcal{M}$.

When every tangent space to \mathcal{M} is equipped with an inner product smoothly varying with $x \in \mathcal{M}$, we speak of *Riemannian metric* $g(\xi, \eta)$ for $\xi, \eta \in T_x\mathcal{M}$. The Riemannian metric is often interchangeably noted $g(\xi, \eta) = g_x(\xi, \eta) = \langle \xi, \eta \rangle_x$.

Definition 2.30. A pair (\mathcal{M}, g) with \mathcal{M} a manifold and g a Riemannian metric on \mathcal{M} is a *Riemannian manifold*.

The metric implies a notion of *distance* between two points on the Riemannian manifold (to be exact, the manifold must be *connected* to define the distance) and a notion of *length* of a curve.

Definition 2.31. The *length* of a curve $\gamma : [a, b] \rightarrow \mathcal{M}$ on a Riemannian manifold (\mathcal{M}, g) is

$$L(\gamma) = \int_0^1 \sqrt{g(\dot{\gamma}(t), \dot{\gamma}(t))} dt, \tag{2.1}$$

and its *energy* is given by

$$E(\gamma) = \int_0^1 g(\dot{\gamma}(t), \dot{\gamma}(t)) dt. \tag{2.2}$$

Let Γ be the set of all curves joining two points x and y on \mathcal{M} , the *Riemannian distance* is then simply

$$d_{\mathcal{M}} : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R} : (x, y) \mapsto d_{\mathcal{M}}(x, y) := \inf_{\Gamma} L(\gamma).$$

inner product

norm

metric

Riemannian manifold

Riemannian distance

In this thesis, the distance is usually simply denoted by $d(x, y)$, but the notation $d_{\mathcal{M}}(x, y)$ is used when it makes sense to specify the considered distance.

Remark 2.32. When the Riemannian manifold \mathcal{M} is Hausdorff (see Remark 2.6), the Riemannian distance turns \mathcal{M} into a metric space, *i.e.*, the distance can be seen as a metric on \mathcal{M} (it is *symmetric*, *positive definite*, and respects the *triangle inequality*) [AMS08, p. 46].

Example 2.33. The Euclidean metric is of course the canonical scalar product $\langle \eta, \xi \rangle = \eta^\top \xi$. The distance between two points $x, y \in \mathbb{R}^n$ is $d_{\mathbb{R}^n}(x, y) = \sqrt{\|(x - y)\|^2}$.

Example 2.34. When the sphere S^{n-1} is seen as an embedded manifold on \mathbb{R}^n , the metric is inherited from the canonical Euclidean one $g_x(\xi, \eta) = \xi^\top \eta$. The distance is the arc length between two points, *i.e.*, $d_{S^{n-1}}(x, y) = \arccos x^\top y$. Note that, in this case, the metric does not depend on the root of the tangent space. This is not always the case for other manifolds.

Note 2.35. Let (\mathcal{U}, φ) be a chart of \mathcal{M} . The components of the metric in the chart are given by $g_{ij} := g(E_i, E_j) \in \mathbb{R}$, where E_i is the i^{th} coordinate vector field, *i.e.*, coordinates such that any vector field $\xi \in \mathcal{X}(\mathcal{M})$ can be written $\xi = \sum_i \xi_i E_i$. Therefore, for $\xi, \eta \in \mathcal{X}(\mathcal{M})$, one has

$$g(\xi, \eta) = \sum_i \sum_j g_{ij} \xi_i \eta_j.$$

Thanks to the inner product, we can also introduce the notion of gradient of a real-valued function on \mathcal{M} that can only be defined based on the inner product. The definition is extracted from [AMS08, Eq. (3.31), p. 46].

Definition 2.36. Let $f: \mathcal{M} \rightarrow \mathbb{R}$ be a real-valued function on a manifold \mathcal{M} , $x \in \mathcal{M}$ and $\xi \in T_x \mathcal{M}$. The (Riemannian) gradient $\nabla_{\mathcal{M}} f(x) \in T_x \mathcal{M}$ of f at x is defined as the tangent vector that fulfills

$$\langle \nabla_{\mathcal{M}} f(x), \xi \rangle_x = Df(x)[\xi] \quad \text{for all } \xi \in T_x \mathcal{M}. \quad (2.3)$$

It is interesting to note that the Riemannian gradient depends on the chosen metric while the directional derivatives do not. The notion of Riemannian gradient will be exploited mainly in Chapter 5.

2.5 Remarkable mappings on Riemannian manifolds

Optimization and fitting on manifolds strongly rely on second order information (or acceleration) of a curve. For instance, to perform the Newton's algorithm on a manifold, the notion of "derivative of the gradient" is needed in order to compute the Hessian of the cost function. In the case of fitting or interpolation on manifolds, one of the key element is the *geodesic*, *i.e.*, a curve with "zero acceleration". Geodesics will be extensively used in this work, and are also implied in the notions of *exponential map* and *vector transportation*.

2.5.1 Connections

To define properly the notion of geodesic, we need to introduce another (more abstract) concept: the *affine connection*. A connection is just an additional structure that, roughly speaking, permits to compare vectors in tangent spaces or nearby points.

We first present connections on general manifolds but quickly move to the so-called Levi-Civita connection that is defined on Riemannian manifolds. We also present the derivative of a vector field, and, just after, the acceleration of a curve.

Definition 2.37. An *affine connection* on a manifold \mathcal{M} is the mapping

$$\nabla : \mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M}) \rightarrow \mathcal{X}(\mathcal{M}) : (\xi, \eta) \mapsto \nabla_{\eta}\xi,$$

satisfying the following three conditions. For $f, g \in \mathcal{F}(\mathcal{M})$, $\xi, \eta, \chi \in \mathcal{X}(\mathcal{M})$ and $a, b \in \mathbb{R}$, we have

1. $\mathcal{F}(\mathcal{M})$ -linearity: $\nabla_{f\eta+g\chi}\xi = f\nabla_{\eta}\xi + g\nabla_{\chi}\xi$;
2. \mathbb{R} -linearity: $\nabla_{\xi}(a\eta + b\chi) = a\nabla_{\xi}\eta + b\nabla_{\xi}\chi$;
3. Product rule (Leibniz' law): $\nabla_{\xi}(f\eta) = (\xi f)\eta + f\nabla_{\xi}\eta$.

Note that the symbol ∇ used here is *not* to be confused with the Euclidean gradient operator. It is often called "nabla" or "del" [AMS08, p. 94].

Definition 2.38 (*Lie bracket*). The *Lie bracket* $[\xi, \eta]$ of two vector fields $\xi, \eta \in$

affine
connection ∇

Lie bracket

2 | Concerning manifolds

$\mathcal{X}(\mathcal{M})$ is the mapping $[\cdot, \cdot] : \mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M}) \rightarrow \mathcal{X}(\mathcal{M}) : (\xi, \eta) \mapsto [\xi, \eta]$ defined as

$$[\xi, \eta]f := \eta(\xi f) - \xi(\eta f),$$

where ξf has to be understood as the application of ξ to f as defined in Definition 2.27

Definition 2.39. Let $\xi, \eta \in \mathcal{X}(\mathcal{M})$ be two vector fields on \mathcal{M} . The vector field $\nabla_\eta \xi$ is called the *covariant derivative of ξ with respect to η* for the affine connection ∇ . For $\eta_x \in T_x \mathcal{M}$, we have that $(\nabla_{\eta_x} \xi) = (\nabla_\eta \xi)_x \in T_x \mathcal{M}$.

In other words, at a point $x \in \mathcal{M}$, $(\nabla_\eta \xi)_x$ captures how the vector field ξ varies at x in the direction η_x .

Example 2.40. The *canonical connection* (also named *Euclidean connection*) corresponds to the directional derivatives in \mathbb{R}^n . For $\eta, \xi \in \mathcal{X}(\mathbb{R}^n)$, this connection is defined as

$$\nabla_\eta \xi := \lim_{t \rightarrow 0} \frac{\xi_{x+t\eta_x} - \xi_x}{t}.$$

Connections are thus often viewed as suitable generalization of the classical directional derivative.

Every second-countable Hausdorff manifold admits an affine connection [AMS08, Prop. 5.2.1]. Actually, there exists an infinity of affine connections, but some of them may be a better choice depending on the context. A remarkable one, called equivalently *Riemannian connection* or *Levi-Civita connection*, arises when the Riemannian structure is added on top of the general definition of the affine connection.

Theorem 2.41 (Levi-Civita). *Let (\mathcal{M}, g) be a Riemannian manifold and $\eta, \xi, \chi \in \mathcal{X}(\mathcal{M})$. There exists a unique affine connection ∇ (called Levi-Civita connection or Riemannian connection) that satisfies the two following properties:*

1. $\nabla_\eta \xi - \nabla_\xi \eta = [\eta, \xi]$ (symmetry),
2. $\chi g(\eta, \xi) = g(\nabla_\chi \eta, \xi) + g(\eta, \nabla_\chi \xi)$ (compatibility with the Riemannian metric).

Note 2.42. Every affine connection can be expressed as a table of real values directly related to the choice of charts (in practice, this representation can be cumbersome, but we mention it for the sake of completeness and refer to Absil *et al.* [AMS08] or Lee [Lee13] for more details). If we consider

the canonical basis $(e_1, \dots, e_n) \in \mathbb{R}^d$ and one of the possible connections on \mathbb{R}^d , we can compute the n^2 covariant derivatives of one basis element with respect to another. The k^{th} component of the $(i, j)^{\text{th}}$ covariant derivative is noted $\Gamma_{ij}^k := \nabla_{e_i} e_j$. They are called the *Christoffel symbols* in the basis (e_1, \dots, e_n) . The Euclidean connection (Example 2.40) obviously corresponds to $\Gamma_{ij}^k = 0$ as $e_i \perp e_j$ when $i \neq j$. If we consider again the basis of n coordinate vector fields $\{E_i\} \in \mathcal{X}(\mathcal{M})$, $i = 1, \dots, n$, of Remark 2.35, and a chart $(\mathcal{U}, \varphi) \in \mathcal{M}$, we can characterize the connection thanks to the Christoffel symbols. Indeed, the n^2 vector fields are given by

$$\nabla_{E_i} E_j = \sum_{k=1}^n \Gamma_{ij}^k E_k.$$

Here, the basis (E_1, \dots, E_n) is transferred to a Euclidean basis (e_1, \dots, e_n) through the chart as $e_i = D\varphi(x)[(E_i)_x]$, $x \in \mathcal{U}$, from which one can compute the associated Christoffel symbols.

2.5.2 Acceleration and geodesics

As connections encapsulate the notion of second order derivative, they enable to generalize the concept of straight lines (in \mathbb{R}^d) to the concept of *geodesics* (on a manifolds \mathcal{M}), *i.e.*, a curve with “zero acceleration”.

On the Euclidean space, one would verify that

$$\frac{d^2}{dt^2} \gamma(t) = 0, \quad \forall t.$$

On manifolds, we have only presented the definition of the velocity $\dot{\gamma}(t)$ of a curve (see Definition 2.20). Indeed, the mapping $t \mapsto \dot{\gamma}(t)$ is the *velocity vector field* along the curve γ at t . Similarly, we must properly define the *acceleration* of a curve $\gamma : \mathbb{R} \rightarrow \mathcal{M}$ at t . For this though, we must clarify the notion of derivation of a vector field.

Theorem 2.43 (Vector field derivation [AMS08, p. 102]). *Consider a Riemannian manifold (\mathcal{M}, g) equipped with a connection ∇ , and a C^2 curve $\gamma : \mathbb{R} \rightarrow \mathcal{M} : t \mapsto \gamma(t)$. Let $\zeta, \eta \in \mathcal{X}(\mathcal{M})$, $f \in \mathcal{F}(\mathbb{R})$ and $a, b \in \mathbb{R}$. There exists a unique operator $\frac{D}{dt} : \mathcal{X}(\gamma) \rightarrow \mathcal{X}(\gamma) : \zeta \mapsto \frac{D}{dt} \zeta$ satisfying*

1. $\frac{D}{dt}(a\zeta + b\eta) = a\frac{D}{dt}\zeta + b\frac{D}{dt}\eta$ (linearity),
2. $\frac{D}{dt}(f\zeta) = f'\zeta + f\frac{D}{dt}\zeta$ (product with a smooth function),

2 | Concerning manifolds

$$3. \frac{D}{dt}(\eta \circ \gamma)(t) = \nabla_{\dot{\gamma}(t)}\eta \text{ (composition).}$$

Definition 2.44. Consider a Riemannian manifold (\mathcal{M}, g) equipped with a connection ∇ , and a C^2 curve $\gamma : \mathbb{R} \rightarrow \mathcal{M} : t \mapsto \gamma(t)$. The *acceleration vector field* of a curve γ is the operator $\frac{D^2}{dt^2}$ given by

$$\frac{D^2}{dt^2}\gamma := \frac{D}{dt}\dot{\gamma}.$$

When the connection ∇ is the Levi-Civita connection, this operator is called the *Levi-Civita second covariant derivative*.

Often in the literature, the acceleration is also noted $\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t)$.

Note 2.45. The Levi-Civita second covariant derivative can also be computed via the Christoffel symbols (this approach is still cumbersome in most of the practical cases, but it offers a good alternative when simple and intuitive solutions do not exist). In a coordinate chart (\mathcal{U}, φ) of \mathcal{M} , let $(x^1(t), \dots, x^d(t)) := \varphi(\gamma(t))$ correspond to the coordinates in φ of a curve $\gamma(t) : \mathbb{R} \rightarrow \mathcal{M}$. The k^{th} component of the acceleration reads

$$\left(\frac{D^2}{dt^2}\gamma\right)^k := \frac{d^2}{dt^2}x^k(t) + \sum_{i=1}^d \sum_{j=1}^d \Gamma_{ij}^k \frac{d}{dt}x^i(t) \frac{d}{dt}x^j(t),$$

where Γ_{ij}^k are the Christoffel symbols of the affine connection of (\mathcal{U}, φ) evaluated at $\gamma(t)$. Note also that the velocity $\dot{\gamma}(t)^k = \frac{d}{dt}x^k(t)$ does not depend on the affine connection.

The following definitions directly follow from this notion of acceleration.

Definition 2.46. A *geodesic* on \mathcal{M} equipped with a connection ∇ is a C^2 curve $\gamma : \mathbb{R} \rightarrow \mathcal{M} : t \mapsto \gamma(t)$ with zero acceleration (for all t in the domain) in the sense of this connection.

Of course, different connections will produce different geodesics.

Remark 2.47. The paths γ minimizing the energy (2.2) subject to the endpoint conditions $\gamma(0) = x$ and $\gamma(1) = y$ for $x, y \in \mathcal{M}$, are called *minimizing geodesics* and their length is the *Riemannian distance* $d_{\mathcal{M}}(x, y)$. When it exists and is unique (a generic property when the manifold is complete [IT98]), the minimizing geodesic is denoted by $g(t; x, y)$. Note also that, as $g(t; x, y)$ minimizes (2.2), it also minimizes (2.1) as mentioned in [dC92, Chapter 3].

2.5.3 Exponential and logarithmic maps

According to [O’N83, Chap. 3, Lemma 22], for any tangent vector $\xi_x \in T_x\mathcal{M}$, there exists an interval $I \in \mathbb{R}$ around 0 on which there exists a unique geodesic $\gamma(t)$ satisfying $\gamma(0) = x$ and $\dot{\gamma}(0) = \xi_x$. This geodesic enables to define a remarkable retraction on Riemannian manifolds called the *exponential map*.

Definition 2.48. Let (\mathcal{M}, g) be a Riemannian manifold equipped with a connection ∇ . Let $x \in \mathcal{M}$, $\xi_x \in T_x\mathcal{M}$ and consider the unique geodesic $\gamma(t)$ from [O’N83, Chap.3, Lemma 22], satisfying $\gamma(0) = x$ and $\dot{\gamma}(0) = \xi_x$. The mapping $\exp_x: T_x\mathcal{M} \rightarrow \mathcal{M} : \xi \mapsto \exp_x(\xi) := \gamma(1)$ is called the *Riemannian exponential map*.

Riemannian
exponential
 $\exp_x(\xi)$
retraction

The exponential map is thus a retraction on \mathcal{M} , *i.e.*, it permits to map a tangent vector from $T\mathcal{M}$ to \mathcal{M} . Furthermore, the exponential map generalizes well the idea of moving “straight” in the direction of a given tangent vector, as it follows a geodesic. In many applications, it is very useful to map points from \mathcal{M} to $T\mathcal{M}$ via an “inverse retraction”. The inverse of the exponential map is called the *logarithmic map*. However, the exponential map is usually a many-to-one mapping. For instance, think about a cylinder of radius r : it is possible to build as an infinity of geodesics between two points a and b by looping around the cylinder. The initial tangent vectors at a have a norm equal to $v + k2\pi r$, $k \in \mathbb{Z}$, where v is the shortest distance between a and b , but the end-point will always be b .

Definition 2.49. Let $x, y \in \mathcal{M}$. If, among all tangent vectors ξ_x satisfying $\exp_x(\xi_x) = y$, the shortest one is unique, then this tangent vector is called *logarithmic map* (or *logarithm map*) and is denoted by $\log_x(y)$.

Riemannian
logarithm
 $\log_x(y)$

The exponential and the logarithmic map are illustrated in Figure 2.2.

Note 2.50. Like in the Note 2.45, the Christoffel symbols allow to compute the exponential map when there is no other simple solution. Computing the exponential maps reduces to computing the point at $t = 1$ of the curve obtained by the following differential equation in the chart (\mathcal{U}, φ)

$$\frac{d^2}{dt^2}x^k(t) + \sum_{i=1}^d \sum_{j=1}^d \Gamma_{ij}^k \frac{d}{dt}x^i(t) \frac{d}{dt}x^j(t) = 0, \text{ for } k = 1, \dots, d.$$

In general, solving this equation is computationally expensive (sometimes, even the Christoffel symbols have no closed form either). In this thesis,

2 | Concerning manifolds

we will consider “nice” manifolds where the exponential map comes in a closed-form solution.

Exponentials and logarithm maps are central to this thesis, as they are the principal maps that permit to pass from or to a tangent space or a manifold. In that case, one wants to ensure that for $x, y \in \mathcal{M}$, we can have $\exp_x(\log_x y) = y$. In order to ensure the well-posedness of the exponential and logarithmic map, we must restrict the domain of the exponential map. This comes with the notion of *injectivity radius*.

Definition 2.51. The *local injectivity radius* is the largest value $r_x \in \mathbb{R}$ such that the exponential map is a diffeomorphism over the ball $B_x(r_x) \subset T_x\mathcal{M}$. The *injectivity radius of a manifold* \mathcal{M} is the infimum over $x \in \mathcal{M}$ of all the local injectivity radii. The *cut locus* of $x \in \mathcal{M}$ is the set of points $y \in \mathcal{M}$ such that $y = \exp_x(t\xi_x)$ is a minimizing geodesic for $t \in [0, 1]$, but is no more minimizing for $t = 1 + \epsilon$. The shortest distance from x to its cut locus is r_x .

The definition of the injectivity radius allows to give conditions such that the exponential map and the logarithmic map are inverse functions.

Proposition 2.52. Let $x, y \in \mathcal{M}$ be two points, and r_x be the injectivity radius at x . Consider the sets $D_x := \{\xi_x \in T_x\mathcal{M} : \|\xi_x\|_x < r_x\}$ and $\mathcal{D}_x := \{y \in \mathcal{M} : d_{\mathcal{M}}(x, y) < r_x\}$, and the restricted logarithmic maps and exponential maps defined as

$$\exp_x : D_x \rightarrow \mathcal{M} : \xi_x \mapsto \exp_x(\xi_x), \quad (2.4)$$

$$\log_x : \mathcal{D}_x \rightarrow T_x\mathcal{M} : y \mapsto \log_x(y). \quad (2.5)$$

Then $\exp_x \circ \log_x$ is the identity on \mathcal{D}_x and $\log_x \circ \exp_x$ is the identity on D_x . Furthermore, for $x \in \mathcal{M}$, $y \in \mathcal{D}_x$, one can compute the minimizing geodesic between x and y as $g(t; x, y) = \exp_x(t \log_x(y))$.

2.5.4 Weighted geodesic average

In the following chapters, we will consider Bézier curves and the De Casteljau algorithm. To generalize this to bidimensional curves (*i.e.*, surfaces), we will need the notion of weighted geodesic average between points on \mathcal{M} . The weighted averages in metric spaces can be traced back to Fréchet [Fré48]. On Riemannian manifolds, they have been analyzed under var-

ious names (Riemannian center of mass, Riemannian barycenter, Karcher mean, Riemannian average, etc. [Kar77]).

Karcher mean

Definition 2.53. Let $n \in \mathbb{N}$. A *weighted geodesic average* of points $x_1, \dots, x_n \in \mathcal{M}$ for convex combination weights $w_1, \dots, w_n \in [0, 1]$ with $\sum_{i=1}^n w_i = 1$ is any point $x \in \mathcal{M}$ solving

geodesic average

$$\min_{x \in \mathcal{M}} J(x) = \min_{x \in \mathcal{M}} \sum_{i=1}^n w_i d^2(x_i, x).$$

If the minimizer exists and is unique, it will be denoted as

$$\text{av}[(y_1, \dots, y_n), (w_1, \dots, w_n)].$$

In the Euclidean space, the weighted geodesic average is a convex combination of the points $x_1, \dots, x_n \in \mathbb{R}^d$. It is straightforward to check that

$$\text{av}[(x_1, \dots, x_n), (w_1, \dots, w_n)] = \sum_{i=1}^n w_i x_i.$$

Likewise, the bilinear interpolation of points $x_{ij} \in \mathbb{R}^d, i, j \in \{0, 1\}$, at coordinates $(t_1, t_2) \in [0, 1]^2$ can be expressed as a weighted geodesic average

$$\begin{aligned} & (1 - t_1)(1 - t_2)x_{00} + (1 - t_1)t_2x_{01} + t_1(1 - t_2)x_{10} + t_1t_2x_{11} \\ & = \text{av}[(x_{00}, x_{01}, x_{10}, x_{11}), ((1 - t_1)(1 - t_2), (1 - t_1)t_2, t_1(1 - t_2), t_1t_2)]. \end{aligned}$$

On a Riemannian manifold, minimizing $J(x)$ requires to use geometric optimization procedures like those described in Absil *et al.* [AMS08] and implemented in the toolbox Manopt [BMAS14]. These methods require the derivative of $J(x)$ which is given by [Kar77, §1.2.]

distance (derivative)

$$\frac{dJ}{dx}(x) = - \sum_{i=1}^n w_i \log_x x_i \in T_x \mathcal{M}.$$

Note that, according to [AGSW16b, Remark 3], the weighted average between two points $x_1, x_2 \in \mathcal{M}$ reduces to a geodesic

$$\text{av}[(x_1, x_2), (1 - w, w)] = \hat{x} = g(w; x_1, x_2). \tag{2.6}$$

It can be expressed as $g(w; x, y) = \exp_x(w \log_x(y))$.

Exp-Log

The weighted averaging permits to introduce the notion of proper sub-

set, which will be a very useful tool to prove the existence of Bézier curves, later on (see Chapters 3 and 6).

Definition 2.54 (*Multigeodesic convexity [AGSW16b, Def. 5]*). A subset $U \subset \mathcal{M}$ is called *multigeodesically convex* if it contains any weighted geodesic average of any of its points. The *multigeodesically convex hull*, $\text{co}(U)$, of a set $U \subset \mathcal{M}$ is the smallest multigeodesically convex set $C \subset \mathcal{M}$ containing U .

Definition 2.55 (*Proper subset [AGSW16b, Def. 11]*). We call a subset $U \subset \mathcal{M}$ *proper* if the weighted geodesic averages between any finitely many points in U are unique and smoothly depend on the points and the weights.

Proper neighborhoods to $x \in \mathcal{M}$ always exist [AGSW16b, Prop. 12].

2.5.5 Parallel transport

As last concepts of this section, we present the notion of weighted geodesic average, of vector transportation and parallel transportation along a curve. These maps allow to move a tangent vector from one tangent space to another, the latter preserving some isometry.

Definition 2.56. Consider a manifold \mathcal{M} and a mapping

$$\mathbf{T} : T\mathcal{M} \oplus T\mathcal{M} \rightarrow T\mathcal{M} : (\xi, \eta) \mapsto \mathbf{T}_\eta(\xi),$$

where $T\mathcal{M} \oplus T\mathcal{M} = \{(\xi, \eta) \text{ such that } \xi, \eta \in T_x\mathcal{M}, x \in \mathcal{M}\}$ is the *Whitney sum*. The mapping \mathbf{T} is a *vector transportation* if it satisfies the following conditions for all $x \in \mathcal{M}$, $\eta, \xi, \chi \in T_x\mathcal{M}$ and $a, b \in \mathbb{R}$:

1. There exists a retraction R (associated with \mathbf{T}) on \mathcal{M} such that $\mathbf{T}_\eta(\xi) \in T_{R(\eta)}\mathcal{M}$ (associated retraction),
2. $\mathbf{T}_0(\xi) = \xi$ (consistency),
3. $\mathbf{T}_\eta(a\xi + b\chi) = a\mathbf{T}_\eta(\xi) + b\mathbf{T}_\eta(\chi)$ (linearity).

A particular vector transportation is called the *parallel transport* along a curve γ (usually, a geodesic). This transportation ensures that the vector remains constant while moving along the curve.

Definition 2.57. Consider a Riemannian manifold (\mathcal{M}, g) equipped with a connection ∇ . Let $x \in \mathcal{M}$, $\xi_x \in T_x\mathcal{M}$ and $\gamma : \mathbb{R} \rightarrow \mathcal{M}$ such that $\gamma(0) = x$. The *parallel transport* of ξ_x along γ is the unique vector field $\xi : \mathbb{R} \rightarrow T\mathcal{M}$ satisfying $\nabla_{\dot{\gamma}(t)}\xi = 0$.

We also refer to Section 5.2.1, where the so-called Jacobi fields are defined and are used as derivatives of geodesics.

Remark 2.58. For algorithmic reasons, one could want to transport a vector from the tangent space at x to the tangent space at y . To do so, a usual notation is

$$P_{x \rightarrow y}(\xi_x) = \mathbf{T}_{\log_x(y)}(\xi_x)$$

that is, the vector field that moves ξ_x along the geodesic $g(t; x, y)$ and that is evaluated at $t = 1$.

Remark 2.59. In general, exponential maps and logarithm maps can be very costly to compute (and parallel transportation even more). For computational reasons, it is sometimes recommended to approximate the exponential by a simpler (yet less accurate) retraction. This is usually done in optimization, like in the steepest gradient algorithm, where one has to move in a given direction at every step. In that case, using a retraction is useful because the retraction is applied in many iterations and the step is usually “sufficiently small”, such that any loose retraction is close to a geodesic. However, in this project, exponentials and logarithm will be called as few times as possible; furthermore, using a retraction might lead to give up on the principal quality of the reconstructed path: its differentiability. One goal of this work will be to analyze the number of exponentials and logarithm computed. This number will become a measure of the complexity of our algorithms, as the main part of the computational time will be caused by the evaluation of those maps.

Many other concepts can be defined and derive from the affine connection. For instance, the concepts related to the curvature of a manifold is a key to measure how far the manifold departs from a (flat) Euclidean space. It serves for instance to evaluate the distance of solutions computed on a tangent space compared to the one computed directly on a manifold. However, those concepts will not be used in this thesis, so we refer the reader to textbooks like [O’N83, Lee97] for a detailed presentation.

parallel
transport
 $P_{x \rightarrow y}(\xi_x)$

2.6 Additional numerical tools

We finish this chapter by introducing some additional numerical tools. They are used along this thesis but didn’t fit well in the previous sections.

2 | Concerning manifolds

We present here the *Riemannian finite differences* (one possible generalization of the Euclidean finite differences) and present a way to compute discrete geodesics. Some examples of geometric elements can be found in the Appendix E.

It is possible to exploit the Euclidean structure of tangent spaces to generalize the finite differences to manifolds.

Definition 2.60. Let $\gamma : \mathbb{R} \rightarrow \mathcal{M} : t \mapsto \gamma(t)$ be a curve on \mathcal{M} and $h \in \mathbb{R}$ be a step size. The two first order *finite differences* on \mathcal{M} are generalized as [Bou20]

$$\begin{aligned}\gamma'(t) &\simeq \frac{\log_{\gamma(t)}(\gamma(t+h)) - \log_{\gamma(t)}(\gamma(t-h))}{2h}; \\ \gamma''(t) &\simeq \frac{\log_{\gamma(t)}(\gamma(t+h)) + \log_{\gamma(t)}(\gamma(t-h))}{h^2}.\end{aligned}$$

In many interesting manifolds, the standard Riemannian operators can be expressed as closed formulae [AMS08, BA11, Ren11] or can be estimated via retractions as in [BMAS14]. More complicated manifolds (like the manifold of triangulated shells illustrated in Figure 2.5) require a numerical approximation of these operators.

The next definitions are summarized from the work of Rumpf and Wirth [RW15] and concern discrete geodesic calculus.

Definition 2.61. Let \mathcal{M} be a smooth Riemannian manifold. A *discrete distance* on \mathcal{M} is a smooth application $W : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$ that approximates the squared Riemannian distance as

$$d^2(x, y) = W[x, y] + \mathcal{O}(d^3(x, y)), \quad x, y \in \mathcal{M}. \quad (2.7)$$

To ensure the efficiency of the numerical methods, it is important to choose W easy to evaluate.

Definition 2.62. Consider a $(n+1)$ -tuple of $(n+1)$ points (x_0, \dots, x_n) on \mathcal{M} . This tuple is called *discrete n -path* between x_0 and x_n . Its length L and energy E are defined as

$$L[x_0, \dots, x_n] = \sum_{i=1}^n \sqrt{W[x_{i-1}, x_i]}, \quad (2.8)$$

$$E[x_0, \dots, x_n] = n \sum_{i=1}^n W[x_{i-1}, x_i]. \quad (2.9)$$

The $(n + 1)$ -tuple minimizing the energy of the discrete n -path is called a *discrete n -geodesic*:

$$\min_{x_1, \dots, x_{n-1} \in \mathcal{M}} E[x_0, x_1, \dots, x_{n-1}, x_n]. \tag{2.10}$$

geodesic

In [RW15] it is shown that discrete geodesics approximate the true continuous geodesics as $n \rightarrow \infty$. A discrete n -geodesic is represented in Figure 2.5.

Now that the discrete n -geodesic is defined, it is possible to express the discrete analog of the weighted geodesic average of Definition 2.53.

Definition 2.63. Consider a set of k points $X = \{x^1, \dots, x^k\} \in \mathcal{M}$. The *discrete weighted average* of the set X is the point $x \in \mathcal{M}$ solving

average

$$\min_{x \in \mathcal{M}} \min_{\substack{x_i^j \in \mathcal{M} \\ i=1, \dots, n-1, \\ j=1, \dots, k}} \sum_{j=1}^k w_j E[x, x_1^j, \dots, x_{n-1}^j, x^j]. \tag{2.11}$$

When the optimal point x is found, $(x, x_1^j, \dots, x_{n-1}^j, x^j)$ is the discrete n -geodesic from x to x^j , and $E[x, x_1^j, \dots, x_{n-1}^j, x^j]$ is the discrete approximation of the squared Riemannian distance $d^2(x, x^j)$.

To define the discrete logarithm, we will require \mathcal{M} to be identified with a subset of some embedding Banach space B such that, for a discrete n -geodesic (x_0, \dots, x_n) with $x_0 = x_A$ and $x_k = x_B$, the difference $x_1 - x_0$ is well-defined.

Definition 2.64. The *discrete logarithm* $\left(\frac{1}{n}\text{LOG}\right)$ is defined as

discrete logarithm

$$\left(\frac{1}{n}\text{LOG}\right)_{x_A} (x_B) = x_1 - x_0, \tag{2.12}$$



Fig. 2.5 Discrete 4-geodesic between two triangulated shells (Section 6.6). The mesh data used in this figure was made available by Robert Sumner and Jovan Popovic, MIT Computer Graphics Group.

2 | Concerning manifolds

where $\frac{1}{n}$ is to be interpreted as part of the symbol $\left(\frac{1}{n}\text{LOG}\right)_x(y)$.

Under certain regularity assumptions on the Riemannian metric and the functional W , it can be shown that $n \left(\frac{1}{n}\text{LOG}\right)_{x_A}(x_B)$ tends to $\log_{x_A}(x_B)$ when n tends to infinity [RW15].

Finally, we define the discrete exponential map EXP^n . For a discrete k -geodesic (y_0, \dots, y_k) , we expect EXP^n to reflect the properties of its continuous counterpart, *i.e.*, to satisfy $\text{EXP}_{x_0}^n(v) = x_n$ where $v = \left(\frac{1}{n}\text{LOG}\right)_{x_0}(x_n)$.

Definition 2.65. Let $v \in B$ and let

$$\begin{aligned}\text{EXP}_x^1(v) &= \left(\frac{1}{1}\text{LOG}\right)_x^{-1}(v) = x + v, \\ \text{EXP}_x^2(v) &= \left(\frac{1}{2}\text{LOG}\right)_x^{-1}(v).\end{aligned}$$

The *discrete exponential map* is defined recursively as

$$\text{EXP}_x^n(v) = \text{EXP}_{\text{EXP}_x^{n-2}(v)}^2(\tilde{v}) \quad \text{with} \quad \tilde{v} = \text{EXP}_x^{n-1}(v) - \text{EXP}_x^{n-2}(v). \quad (2.13)$$

Note that EXP^2 is non-trivial and is here simply expressed as the inverse of the discrete logarithm. It remains then to determine a way to compute this object. It follows from the definition of $\left(\frac{1}{2}\text{LOG}\right)$ that $x_2 = \text{EXP}_{x_0}^2(v)$ satisfies

$$x_0 + v = \operatorname{argmin}_{x \in \mathcal{M}} (W[x_0, x] + W[x, x_2]).$$

Therefore, x_2 can be obtained by solving the corresponding Euler-Lagrange equation

$$x_2 \in \mathcal{M} : \partial_2 W[x_0, x_0 + v] + \partial_1 W[x_0 + v, x_2] = 0. \quad (2.14)$$

Finally, we present the *discrete parallel transport*. To transport a vector along a discrete curve, we use a first order approximation of the parallel transport called *Schild's ladder*.

Definition 2.66. Let (x_0, \dots, x_n) be a discrete curve in \mathcal{M} and $v_0 \in T_{x_0}\mathcal{M}$, the vector to transport from x_0 to x_n . The *discretely transported tangent vector* v_i at a point x_i , $i \in \{1, \dots, n\}$, is computed recursively following the

algorithm illustrated in Figure 2.6:

$$\begin{aligned} x_{i-1}^p &= \text{EXP}_{x_{i-1}}^1(v_{i-1}), \\ x_i^{\text{mid}} &= \text{EXP}_{x_{i-1}^p}^1\left(\left(\frac{1}{2}\text{LOG}\right)_{x_{i-1}^p}(x_i)\right), \\ x_i^p &= \text{EXP}_{x_{i-1}^p}^2\left(\left(\frac{1}{1}\text{LOG}\right)_{x_{i-1}^p}(x_i^{\text{mid}})\right), \\ v_i &= \left(\frac{1}{1}\text{LOG}\right)_{x_i}(x_i^p). \end{aligned}$$

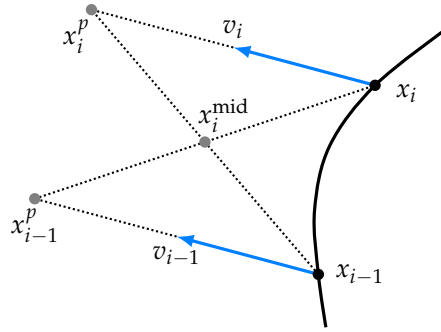


Fig. 2.6 Illustration of one iteration of Schild's ladder, approximating the parallel transport of a vector along a discrete curve (x_0, \dots, x_n) .

To find a numerical solution to the discrete geodesic (2.10), the discrete averaging (2.11) and the discrete exponential (2.14) (all other equations are trivial to solve), a Newton's method can be used. These discrete Riemannian operators are used in Sections 6.6.3 and 6.6.4. Note that as these objects require to solve an optimization problem, they are usually costly to compute.

3

Interpolation with Bézier curves

INTERPOLATION ON MANIFOLDS with composite Bézier curves is the first contribution of this thesis. The journey started with my Master thesis (summarized in a publication [GSA14]); it was then followed by the Master thesis of Arnould (and published in collaboration [AGS⁺15]); finally, it has been consolidated in [GMA18c]. In those three works, the composite Bézier curve interpolates $n + 1$ data points by means of n Bézier curves. Initially, the first and last segments of the composite Bézier curves were *quadratic* while all the others were *cubic*. In [GMA18c], all curves are cubic, such that some remarkable properties are achieved.

In short, the approach works as follows. Given the manifold-valued data points d_0, \dots, d_n at increasing parameter-values $t_0 < t_1 < \dots < t_n$ (without loss of generality, we will always set $t_i = i$, $i = 0, \dots, n$), one seeks the interpolating C^1 composite cubic Bézier curve $\mathbf{B}: [t_0, t_n] \rightarrow \mathcal{M}$ minimizing (1.1) for $\lambda \rightarrow \infty$, *i.e.*,

$$\min_{\mathbf{B}} \int_{t_0}^{t_n} \left\| \frac{D^2 \mathbf{B}(t)}{dt^2} \right\|_{\mathbf{B}(t)}^2 dt \quad \text{s.t.} \quad \mathbf{B}(t_i) = d_i. \quad (3.1)$$

The first two sections of this chapter are dedicated to reminders on Bézier curves (Section 3.1) and on their generalization to manifolds (Sec-

tion 3.2). In Section 3.3, the approach of [GSA14] will be recalled as a background necessary to the application proposed in Section 3.5. The main contributions of this chapter are Section 3.4 and Section 3.5. The former summarizes and analyzes an interpolation technique by means of composite cubic Bézier splines. It is mainly based on the journal publication [GMA18c, §3]; the latter is a comparison of the interpolation methods described in Section 3.3 and [AGS⁺15], applied to a medical application called transvaginal ultrasounding. This last section is based on a collaboration with Dr. Chafik Samir (Université de Clermont, France) [SGJ15].

Note also that the methods described in this chapter will be generalized to bidimensional interpolation (see Chapter 6).

3.1 Euclidean Bézier curves

In this section, we briefly summarize the concept of Bézier curves in a Euclidean space \mathbb{R}^m . We also define the *composite* Bézier curve and the conditions needed to obtain C^1 -continuity along this curve. Finally, we present the De Casteljau algorithm that evaluates Bézier curves in a recursive way and admits a well-known generalization to manifolds [PN07]. More details about Bézier curves can be found in [Far02].

Bézier curves in \mathbb{R}^m are nothing else than polynomials expressed in a particular basis.

Definition 3.1 (*Bézier curve*). Let $b_0, \dots, b_K \in \mathbb{R}^m$, $K \in \mathbb{N}$, be a sequence of *control points*. The *Bézier curve* $\beta_K: [0, 1] \rightarrow \mathbb{R}^m$ of degree K is defined as

$$\beta_K(\cdot; b_0, \dots, b_K): [0, 1] \rightarrow \mathbb{R}^m, t \mapsto \sum_{j=0}^K b_j B_{jK}(t), \quad (3.2)$$

where K is called the order of the curve ($K = 3$ for cubic Bézier curves) and $B_{jK}(t)$ denotes the j^{th} Bernstein polynomial of degree K ,

$$B_{jK}(t) = \binom{K}{j} t^j (1-t)^{K-j}. \quad (3.3)$$

Example 3.2. The linear Bézier curve $\beta_1(t; b_0, b_1)$ is just the (straight) line segment

$$\beta_1(t; b_0, b_1) = (1-t)b_0 + tb_1 \quad (3.4)$$

control point
Bézier curve
 β_K

Bernstein
polynomial

connecting the control points b_0 and b_1 . The explicit formulae for the quadratic (see Figure 3.1) and cubic Bézier curves read

$$\beta_2(t; b_0, b_1, b_2) = b_0(1-t)^2 + 2b_1(1-t)t + b_2t^2, \quad (3.5)$$

$$\beta_3(t; b_0, b_1, b_2, b_3) = b_0(1-t)^3 + 3b_1(1-t)^2t + 3b_2(1-t)t^2 + b_3t^3, \quad (3.6)$$

for given control points $b_0, b_1, b_2 \in \mathbb{R}^m$ and an additional point $b_3 \in \mathbb{R}^m$ for the cubic case.

Property 3.3 (Convex hull). The Bézier curve fully lies inside the convex hull of its control points b_0, \dots, b_K , since for each fixed t , the weights $B_{jK}(t)$ form a partition of unity and thus can be interpreted as convex combination coefficients (Figure 3.1).

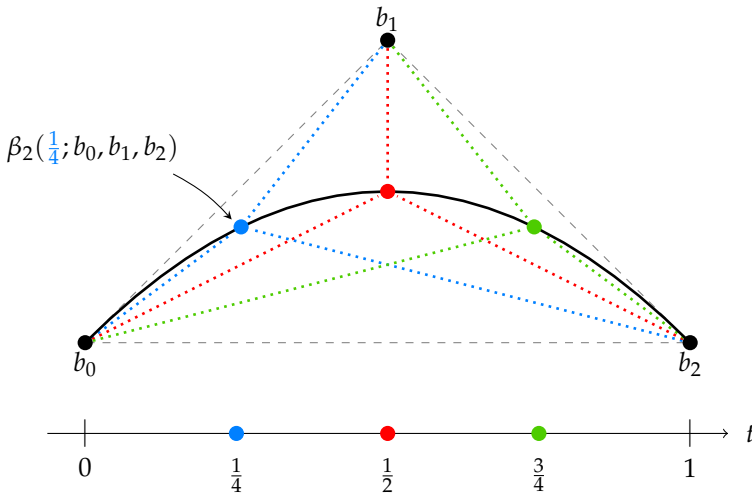


Fig. 3.1 Computation of a quadratic ($K = 2$) Bézier curve via equation (3.2) as a weighted mean of all control points. Control points are indicated by filled black dots; the Bézier polygon is shown as gray dashed lines.

Property 3.4 (Continuity and differentiability). The Bézier curve interpolates the first (b_0) and last (b_K) control points. Its velocity (i.e., time-derivative) $\dot{\beta}_K$ at $t = 0$ (resp. $t = 1$) is tangent to the segment joining its two first (resp.

3 | Interpolation with Bézier curves

two last) control points (see Figure 3.1). That is:

$$\beta_K(0; b_0, \dots, b_K) = b_0, \quad (3.7)$$

$$\beta_K(1; b_0, \dots, b_K) = b_K, \quad (3.8)$$

$$\dot{\beta}_K(0; b_0, \dots, b_K) = K(b_1 - b_0), \quad (3.9)$$

$$\dot{\beta}_K(1; b_0, \dots, b_K) = K(b_K - b_{K-1}). \quad (3.10)$$

A simple algorithm to evaluate $\beta_K(t; b_0, \dots, b_K) =: x_0^{[K]}$ at time $t \in [0, 1]$ is the so-called *De Casteljau algorithm* [Far02, §4.2]. As it is based only on convex combinations of two points, it has a simple geometric interpretation: every step of the algorithm consists in joining two points by a straight line and evaluating it at the right position, given by the weights. The algorithm is defined in Definition 3.5 and is represented in Figures 3.2 and 3.3.

Definition 3.5 (*De Casteljau algorithm*). The De Casteljau algorithm on \mathbb{R}^m reads

$$\begin{aligned} x_i^{[0]} &:= b_i && \text{for } i = 0, \dots, K \\ x_j^{[k]} &:= (1-t)x_j^{[k-1]} + tx_{j+1}^{[k-1]} && \text{for } k = 1, \dots, K \\ &&& \text{and } j = 0, \dots, K-k. \end{aligned}$$

Several Bézier curves can be joined to form a composite Bézier curve.

Definition 3.6 (*Composite Bézier curve*). Consider a sequence $(\beta_{K_i}^i)_{i=0}^{N-1}$ of N Bézier curves of degree K_i , $i = 0, \dots, N-1$, determined by control points $b_0^i, \dots, b_{K_i}^i \in \mathbb{R}^m$. The *composite Bézier curve* \mathbf{B} is defined as

$$\mathbf{B}: [0, N] \rightarrow \mathbb{R}^m, \quad t \mapsto \beta_{K_i}^i(t-i; b_0^i, \dots, b_{K_i}^i), \quad i = [t], \quad (3.11)$$

where $[t]$ is the largest integer $i \leq t$, with an exception for $[N] := N-1$.

Remark 3.7. For simplicity, we restrict ourselves to composite curves whose pieces are defined on intervals $[i, i+1]$, $i \in \mathbb{Z}$, but the step to define them on any interval $[t_i, t_{i+1}]$ is direct.

The next proposition follows directly from (3.7) and (3.8).

Proposition 3.8 (*Continuity of the composite Bézier curve*). *The composite Bézier curve made of N segments is continuous if the last and first control points of every two consecutive Bézier curves are the same. We introduce them as $p_i := b_{K_i}^i = b_0^{i+1}$, $i = 0, \dots, N-2$.*

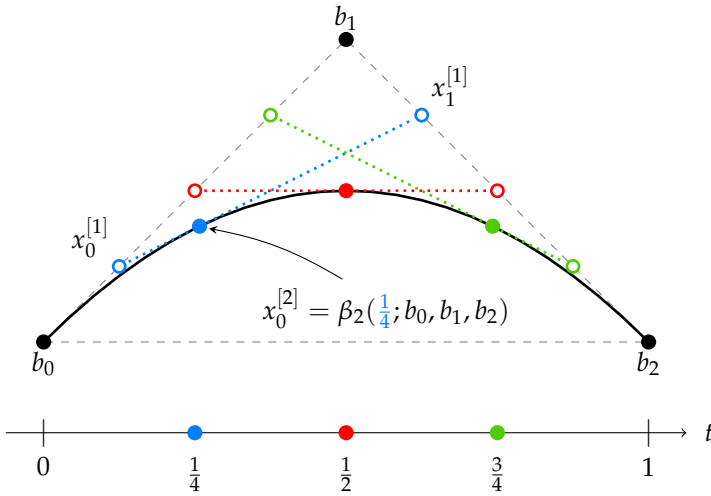


Fig. 3.2 Geometric interpretation of the De Casteljau algorithm [Far02] applied to a quadratic Bézier curve $\beta_2(t; b_0, b_1, b_2) \in \mathbb{R}^2$. At each step, one computes a convex combination (or *weighted mean*) of two consecutive points. Control points are indicated by filled black dots; the Bézier polygon is shown as gray dashed lines.

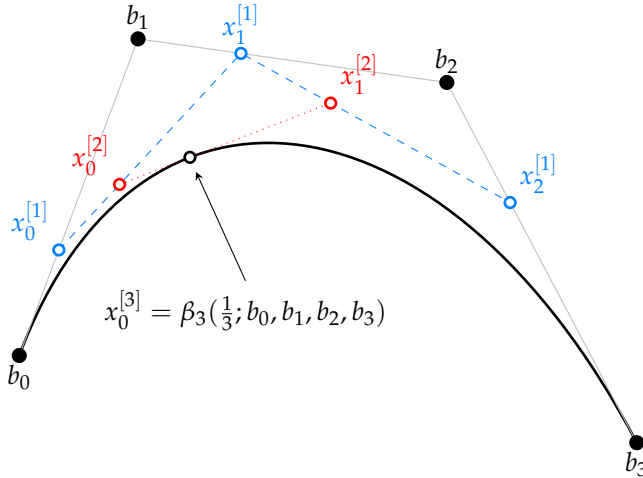


Fig. 3.3 Construction of a cubic Bézier curve via the De Casteljau algorithm.

3 | Interpolation with Bézier curves

In addition to continuity, we state now conditions for \mathbf{B} to be continuously differentiable at $t = i$ (differentiability is ensured on the rest of the domain since Bézier curves are polynomials). We introduce the notations b_i^- for the second to last control point of the $(i - 1)^{\text{th}}$ Bézier curve of \mathbf{B} (namely, $b_{K_{i-1}-1}^{i-1}$), and b_i^+ for the second control point of the i^{th} Bézier curve (i.e., b_1^i). The conditions follow from (3.9) and (3.10).

Proposition 3.9 (Differentiability of the composite Bézier curve). *The composite Bézier curve is differentiable at $t = i$ if the following C^1 -conditions hold:*

differentiability conditions

$$p_i = \frac{K_{i-1}b_i^- + K_i b_i^+}{K_{i-1} + K_i}, \quad i = 1, \dots, N - 1. \quad (3.12)$$

Geometrically, this condition means that p_i, b_i^- and b_i^+ must be aligned.

Example 3.10 (Composite curve of two pieces). A composite Bézier curve composed of two segments of degree K is expressed as

$$\mathbf{B} : [0, 2] \rightarrow \mathbb{R}^m : \quad t \mapsto \begin{cases} \beta_K(t; b_0^1, \dots, b_K^1) & \text{if } t \in [0, 1] \\ \beta_K(t - 1; b_0^r, \dots, b_K^r) & \text{if } t \in (1, 2] \end{cases} \quad (3.13)$$

and is continuous if and only if $b_0^r = b_K^1$. It is first order differentiable if and only if $b_K^1 = \frac{b_{K-1}^1 + b_1^r}{2}$.

Example 3.11 (C^1 -composite cubic Bézier curve). A composite cubic Bézier curve \mathbf{B} is represented in Figure 3.4. $\mathbf{B} : [0, 5] \rightarrow \mathbb{R}$ is composed of five cubic Bézier curves and is defined as $\mathbf{B}(t) = \beta_3(t - i; p_i, b_i^+, b_{i+1}^-, p_{i+1})$ for $i = \lfloor t \rfloor$. Continuity is trivial. Continuous differentiability is given by $p_i = 0.5(b_i^- + b_i^+), i = 1, \dots, 4$.

3.2 Bézier curves on manifolds

We can now recall the definition of composite Bézier curves on manifolds. Crouch [CKS99] and Lin and Walker [LW01] proposed a generalization of the De Casteljau algorithm on Lie Groups and on Riemannian manifolds. See also [BF01, NYP13]. Furthermore, the C^1 -conditions on the composite Bézier curve can be obtained thanks to the work of Popiel and Noakes [PN07]. We briefly review these results.

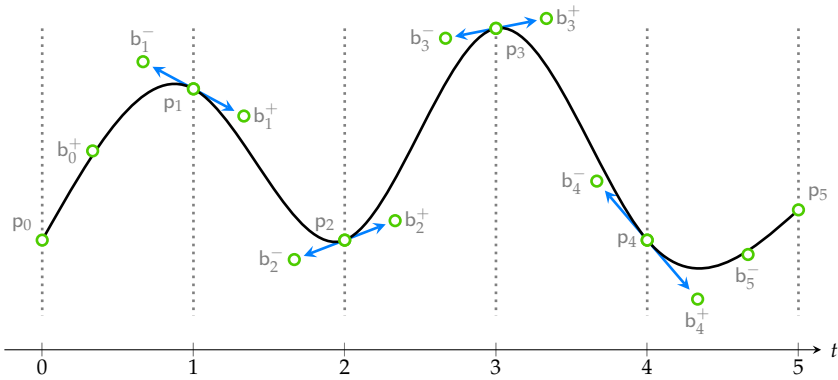


Fig. 3.4 Schematic representation of a composite cubic Bézier curve $\mathbf{B}: [0,5] \rightarrow \mathcal{M}$, for $\mathcal{M} = \mathbb{R}$, composed of five segments. The control points (green circles) fully determine the curve; continuous differentiability is reached at the junction p_i of the segments if the consecutive control points (b_i^-, p_i, b_i^+) , $i = 1, \dots, n - 1$, are aligned on a geodesic (the blue arrows draw the first derivative of \mathbf{B}).

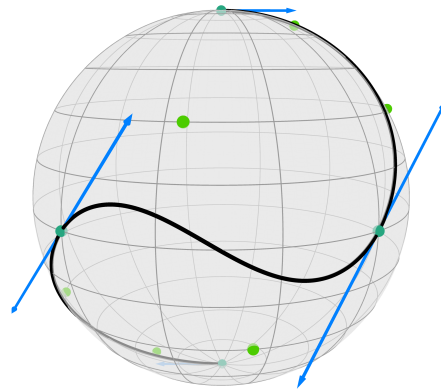


Fig. 3.5 A composite cubic Bézier curve $\mathbf{B}: [0,3] \rightarrow \mathbb{S}^2$. The end points p_i , $i = 0, \dots, 3$, (cyan) and intermediate points b_i^\pm (green) determine its shape; continuous differentiability is illustrated by the logarithmic maps $\log_{p_i}(b_i^\pm)$, $i \in \{1, 2\}$ (blue arrows).

The following definition generalizes the Bézier curves to manifolds. It makes use of the De Casteljau algorithm (Definition 3.5) and only requires Riemannian exponentials and logarithms.

3 | Interpolation with Bézier curves

Definition 3.12 (*Bézier curve on manifold*). Consider a set of control points $b_0, \dots, b_K \in \mathcal{M}$, $K \in \mathbb{N}$, such that the convex hull $\text{co}(b_0, \dots, b_K)$ is proper (see Definition 2.55).

To compute the Bézier curve $\beta_K(\cdot; b_0, \dots, b_K) : [0, 1] \rightarrow \mathcal{M}$ of order K at time t , we introduce the points $x_j^{[0]} = b_j$ and iterate the construction of further points. For $k = 1, \dots, K$ and $j = 0, \dots, K - k$, we have

$$\begin{aligned} x_j^{[k]} &:= \beta_k(t; b_j, \dots, b_{j+k}) \\ &= g(t; x_j^{[k-1]}, x_{j+1}^{[k-1]}) \\ &= \exp_{x_j^{[k-1]}} \left(t \log_{x_j^{[k-1]}} x_{j+1}^{[k-1]} \right), \end{aligned}$$

as the j^{th} point of the k^{th} step of the De Casteljau algorithm. Finally, we obtain $\beta_K(t; b_0, \dots, b_K) := x_{[K]}^0$. Note that, the exp-log formulation of x_j^k gives a geodesic between x_j^{k-1} and x_{j+1}^{k-1} , and replaces the straight line from Definition 3.5.

Example 3.13. The De Casteljau algorithm is illustrated on Fig. 3.3 for a Euclidean cubic Bézier curve $\beta_3(t; b_0, b_1, b_2, b_3)$. The general cubic Bézier curve can be explicitly expressed on a manifold \mathcal{M} as

$$\begin{aligned} \beta_3(t; x_0^{[0]}, x_1^{[0]}, x_2^{[0]}, x_3^{[0]}) &= g \left(t; g(t; g(t; x_0^{[0]}, x_1^{[0]}), g(t; x_1^{[0]}, x_2^{[0]})), \right. \\ &\quad \left. g(t; g(t; x_1^{[0]}, x_2^{[0]}), g(t; x_2^{[0]}, x_3^{[0]})) \right) \\ &= g(t; g(t; x_0^{[1]}, x_1^{[1]}), g(t; x_1^{[1]}, x_2^{[1]})) \\ &= g(t; x_0^{[2]}, x_1^{[2]}) \\ &= x_0^{[3]}. \end{aligned}$$

Remark 3.14. Another way to generalize Bézier curves is to rely on averages. Indeed, (3.2) generalizes well to

$$\beta_K(t; b_0, \dots, b_K) = \text{av}[(b_j)_{j=0, \dots, K}, (B_{jK}(t))_{j=0, \dots, K}].$$

Similarly, the geodesics used in the De Casteljau algorithm can be replaced by $x_j^{[k]} = \text{av}[(x_j^{[k-1]}, x_{j+1}^{[k-1]}), (1-t, t)]$, for $k = 1, \dots, K$ and $j = 0, \dots, K - k$.

Popiel and Noakes [PN07] have shown that properties (3.7)–(3.10) carry

over to manifolds in the following form:

$$\beta_K(0; b_0, \dots, b_K) = b_0, \quad (3.14)$$

$$\beta_K(1; b_0, \dots, b_K) = b_K, \quad (3.15)$$

$$\dot{\beta}_K(0; b_0, \dots, b_K) = K \log_{b_0}(b_1), \quad (3.16)$$

$$\dot{\beta}_K(1; b_0, \dots, b_K) = -K \log_{b_K}(b_{K-1}). \quad (3.17)$$

These properties confirm that the first and last control points of the manifold-valued Bézier curve are interpolated. They also mean that the first (resp. last) temporal derivative of the curve is tangent to the geodesic joining the two first (resp. two last) control points of the curve.

We can now generalize the composite Bézier curve to manifolds.

Definition 3.15 (*Composite Bézier curve on manifold*). Let the sequence of control points $(p_i = b_0^i, b_i^+ = b_1^i, \dots, b_{i+1}^- = b_{K_i-1}^i, p_{i+1} = b_{K_i}^i)_{i=0}^{N-1} \in \mathcal{M}$ define a sequence of N manifold-valued Bézier curves $(\beta_{K_i}^i)_{i=0}^{N-1}$ with $\beta_{K_i}^i: [0, 1] \rightarrow \mathcal{M}$ for all i . The composite Bézier curve $\mathbf{B}: [0, N] \rightarrow \mathcal{M}$ is defined analogously to the Euclidean case, *i.e.*, according to equation (3.11).

The next proposition states the conditions for a composite Bézier curve to be of class C^1 . Along with the fact that Definition 3.12 only requires exponential and logarithm maps, the simplicity of these C^1 conditions is the main motivation for resorting to composite Bézier curves in order to generalize C^1 piecewise-polynomial curves to manifolds.

Proposition 3.16 (*Continuity and differentiability*). *The composite Bézier curve composed of N segments on a manifold \mathcal{M} is continuous if $b_{K_i}^i = b_0^{i+1}$, for $i = 0, \dots, N - 2$. It will be C^1 -continuous if*

$$K_{i-1} \log_{p_i}(b_i^-) = -K_i \log_{p_i}(b_i^+), \quad i = 1, \dots, N - 1. \quad (3.18)$$

Similarly to (3.12), this condition holds if p_i is on a geodesic between b_i^+ and b_i^- and satisfies

$$p_i = g\left(\frac{K_i}{K_{i-1} + K_i}; b_i^-, b_i^+\right). \quad (3.19)$$

Example 3.17 (C^1 composite cubic Bézier curve on manifold). We consider the same composite cubic Bézier curve as in Example 3.11, but now on a Riemannian manifold \mathcal{M} . The curve is defined as $\mathbf{B}: [0, 5] \rightarrow \mathcal{M}$: $\mathbf{B}(t) =$

3 | Interpolation with Bézier curves

$\beta_3(t - i; p_i, b_i^+, b_{i+1}^-, p_{i+1})$, for $i = \lfloor t \rfloor$. Continuity is again trivial. Continuous differentiability is verified if

$$p_i = g(0.5; b_i^-, b_i^+), \quad i = 1, \dots, 4. \quad (3.20)$$

Example 3.18 (*Composite cubic Bézier curves on the sphere*). The composite cubic Bézier curve $\mathbf{B}(t): [0, n] \rightarrow \mathcal{M}$ is C^1 if $p_i = g(\frac{1}{2}; b_i^-, b_i^+)$. See Figure 3.5 for an example on $\mathcal{M} = \mathbb{S}^2$ with $n = 3$ segments.

Another option to generalize the Bézier curves is to resort to geodesic averaging, as defined in Definition 2.53. Indeed, this is a natural generalization of the Euclidean definition 3.1. This approach, however, is in practice computationally more expensive than the De Casteljau algorithm because it requires to solve an optimization problem at each evaluation of the curve β_K . Therefore, this generalization will not be used in this chapter. However, it will be extensively discussed in Chapter 6; in the same chapter, we will also discuss how to generalize the De Casteljau algorithm to the bidimensional case.

Note also that, if the De Casteljau algorithm and the averaging method provide equivalent solutions on a Euclidean space, this is generally not true anymore on general manifolds. As an example of this, we refer to Figure 3.6.

3.3 Interpolation with velocity-imposed curves

Now that Bézier curves are introduced on manifolds, it is possible to add *data points* into the equation. This section is a summary of the method communicated in [GSA14]. It is not a proper contribution of this thesis, but presenting the idea underlying this method is important to fully understand the application in Section 3.5.

Recall that we consider $n + 1$ data points d_0, \dots, d_n associated with time-parameters t_0, \dots, t_n with $t_i = i$, $i = 0, \dots, n$. In this section, we will interpolate those data points with a composite Bézier curve \mathbf{B} defined as

$$\mathbf{B}(t) = \begin{cases} \beta_2(t - i; d_i, b_{i+1}^-, d_{i+1}), & i = 0 \\ \beta_3(t - i; d_i, b_i^+, b_{i+1}^-, d_{i+1}), & i = 1, \dots, n - 2 \\ \beta_2(t - i; d_i, b_i^+, d_{i+1}), & i = n - 1. \end{cases} \quad (3.21)$$

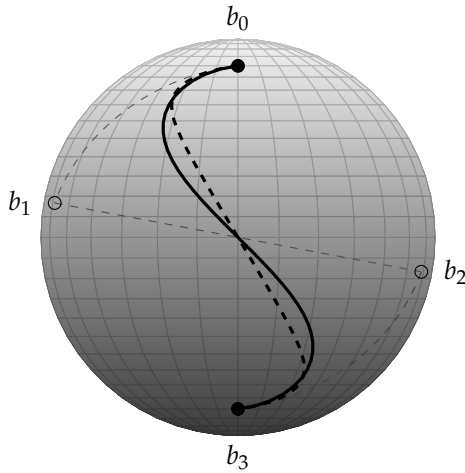


Fig. 3.6 In a Euclidean space, the averaging method is equivalent to the De Casteljau algorithm. However, they will usually not lead to identical results if they are generalized to manifolds (like the sphere). Here, the cubic Bézier curve is clearly different: the dashed curve is obtained with the weighted mean and the solid one with the De Casteljau algorithm. Interpolation and control points are indicated by filled and open circles, respectively; the Bézier polygon is shown as gray dashed lines.

Observe that the composite curve is hybrid: the first and last segments are quadratic while all the other are cubic; furthermore, we set $p_i = d_i$.

As the first and last control points of each segment are the data points, interpolation is ensured by equations (3.14) and (3.15). As a result, the optimization problem (3.1) reduces to finding the remaining control points $b_i^-, b_i^+ \in \mathcal{M}$, $i = 1, \dots, n - 1$, such that the mean square acceleration of \mathbf{B} is minimized under the differentiability conditions (3.18). This problem, however, leads to a time-consuming non-linear constrained optimization problem on manifolds.

To overcome this difficulty but still satisfying the differentiability conditions, the idea of the method is to fix the vector velocity v_i of the curve, $i = 1, \dots, n - 1$, at the intermediate interpolation points. That way, all the remaining control points are determined. The remaining task is then reduced to *choosing* those velocities. The problem is even more simplified as follows: the velocity *directions* are chosen arbitrarily (the manner to choose them is left to any suitable heuristic), such that only the velocity magni-

tudes must be optimized.

On a Euclidean space, minimizing the mean square acceleration of \mathbf{B} under those constraints permits to express those magnitudes as the solution of a tridiagonal system of linear equations. We then generalize this linear system to manifolds and use it to set the magnitudes to the velocities at the intermediate interpolation points. Even though the velocities are chosen in a suboptimal way and magnitudes are suboptimal on nonlinear manifolds, the results proposed in Section 3.5 indicate that the method produces acceptable visual results while remaining computationally tractable.

3.3.1 Computation of the control points in \mathbb{R}^m

Let us consider the Euclidean space $\mathcal{M} = \mathbb{R}^m$, and let us specify, at each intermediate data point d_i , a unitary velocity direction v_i ($\|v_i\| = 1$) chosen arbitrarily, and magnitude coefficients $\alpha_i \in \mathbb{R}$ to be optimized, for $i = 1, \dots, n - 1$. For instance, the velocities can be chosen as the unitary orientation vector of the bisector of the angle between $d_{i-1} - d_i$ and $d_{i+1} - d_i$. Namely, one can choose

$$v_i := \frac{v_i^+ - v_i^-}{\|v_i^+ - v_i^-\|}, \tag{3.22}$$

where $v_i^- := (d_{i-1} - d_i) / \|d_{i-1} - d_i\|$ and $v_i^+ := (d_{i+1} - d_i) / \|d_{i+1} - d_i\|$.

The control points associated to the hybrid composite Bézier curve are easily identified as

$$b_1^- = d_1 - \frac{3}{2}\alpha_1 v_1, \tag{3.23}$$

$$b_i^+ = d_i + \alpha_i v_i, \quad i = 1, \dots, n - 2, \tag{3.24}$$

$$b_i^- = d_i - \alpha_i v_i, \quad i = 2, \dots, n - 1, \tag{3.25}$$

$$b_{n-1}^+ = d_{n-1} + \frac{3}{2}\alpha_{n-1} v_{n-1}. \tag{3.26}$$

Observe that the differentiability conditions (3.9) and (3.10) are satisfied, since $\dot{\mathbf{B}}(i) = 3\alpha_i v_i$, $i = 1, \dots, n - 1$. The setup is represented in Figure 3.7.

As the second order covariant derivative is simply the second derivative in \mathbb{R}^m , the optimization problem (3.1) is reduced to a simple scalar optimization problem in the $n - 1$ variables α_i , $i = 1, \dots, n - 1$. Lengthy calculations yield that $x = [\alpha_1 \ \dots \ \alpha_{n-1}]^\top$ is the solution of a linear sys-

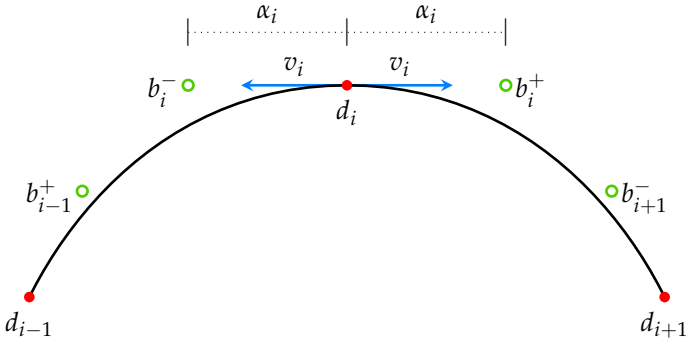


Fig. 3.7 Illustration of the construction of the hybrid composite Bézier curve $\mathbf{B}(t)$ in \mathbb{R}^2 . The path is fully determined by the velocity vectors at the intermediate data points.

tem $A \cdot x = C$, where A is a tridiagonal matrix whose entries are given (in Matlab notation) as

$$\begin{aligned} A(1, 1 : 2) &= [12v_1^\top v_1 \quad 3v_2^\top v_1] \\ A(i, i-1 : i+1) &= [v_{i-1}^\top v_i \quad 4v_i^\top v_i \quad v_{i+1}^\top v_i] \\ A(n-1, n-2 : n-1) &= [3v_{n-2}^\top v_{n-1} \quad 12v_{n-1}^\top v_{n-1}] \end{aligned}$$

and where C is given (also in Matlab notation) as

$$\begin{aligned} C(1) &= 3d_2^\top v_1 - 2d_0^\top v_1 - d_1^\top v_1 \\ C(i) &= (d_{i+1} - d_{i-1})^\top v_i \\ C(n-1) &= 2d_n^\top v_{n-1} + d_{n-1}^\top v_{n-1} - 3d_{n-2}^\top v_{n-1}. \end{aligned}$$

3.3.2 Generalization to manifolds

To generalize this result to a general manifold \mathcal{M} , the problem 3.1 is not tackled directly (it would be way too complicated). Instead, the conditions (3.23–3.26), the velocity evaluation (3.22) and the entries of the matrices A and C are adapted such that they make sense on \mathcal{M} (keeping the geometric interpretation, though).

The unit norm vectors v_i are now computed in $T_{d_i}\mathcal{M}$ thanks to the logarithmic map. Indeed, the velocities v_i^+ and v_i^- are vectors in $T_{d_i}\mathcal{M}$, as the differences $a - b$ can be interpreted as the logarithmic map $\log_b(a)$ in the

3 | Interpolation with Bézier curves

Euclidean space. Therefore, we have got that

$$v_i^- = -\frac{\log_{d_i}(d_{i-1})}{\|\log_{d_i}(d_{i-1})\|_{\mathcal{M}}}$$

and

$$v_i^+ = \frac{\log_{d_i}(d_{i+1})}{\|\log_{d_i}(d_{i+1})\|_{\mathcal{M}}}.$$

Equations of the form $b = p + \alpha v$, like in Equations (3.23–3.26), are generalized as $b = \exp_p(\alpha v)$.

Finally, in the matrices A and C , inner products $v_i^\top v_j$ are replaced by the inner product $\langle v_i, \tilde{v}_j \rangle$ in \mathcal{M} , where $\tilde{v}_j = P_{d_j \rightarrow d_i}(v_j)$. The expressions of the form $(d_j - d_i)^\top v_i$ are replaced by $\langle \log_{d_i}(d_j), v_i \rangle$. This yields

$$\begin{aligned} A(1, 1 : 2) &= [12\langle v_1, v_1 \rangle \quad 3\langle v_2, \tilde{v}_1 \rangle] \\ A(i, i - 1 : i + 1) &= [\langle v_{i-1}, \tilde{v}_i \rangle \quad 4\langle v_i, v_i \rangle \quad \langle v_{i+1}, \tilde{v}_i \rangle] \\ A(n - 1, n - 2 : n - 1) &= [3\langle v_{n-2}, \tilde{v}_{n-1} \rangle \quad 12\langle v_{n-1}, v_{n-1} \rangle] \end{aligned}$$

and

$$\begin{aligned} C(1) &= 3\langle \log_{d_1}(d_2), v_1 \rangle - 2\langle \log_{d_1}(d_0), v_1 \rangle \\ C(i) &= \langle \log_{d_i}(d_{i+1}) - \log_{d_i}(d_{i-1}), v_i \rangle \\ C(n - 1) &= 3\langle \log_{d_{n-1}}(d_{n-2}), v_{n-1} \rangle - 2\langle \log_{d_{n-1}}(d_n), v_{n-1} \rangle \end{aligned}$$

Then, this *hybrid composite Bézier curve* is reconstructed with the manifold version of the De Casteljau algorithm (Definition 3.12).

3.3.3 Differentiability of the hybrid composite curve

There is no guarantee that the coefficients α_i will remain optimal in a non-linear manifold \mathcal{M} . However, optimality holds when \mathcal{M} is Euclidean. This suggests that the results should remain qualitatively adequate on sufficiently flat manifolds. A confirmation is provided by the experiments presented in Section 3.5.

Theorem 3.19 (C^1 interpolation). *Consider the data points $d_0, \dots, d_n \in \mathcal{M}$ associated with the parameter values $t_i = i, i = 0, \dots, n$. The hybrid composite Bézier curve $\mathbf{B}(t)$ fulfills the following properties:*

differentiability
conditions

interpolation

- (i) $\mathbf{B}(i) = d_i, i = 0, \dots, n;$
- (ii) $\mathbf{B}(i)$ is differentiable at $t \in [0, n]$ if $\alpha_i < r_i$, where r_i is the injectivity radius of \mathcal{M} at d_i .

Proof. Property (i) is direct by construction of the curve (3.21). Property (ii) is ensured also by construction. Indeed, each Bézier curve composing the composite curve is differentiable on its domain. At $t = i, i = 1, \dots, n - 1$, we get differentiability by construction of the control points $b_i^+ = \exp_{d_i}(\alpha_i v_i)$ and $b_i^- = \exp_{d_i}(-\alpha_i v_i)$ and by properties (3.16–3.17). Indeed, if $\alpha_i < r_i$, then we have that $\log_{d_i}(b_i^+) = \alpha_i v_i = -\log_{d_i}(b_i^-)$ by reciprocity of the exp and log maps. \square

3.4 Interpolation with composite Bézier curves

In this section, we consider interpolation with a composite cubic Bézier curve

$$\mathbf{B}(t) = \beta_3(t - i; d_i, b_i^+, b_{i+1}^-, d_{i+1}), i = \lfloor t \rfloor. \quad (3.27)$$

The method to determine the control points of the $n - 1$ segments of Bézier is similar to [AGS⁺15] but the main difference is that all pieces are now cubic curves, such that the final composite Bézier curve is the natural cubic spline when the manifold is a Euclidean space.

As in Section 3.3, the first and last control points of each segment are the data points, so interpolation is still ensured by equations (3.14) and (3.15). The remaining degrees of freedom when building \mathbf{B} are the control points $b_i^+, b_{i+1}^- \in \mathcal{M}, i = 0, \dots, n - 1$. We face the same problem as in Section 3.3: minimizing the mean square acceleration of \mathbf{B} under the differentiability conditions (3.18) is generally unfeasible on manifolds, at least not in an acceptable computational time. Therefore, we approximate the optimal solution by proceeding as follows: we consider the case where $\mathcal{M} = \mathbb{R}^m$ and compute the optimal (Euclidean) control points $(b_i^+, b_{i+1}^-), i = 0, \dots, n - 1$, of \mathbf{B} such that (3.1) is minimized. This, of course, is an “easy” problem that can be solved in closed form.

Then, the optimality conditions on the control points are generalized to a manifold \mathcal{M} .

Since the composite cubic Bézier curve encompasses all cubic splines, the optimal control points correspond to the interpolating natural cubic

(which is known to minimize the mean squared acceleration under interpolation conditions [GS93, Theorem 2.3]), and thus satisfy the differentiability conditions of Proposition 3.9.

Of course, the resulting C^1 interpolating composite cubic Bézier curve is not guaranteed to have a minimal mean squared acceleration, even among all C^1 interpolating composite cubic Bézier curves. However, by construction, it is optimal when the manifold is Euclidean. Hence, it can be expected to be close to optimal when the manifold is “sufficiently flat”.

3.4.1 Computation of the control points

Let us consider the case $\mathcal{M} = \mathbb{R}^m$ and the data points $d_0, \dots, d_n \in \mathbb{R}^m$ at parameter-values $t_i = i, i = 0, \dots, n$. The optimization problem (3.1) becomes

$$\min_{\Gamma} \int_0^n \|\ddot{\mathbf{B}}(t)\|_2^2 dt, \quad (3.28)$$

as the covariant second derivative $\frac{D^2}{dt^2}$ is the classical time-derivative on \mathbb{R}^m , and the manifold-valued norm is the classical 2-norm. The search space Γ is reduced to the space of composite cubic Bézier curves $\Gamma_{\mathbf{B}}$ on \mathbb{R}^m that interpolate the data points, namely,

$$\Gamma_{\mathbf{B}} = \{\mathbf{B}(t) = \beta_3(t - i; d_i, b_i^+, b_{i+1}^-, d_{i+1}), i = \lfloor t \rfloor\}.$$

The remaining optimization variables are

$$\Gamma'_{\mathbf{B}} = \{b_0^+, b_1^-, \dots, b_{n-1}^+, b_n^-\}.$$

The differentiability constraints (3.18), on \mathbb{R}^m , read $b_i^+ = 2d_i - b_i^-$ (Proposition 3.9). Hence, the only remaining degrees of freedom of the Euclidean problem are $\{b_0^+, b_i^-, i = 1 \dots, n\}$, which will be our optimization variables.

In summary, the optimization problem reduces to

$$\min_{\Gamma'_{\mathbf{B}}} \int_0^n \|\ddot{\mathbf{B}}(t)\|_2^2 dt, \text{ s.t. } b_i^+ = 2d_i - b_i^-, \quad (3.29)$$

for $i = 1, \dots, n - 1$.

This problem is actually quadratic in its $n + 1$ variables, as $\mathbf{B}(t)$ depends linearly on the control points. Furthermore, it can be split into m smaller scalar problems in each of the components of \mathbb{R}^m . The optimality

conditions of (3.29) take then the form of m independent linear systems, that we formulate as $A \cdot X = C \cdot D$, where $X = [b_0^+, b_1^-, b_2^-, \dots, b_n^-]^T \in \mathbb{R}^{(n+1) \times m}$ contains the remaining optimization variables, $D = [d_0, \dots, d_n]^T \in \mathbb{R}^{(n+1) \times m}$ contains the data points of the problem, and $A, C \in \mathbb{R}^{(n+1) \times (n+1)}$ are matrices of coefficients. The details are given in Appendix A.

The solution of this linear system is given by

$$X = A^{-1} \cdot C \cdot D =: W \cdot D.$$

In other words, each control point x_i of X is obtained as a simple weighted sum of the data points $(d_j)_{j=0}^n$:

control point

$$x_i = \sum_{j=0}^n w_{ij} d_j, \quad i = 0, \dots, n. \quad (3.30)$$

To generalize this result to Riemannian manifolds, we do not solve the manifold-valued version of (3.29) directly (note that this will actually be done in Chapter 5) but we rather generalize the optimality conditions (3.30) to get a formula to compute the control points on \mathcal{M} . Note that the resulting piecewise-Bézier curve is not guaranteed to be a solution of (3.1), except if \mathcal{M} is flat (see Proposition (3.21)).

To do so, we observe that $\sum_{j=0}^n w_{ij} = 1$. This property can be deduced from the fact that the problem is invariant to translations. Indeed, one can write identically the shifted optimality conditions (3.30) as

$$x_i - d_{\text{ref}} = \sum_{j=0}^n w_{ij} (d_j - d_{\text{ref}}), \quad i = 0, \dots, n,$$

for any d_{ref} . These differences can be interpreted as Riemannian logarithms $\log_a(b) = b - a$ on the Euclidean space. In other words, one can interpret the shifting of the optimality conditions as a mapping of the point x_i to the tangent space $T_{d_{\text{ref}}} \mathcal{M}$ at d_{ref} .

reference point

Consider now a general manifold \mathcal{M} , the list of data points $d_0, \dots, d_n \in \mathcal{M}$, the search space $\Gamma_{\mathbf{B}}' \subseteq \mathcal{M}^{2n}$ and the remaining optimization variables $x_i \in \mathcal{M}, i = 0, \dots, n$. The optimality conditions (3.30) are naturally generalized as

$$\tilde{x}_i = \log_{d_{\text{ref}}}(x_i) = \sum_{j=0}^n w_{ij} \log_{d_{\text{ref}}}(d_j) \in T_{d_{\text{ref}}} \mathcal{M}, \quad (3.31)$$

for $i = 0, \dots, n$. Then, $x_i = \exp_{d_{\text{ref}}}(\tilde{x}_i)$.

The remaining task consists now in choosing a suitable reference point d_{ref} for each variable x_i . A first possible choice is to use the same reference point for all variables. However, for points far from d_{ref} , the tangent space $T_{d_{\text{ref}}}\mathcal{M}$ will generally be a bad approximation of \mathcal{M} , and the resulting curve may be of poor quality. Instead, we propose to choose a different reference point for each x_i , such that the most important data points averaged in equation (3.31) (i.e., the data points associated with the largest weights w_{ij}) are well approximated on $T_{d_{\text{ref}}}\mathcal{M}$. In the case of the control point $x_i = b_i^-$, $i = 1, \dots, n$, the largest weight is w_{ii} , so $d_{\text{ref}} = d_i$. For $x_0 = b_0^+$, the same rule is applied and $d_{\text{ref}} = d_0$.

In view of the differentiability constraints (3.18), we get the other control points as

$$b_i^+ = \exp_{d_i}(-\tilde{x}_i), \quad i = 1, \dots, n - 1, \quad (3.32)$$

and the final composite Bézier curve \mathbf{B} is reconstructed with the De Casteljau algorithm (see Definition 3.12).

The previous development results in the following definition for an interpolating composite cubic Bézier curve on manifold.

Definition 3.20 (*Interpolating C^1 composite cubic Bézier curve on manifold*). The proposed composite cubic Bézier curve $\mathbf{B}: [0, n] \rightarrow \mathcal{M}$, interpolating the manifold-valued data points d_0, \dots, d_n at parameter values $t_0 = 0, \dots, t_n = n$, is defined as

$$\mathbf{B}(t) = \beta_3(t - i; d_i, b_i^+, b_{i+1}^-, d_{i+1}), \quad i = \lfloor t \rfloor,$$

where β_3 is as in Definition 3.12. The control points b_0^+, b_i^- for $i = 1, \dots, n$ are defined by (3.31), with $d_{\text{ref}} = d_i$, while the remaining control points b_i^+ , $i = 1, \dots, n - 1$, are defined by the C^1 conditions (3.32).

The full algorithm to generate the curve of Definition 3.20 is presented in Algorithm 1.

3.4.2 Properties of the interpolating curve

As mentioned in the introduction, we seek a C^1 interpolating curve that results in a low value of the cost function of (3.1). We see here how the curve of Definition 3.20 fulfills these specifications.

Proposition 3.21 (Natural cubic spline). For $\mathcal{M} = \mathbb{R}^m$, the interpolating composite cubic Bézier curve $\mathbf{B}: [0, n] \rightarrow \mathbb{R}^m$ of Definition 3.20 is the natural cubic

Algorithm 1 Interpolating C^1 composite cubic Bézier curve approaching the solution of (3.1).

Require: $d_0, \dots, d_n \in \mathcal{M}$, A and C from Appendix A

Init: $s_0 = \dots s_n = 0$.

$W \leftarrow A^{-1}C$

% matrix of weights

for $i = 0, \dots, n$ **do**

$d_{\text{ref}} \leftarrow d_i$

% reference point

for $j = 0, \dots, n$ **do**

$s_j \leftarrow \log_{d_{\text{ref}}}(d_j)$

% mapping to $T_{d_{\text{ref}}}\mathcal{M}$

end for

$\tilde{x} \leftarrow \sum_{k=0}^n w_{ik}s_k$

% cp generation

$x \leftarrow \exp_{d_{\text{ref}}}(\tilde{x})$

% mapping to \mathcal{M}

if $i \neq 0, i \neq n$ **then**

$b_i^- \leftarrow x$

$b_i^+ \leftarrow \exp_{d_{\text{ref}}}(-\tilde{x})$

% C^1 condition (3.18)

else if $i = 0$ **then**

$b_0^+ \leftarrow x$

else $\{i = n\}$

$b_n^- \leftarrow x$

end if

end for

% De Casteljau algorithm (Definition 3.12)

$\mathbf{B}(t) = \beta_3(t - i; d_i, b_i^+, b_{i+1}^-, d_{i+1}), \quad i = \lfloor t \rfloor.$

3 | Interpolation with Bézier curves

spline that minimizes the mean square acceleration under the interpolation conditions.

Proof. By construction. \square

Theorem 3.22 (C^1 -interpolation). Consider the data points $d_0, \dots, d_n \in \mathcal{M}$ associated with parameter values $t_i = i, i = 0, \dots, n$. The composite cubic Bézier curve $\mathbf{B}(t): [0, n] \rightarrow \mathcal{M}$ of Definition 3.20 fulfills the following properties:

- (i) $\mathbf{B}(i) = d_i, i = 0, \dots, n$;
- (ii) $\mathbf{B}(t)$ is differentiable at $t \in [0, n]$ if $\|\tilde{x}_i\| < r_{d_i}, i = 1, \dots, n - 1$, where r_{d_i} is the injectivity radius of \mathcal{M} at d_i .

Proof. (i) follows from (3.14) and (3.15) (Property 3.16). Let us prove (ii). By definition, the Bézier curve is differentiable on its domain [PN07]. Therefore, there only remains to prove that the composite Bézier curve is differentiable at $t = i, i = 1, \dots, n - 1$, i.e., at the points where two consecutive segments join. Consider $i \in \{1, \dots, n - 1\}$. As $\|\tilde{x}_i\| < r_{d_i}$, we have that $\log_{d_i}(b_i^-) = \log_{d_i}(\exp_{d_i}(\tilde{x}_i)) = \tilde{x}_i = -\log_{d_i}(\exp_{d_i}(-\tilde{x}_i)) = -\log_{d_i}(b_i^+)$. By equations (3.16) and (3.17), one has

$$\left. \frac{d\mathbf{B}(t)}{dt} \right|_{t=i^-} = \left. \frac{d\mathbf{B}(t)}{dt} \right|_{t=i^+}.$$

\square

Remark 3.23. Note that the control points also satisfy the conditions given in [PN07, Thm. 3], as b_i^- and b_i^+ respect the isometry centered at d_i , by construction.

Proposition 3.24 (Minimal representation of the interpolating curve). The interpolating curve (Definition 3.20) is uniquely represented by $n + 1$ tangent vectors.

Proof. All the control points mentioned in Definition 3.20 are obtained from the $n + 1$ tangent vectors $\tilde{x}_i \in T_{d_i}\mathcal{M}, i = 0, \dots, n$, as stated in the relevant parts of Algorithm 1. \square

Proposition 3.25 (Exponential and logarithm maps required). The number of exponential and logarithm maps required to compute the interpolating curve (Definition 3.20) is

- $n(n + 1)$ logarithms for the construction of the minimal representation of the curve of Proposition 3.24;
- 8 exponential and 6 logarithm maps to reconstruct the curve $\mathbf{B}(t)$, for a given parameter value t , given that minimal representation.

Proof. The $n(n + 1)$ logarithms maps are required for the evaluation of equation (3.31). Then, the reconstruction of the curve at a given time t requires 1 exponential map for $b_{[t]}^-$, 1 exponential map for $b_{[t]}^+$, and 6 additional exponential and logarithm maps for the De Casteljau algorithm. \square

3.5 An application to transvaginal ultrasound

To illustrate the methods presented in Sections 3.3 and 3.4, we apply them to transvaginal ultrasound medical imaging as interpolation on the shape space.

In this application, we consider 2D-slices (the data points d_0, \dots, d_n) extracted from some medical imaging at a given depth in the body (these are the parameters $z_0, \dots, z_n \in \mathbb{R}$ associated to the data points). The data points lie on a space of 2D closed shapes \mathcal{S} , a manifold that is presented in the following subsections.

This section results from a collaboration with Dr. Chafik Samir (Université de Clermont, France) who provided the toolbox to compute geometric elements on the shape space \mathcal{S} . The related publication is [SGJ15].

3.5.1 Some words of motivation

The problem of surface reconstruction from 3D images has been widely studied because of its importance in medical imaging. It also has applications in computer graphics, mechanical simulations, virtual reality, and many more, particularly, to reconstruct a surface from a set of 3D point clouds [BXZ08]. For example, one can use a variational formulation using PDEs and compute the solution as an implicit surface, which is usually the zero level set of a sufficiently smooth function [ZOF01]. In that specific case, the resulting surface construction is controlled by adding physics-inspired constraints depending on geometry or external forces.

However, in practice, the observed data is acquired as 2D image slices, like in MRI where a flat image of inside the body is acquired at a certain

depth. In that case, one can often find convenient planar boundary representations of objects of interest, but a 3D reconstructed volume would be an improvement compared to four or five slices of a given anatomical object. In general, the real data slices are partially (and manually!) segmented by an expert. This is where the methods presented in this chapter make sense: one needs to perform surface reconstruction by computing an “optimal” fitting (in a sense) between the boundaries, taking into account their parameterization and the non-linearity of their spatial evolution.

Note that there exists modern reconstruction algorithms that allow the acquisition of full 3D anatomies, but the 3D surface reconstruction during the post-processing phase is often difficult because the resolution of the imaging device is often inadequate, or because the anatomical object was not perfectly placed in the scanner.

3.5.2 The shape space \mathcal{S}

In order to apply the methods proposed in this chapter to the shape space \mathcal{S} , one needs to (i) define this space, (ii) equip its tangent space with a metric, and (iii) express the formulae of the exponential map and of the logarithmic map.

The data points d_i are seen as closed planar curves in the Riemannian framework of elastic shape analysis. This framework consists in metrics and models in which shapes are invariant to arbitrary rotation, scaling, translation, and re-parameterization (*i.e.*, two rotated curves are considered equivalent, for instance). The Riemannian structure is therefore expressed as a quotient manifold, as we will see later. We refer to [SK16] for a complete analysis or to [JKS07, SKJ11] for summaries.

Consider a level-set curve $\tau : [0, 1] \rightarrow \mathbb{R}^m$. The level-set is represented by (i) its parameterization τ_i , (ii) its length l_i and (iii) its starting point $\tau(0)$. As l_i and $\tau(0)$ belong to Euclidean spaces, their interpolation is quite straightforward. The hard part of the work consists in interpolating the parameterization elements τ_i .

To achieve *rotation and translation invariance*, the parameterized level-set is represented by a function $q : [0, 1] \rightarrow \mathbb{R}^m$ as

$$q(s) = \frac{\dot{\tau}(s)}{\sqrt{\|\dot{\tau}(s)\|_{\mathbb{R}^m}}} \in \mathbb{R}^m. \quad (3.33)$$

By doing this transformation, the curve τ is now seen as a set q of local

descriptors of the geometry of the curve, which makes the representation q invariant to *rotations* and *translation*. Indeed, q is built based on the the tangent velocity vector of τ normalized by the square-root of the instantaneous speed along the curve. The representations are equivalent, as the original curve τ can be reconstructed using

$$\tau(s) = \tau(0) + \int_0^s \|q(t)\|q(t)dt.$$

Observe that, to come back from q to τ , one needs to store the starting point $\tau(0)$ of the curve τ . The representation in q is thus independent from the starting point (which implies an invariance to translations) and the curve is represented by velocity vectors (which implies invariance to rotations).

The *scale invariant* shape representation is given by normalizing the function q by its magnitude as

$$\frac{q}{\sqrt{\int_0^1 \langle q(s), q(s) \rangle ds}},$$

where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner-product in \mathbb{R}^2 . Due to this unit-scaling constraint, the space of all translation, rotations and scale-invariant shapes becomes a Hilbert sphere denoted by \mathcal{Q} [SK16, SGJ15]. Formally, the space \mathcal{Q} is defined as

$$\mathcal{Q} := \left\{ q \in \mathbb{L}^2 \text{ such that } q : [0, 1] \rightarrow \mathbb{R}^2 \text{ and } \int_0^1 \langle q(s), q(s) \rangle ds = 1 \right\}.$$

Consider now that the functions q are always normalized.

The last step is to constraint \mathcal{Q} to be invariant to re-parameterizations. This last invariance is the reason why the analysis done here is called “elastic” shape analysis. Re-parameterization gives rise to a change in speed of the curve without changing its shape. It is given by $\gamma \in \Gamma$ (not to be confounded with the geodesic defined in Chapter 2) where

$$\Gamma = \{ \gamma : [0, 1] \rightarrow [0, 1] \text{ s.t. } \gamma(0) = 0, \gamma(1) = 1, \gamma \text{ is a diffeomorphism} \}.$$

One can specify the action of the re-parameterization of a shape q by γ as $q \cdot \gamma = (q \circ \gamma)\sqrt{\dot{\gamma}}$.

Finally, the elastic shape space of open curves is defined as the quotient space $\mathcal{S}_O = \mathcal{Q}/\Gamma$. The space of elastic closed curves is simply given by im-

3 | Interpolation with Bézier curves

posing a closure constraint on the curves. In the representation in curves, the constraint reads $\int_0^1 \dot{\tau}(s) ds = 0$ and in the coordinate representation, it is $\int_0^1 q(s) \|q(s)\| ds = 0$.

Hence, the elastic shape space of closed curves is given by the quotient space

$$\mathcal{S} := \mathcal{Q}_C / \Gamma,$$

where \mathcal{Q}_C is the set \mathcal{Q} limited to the closed curves.

3.5.3 Geometric elements on the shape space

To define the geometric elements on the shape space (like the exponential map and the logarithm map), one first has to define a metric on $T_q \mathcal{S}$. Actually, the metric on $T_q \mathcal{Q}$ is sufficient (we refer to the book of Srivastava and Klassen [SK16] for a complete justification).

As \mathcal{Q} is a Hilbert sphere, its tangent space is defined as $T_q \mathcal{Q} = \{f \in \mathbb{L}^2 \text{ such that } f \perp q\}$. It is equipped with a Riemannian metric that measures infinitesimal lengths on the shape space. Given a pair of tangent vectors $f, g \in T_q \mathcal{Q}$ the metric is defined as,

$$\langle f, g \rangle_{\mathcal{S}} = \int_0^1 \langle f(s), g(s) \rangle ds. \quad (3.34)$$

Since this metric is invariant to re-parameterizations, it is acceptable for $T_q \mathcal{Q}$.

Given a shape q_1 and a tangent vector $f \in T_q \mathcal{Q}$ equipped with the metric (3.34) (and thus a norm $\|\cdot\|$), the exponential map at q_1 in the direction f is defined as [SK16]

$$q_2 = \exp_{q_1}(f) := \cos(\|f\|)q_1 + \sin(\|f\|)\frac{f}{\|f\|}.$$

Unfortunately, there is no closed form (yet) of the geodesic between two shapes q_1 and q_2 . There exists one, if the initial velocity vector $f \in T_q \mathcal{Q}$ is known, however. The formula is [SK16], for $t \in [0, 1]$,

$$g(t; q_1, q_2, f) = \cos\left(t \cos^{-1} \langle q_1, q_1 \rangle_{\mathcal{S}}\right) q_1 + \sin\left(t \cos^{-1} \langle q_2, q_2 \rangle_{\mathcal{S}}\right) f. \quad (3.35)$$

In practice, on the shape space \mathcal{S} , the geodesic is computed using a path-straightening method that initially connects the two points q_1 and q_2

using an arbitrary path in \mathcal{Q} and then iteratively straightens it to form the shortest path by minimizing a certain energy of the path [JKSJ07]. At each step of the path straightening, a re-parameterization γ^* has to be done. It is found via the minimizer

$$\gamma^* := \operatorname{argmin}_{\gamma \in \Gamma} \int_0^1 \|q_1 - \gamma \cdot q_2\|^2 ds.$$

An efficient way to find it is to solve this optimization problem with dynamic programming. The geodesic path is then given by substituting q_2 by $q_2 \cdot \gamma^*$ in (3.35).

The logarithm map is afterwards given by approximating $\dot{g}(0; q_1, q_2, f)$ with finite differences.

logarithmic
map

Note that, at this step, the logarithm map necessitates that the geodesic is computed entirely before, which leads to computational time that is barely tractable. Nonetheless, when there is no need to obtain the solution in a snap, the results presented in the next section are encouraging.

3.5.4 Results

Finally, let us apply the methods from Sections 3.3 and 3.4 to a data set on the shape space \mathcal{S} . To present a fair comparison between the two methods, the reconstructed curve \mathbf{B} will be, in both cases, the one defined in Equation (3.21). This means that the solution obtained in Section 3.4 must be slightly modified, *i.e.*, the coefficients of the matrix A and C change as given in Appendix B, and the vector of unknowns is limited to the control points b_i^- and b_i^+ for $i = 1, \dots, n$. The technique is specifically detailed in [AGS⁺15], and already applied to a shape space example.

hybrid Bézier
curve

Here, the data points are endometrial tissue surfaces obtained from MRI slices (see [SGJ15] for technical details about acquisition). In our experiments, the magnetic resonance images (MRI) have an average size of $400 \times 400 \times 5$ with a voxel resolution of $0.5 \times 0.5 \times 5 \text{ mm}^3$. An expert identified and selected the slices before extracting level-set curves. The segmentation was performed by a pelvic radiologist and then confirmed with a gynaecologist. Each planar closed curve ($d_i \in \mathcal{S}$) representing the endometrial tissue surface is associated with a constant real parameter $t_i = z_i$, the depth at which the endometrial tissue surface is imaged. Examples of extracted level-sets are shown in Figures 3.8(a) and 3.9(a).

We are interested in two examples of cylindrical surface reconstruc-

3 | Interpolation with Bézier curves

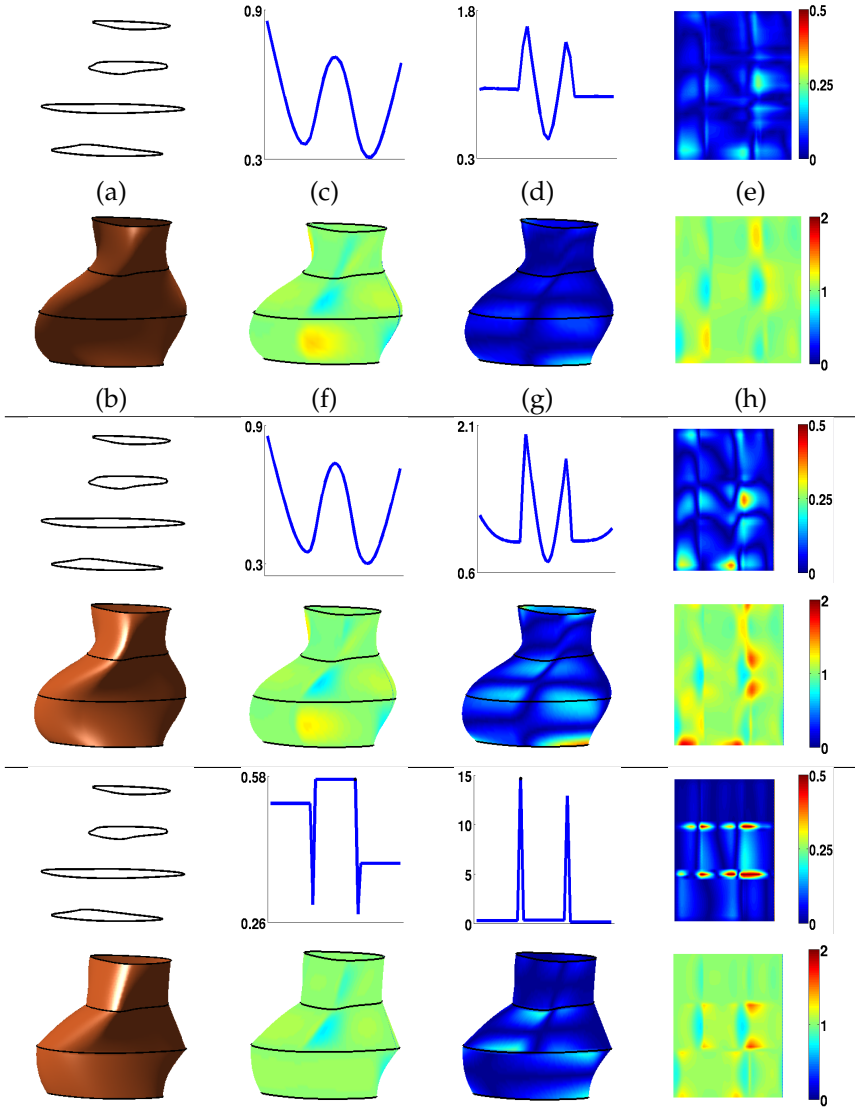


Fig. 3.8 a: original curves, b: reconstructed surface S_{MRI} , c: $\|\dot{S}_{MRI}(z)\|$, d: $\|\ddot{S}_{MRI}(z)\|$, e: $\|\nabla^2 S_{MRI}(r, \theta)\|$, f: $\|\nabla_r S_{MRI}(r, \theta)\|$, g: $\|\nabla_\theta S_{MRI}(r, \theta)\|$, and h: $\|\nabla S_{MRI}(r, \theta)\|$. The two first rows correspond to the solution from Section 3.4, the two next to the solution from Section 3.3 and the last two to a piecewise-geodesic curve.

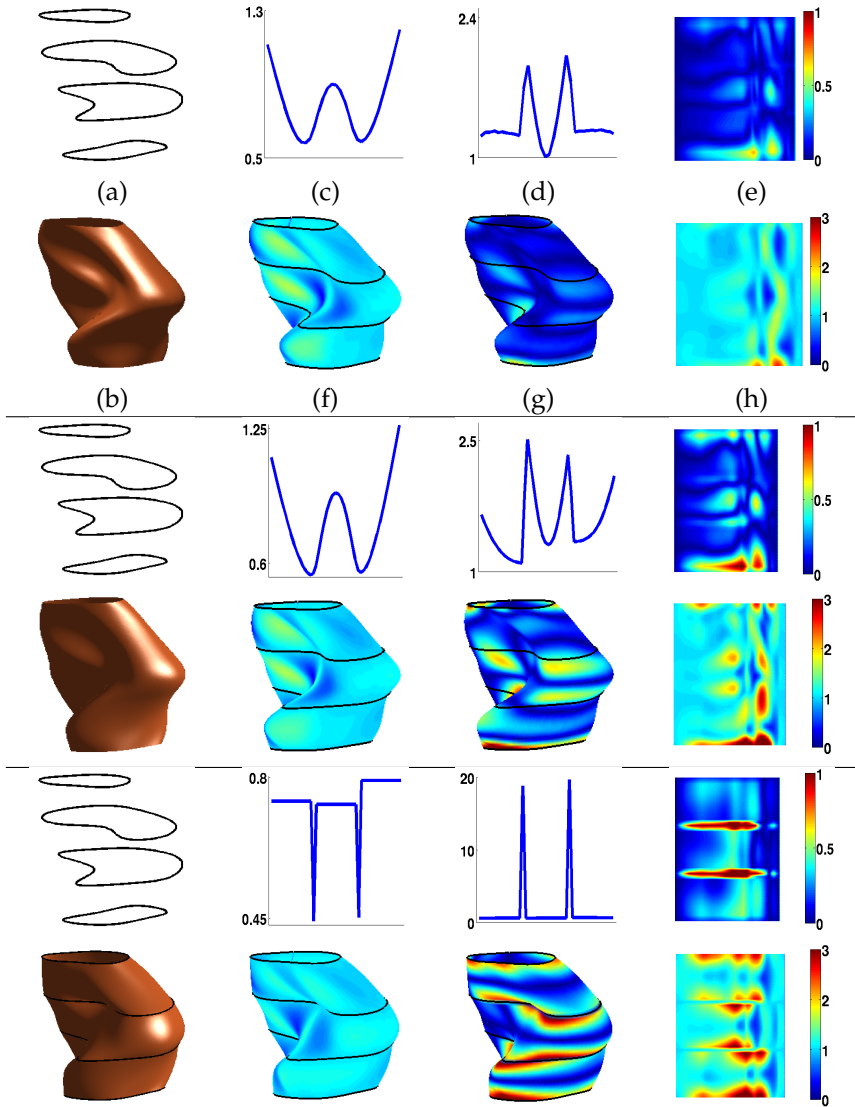


Fig. 3.9 a: original curves, b: reconstructed surface S_{MRI} , c: $\|\dot{S}_{MRI}(z)\|$, d: $\|\ddot{S}_{MRI}(z)\|$, e: $\|\nabla^2 S_{MRI}(r, \theta)\|$, f: $\|\nabla_r S_{MRI}(r, \theta)\|$, g: $\|\nabla_\theta S_{MRI}(r, \theta)\|$, and h: $\|\nabla S_{MRI}(r, \theta)\|$. The two first rows correspond to the solution from Section 3.4, the two next to the solution from Section 3.3 and the last two to a piecewise-geodesic curve.

tion. Three fitting paths are compared on the shape space: (i) the solution from Section 3.4 applied to hybrid composite Bézier curves (3.21) (as described in [AGS⁺15]), (ii) the solution from Section 3.3 and (iii) a piecewise-geodesics solution (*i.e.*, a solution obtained by computing a geodesic between each two consecutive data points).

To summarize, we reconstructed the smooth endometrial surface S_{MRI} in three steps. First, a radiologist selects different slices d_i (with $i = 0, \dots, 3$) at different depths z_i , and segments curves as boundaries of an interest zone on each slice. Second, we represent each curve as a point on the shape manifold \mathcal{S} . The starting point of each curve is aligned with the other, and the length of each curve is normalized. Third, we apply the three methods to reconstruct S_{MRI} , for each example.

In Figures 3.8 and 3.9, the solution from Section 3.4 corresponds to the first two rows, the one from Section 3.3 correspond to the rows 3 and 4, and the piecewise-geodesic curve is displayed in rows 5 and 6. To give an idea about the quality of the reconstructed surface, different outputs are given and numbered accordingly. The surfaces S_{MRI} (b) are reconstructed from a set of curves (a). The norm of the velocity of the fitting path $\|\dot{S}_{MRI}(z)\|$ is represented in (c), and the norm of its acceleration $\|\ddot{S}_{MRI}(z)\|$ is given in (d). In (e), the Laplacian map $\|\Delta S_{MRI}\|$ is displayed, and in (f), the norm of the gradient along the radial curves $\|\nabla_r S_{MRI}\|$. We also show the norm of the gradient along the circular curves $\|\nabla_\theta S_{MRI}\|$ (g), and $\|\nabla_{r,\theta} S_{MRI}\|$ as a function (2D map) of (r, θ) (h).

One can observe that the three methods are interpolating the data points, as expected. The piecewise-geodesic method is continuous but not differentiable at the data points. Moreover, its resulting path has a low velocity cost between any two original curves which is due to the geodesic connecting them (minimizing the piecewise lengths). However, at the data points, the velocity and the acceleration change drastically, which results in a big cost in the acceleration of the curve. Quite the contrary, the two other methods provide smooth paths with smaller acceleration at the data points. More specifically, they lead to roughly the same velocities but the acceleration cost is slightly smaller when the method of Section 3.4 is applied. This makes of course sense, as this method is less restricted when optimized (recall that the velocity is suboptimally imposed in Section 3.3). By the way, this method leads also to better visual reconstruction results as shown in Figure 3.8, (e) and (h); we observe less sharpness at data points (Figure 3.9, (f) and (g)).

However, in the majority of applications, data points are never acquired

perfectly, so it makes little sense to interpolate them exactly. The topic of the next chapter will concern fitting on manifolds, where data points can be approached more or less given a fitting parameter λ .

This chapter is based on the papers [GSA14] and [GMA18c] for the theoretical parts, and on [SGJ15] for Section 3.5. They are sometimes cited verbatim.

The references of these papers are

[GSA14] Pierre-Yves Gousenbourger, Chafik Samir, and P.-A. Absil. Piecewise-Bézier C^1 interpolation on Riemannian manifolds with application to 2D shape morphing. In *International Conference on Pattern Recognition (ICPR)*, pages 4086–4091, 2014. doi: 10.1109/ICPR.2014.700

[SGJ15] Chafik Samir, Pierre-Yves Gousenbourger, and Shantanu H. Joshi. Cylindrical surface reconstruction by fitting paths on shape space. In H. Drira, S. Kurtek, and P. Turaga, editor, *Proceedings of the 1st International Workshop on DIFFerential Geometry in Computer Vision for Analysis of Shapes, Images and Trajectories (DIFF-CV 2015)*, pages 11.1–11.10. BMVA Press, 2015. doi:10.5244/C.29.DIFFCV.11

and

[GMA18c] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Data fitting on manifolds with composite Bézier-like curves and blended cubic splines. *Journal of Mathematical Imaging and Vision*, 61(5):645–671, 2018. doi:10.1007/s10851-018-0865-2

The figures can be reproduced based on the code provided in the toolbox available at this link address:

<https://github.com/pgousenbourg/manint>

4

Fitting with Bézier, blended, and Bézier-like curves

THE NATURAL NEXT STEP after interpolation on manifolds is obviously fitting. Indeed, in most of the applications, data points are subject to external noise and shouldn't be interpolated. On the contrary, it is often useful to generate a fitting curve that strikes a balance between being *straight enough* (minimizing the mean squared acceleration, for instance) and passing *close enough* to the data points while *not* interpolating them exactly.

In this context, the next major contribution of this thesis consists in a generalization to fitting of the method exposed in Section 3.4. The results of this generalization have been published in [GMA18c] in collaboration with Estelle Massart, from UCLouvain. The minimization problem to solve is now

$$\min_{\mathbf{B}} \int_{t_0}^{t_n} \left\| \frac{D^2 \mathbf{B}(t)}{dt^2} \right\|_{\mathbf{B}(t)}^2 dt + \lambda \sum_{i=0}^n d^2(\mathbf{B}(t_i), d_i), \quad (4.1)$$

where \mathbf{B} is a composite curve. Four methods will be presented in this chapter. Like in Chapter 3, the data points d_0, \dots, d_n are associated with parameter values $t_0 < t_1 < \dots < t_n$.

The main result of this chapter consists in the presentation of the so-called *composite blended curve*, which combines the six desirable properties of the introduction. We recall them.

regularizer λ

- (i) As $\lambda \rightarrow \infty$, the data points are interpolated at the given times;
- (ii) The curve is of class C^1 ;
- (iii) If the manifold \mathcal{M} reduces to a Euclidean space, then the produced curve is the natural cubic spline that minimizes the energy function (4.1) over the (infinite-dimensional) Sobolev space $H^2(t_0, t_n)$;
- (iv) The only knowledge that the method requires from the manifold is the Riemannian exponential and the Riemannian logarithm;
- (v) The produced curve is represented by $\mathcal{O}(n)$ tangent vectors to the manifold (or simply points on the manifold);
- (vi) Computing $\mathbf{B}(t)$ at any t requires $\mathcal{O}(1)$ exponential and logarithm once the representation by $\mathcal{O}(n)$ tangent vectors is available.

The first section of this chapter (Section 4.1.1) consists in a naive generalization of the technique presented in Section 3.4. In this section, an analysis is also given and limitations are highlighted (namely, the interpolation property is not always met).

Section 4.2 presents a solution to these limitations with the *blending method*. It consists in building polynomial pieces by solving the optimization problem (4.1) in various tangent spaces and then blending together corresponding pieces by means of carefully chosen weights in order to satisfy all the above-mentioned properties. Though it stems from Bézier-related considerations, the method to construct the blended cubic spline fairly departs from the composite Bézier approach used for interpolation in Chapter 3. Indeed, it can be described without appealing to Bézier curves (any fitting curve on the tangent space is acceptable). Hence, the curve obtained at the limit $\lambda \rightarrow \infty$ is in general not the interpolating Bézier curve.

Other (yet less good) solutions are then presented in Section 4.3. Even if each of them fails to satisfy at least one of the properties, those are nevertheless worth considering because they are not necessarily inferior in practical applications (see Section 4.4 for numerical examples).

Like in Chapter 3, the chapter is concluded with two real-life applications. The first one concerns fitting of wind fields, a work made in collaboration with the MIT and published in [GMM⁺17]. The second one concerns parametric model order reduction, published in [MGS⁺19], with Dr. Thanh Son Nguyen (UCLouvain, Belgium). Both application result from a close collaboration with my colleague Estelle Massart (UCLouvain, Belgium) who provided me with codes to compute geometric elements on the manifolds of positive semidefinite matrices of given rank.

4.1 Fitting with composite Bézier curves

In this section, we generalize the result of Section 3.4 to the fitting problem. We seek now the C^1 composite cubic Bézier curve $\mathbf{B}: [0, n] \rightarrow \mathcal{M}$ defined as

$$\mathbf{B}(t) = \beta_3(t - i; p_i, b_i^+, b_{i+1}^-, p_{i+1}), \quad i = \lfloor t \rfloor, \quad (4.2)$$

minimizing (4.1) for $\lambda > 0$. Note that, compared to Section 3.4, the data points d_i are no longer the end-points of the Bézier curves.

Instead of solving the problem (4.1) directly and similarly to Chapter 3, we use a suboptimal route: we first determine optimality conditions on the control points of \mathbf{B} , such that (4.1) is minimized when \mathcal{M} is Euclidean. Then, we generalize these conditions to any manifold \mathcal{M} . However, we will see that this method sometimes fails to satisfy the first property we required in the introduction, namely, the fact that the curve obtained should interpolate the data points when $\lambda \rightarrow \infty$.

4.1.1 Control points generation for cubic Bézier curves

Similarly as in Section 3.4, one needs to determine the position of the $3n + 1$ control points of \mathbf{B} .

We consider the Euclidean case $\mathcal{M} = \mathbb{R}^m$ and the data points $d_i \in \mathbb{R}^m$ corresponding to parameter values $t_i = i$, $i = 0, \dots, n$. Now, we also consider $\lambda > 0$, the regularization parameter. The problem (4.1) becomes then

$$\min_{\mathbf{B} \in \Gamma'_\mathbf{B}} \int_0^n \|\ddot{\mathbf{B}}(t)\|_2^2 dt + \lambda \sum_{j=0}^n \|d_j - p_j\|_2^2, \quad (4.3)$$

where the search space is the set of control points

$$\Gamma'_\mathbf{B} = \{p_0, b_0^+, b_i^-, p_i, b_i^+, b_n^-, p_n\}_{i=1}^{n-1},$$

satisfying the differentiability constraints

$$p_i = \frac{b_i^- + b_i^+}{2}, \quad i = 1, \dots, n-1. \quad (4.4)$$

Since in \mathbb{R}^m the set of C^1 composite cubic Bézier curves encompasses the cubic splines, the optimal control points correspond to the well-known

4 | Fitting with Bézier, blended, and Bézier-like curves

cubic smoothing spline, see, *e.g.*, [GS93, Theorem 2.4].

The cost function (4.3) is a quadratic function in its control points. Following the same path as in Section 3.4, we decouple the problem into m independent problems, and we formulate these as $(A_0 + \lambda A_1) \cdot X = \lambda C \cdot D$ where

$$X = [p_0, b_0^+, b_1^-, b_1^+, \dots, b_{n-1}^+, b_n^-, p_n]^T \in \mathbb{R}^{(2n+2) \times m}$$

contains the $2n + 2$ remaining control points to optimize, stored as row vectors, and where $D = [d_0, \dots, d_n]^T \in \mathbb{R}^{(n+1) \times m}$ contains the data points. The matrices $A_0, A_1 \in \mathbb{R}^{(2n+2) \times (2n+2)}$ and $C \in \mathbb{R}^{(2n+2) \times (n+1)}$ are matrices of coefficients. They are given in Appendix C.

The Euclidean optimality conditions $X = \lambda(A_0 + \lambda A_1)^{-1} C \cdot D =: W \cdot D$ are weighted combinations of the data points, so that each optimization variable x_i of X satisfies

$$x_i = \sum_{j=0}^n w_{ij} d_j, \quad i = 0, \dots, 2n + 1, \quad (4.5)$$

where $\sum_{j=0}^n w_{ij} = 1$.

Consider now the points d_0, \dots, d_n on a Riemannian manifold \mathcal{M} , the search space $\Gamma_{\mathbf{B}}^l \subseteq \mathcal{M}^{3n+1}$ and the optimization variables $x_i \in \mathcal{M}$, $i = 0, \dots, 2n + 1$. Similarly to Section 3.4, the optimality conditions (4.5) become

$$\tilde{x}_i = \log_{d_{\text{ref}}}(x_i) = \sum_{j=0}^n w_{ij} \log_{d_{\text{ref}}}(d_j) \in T_{d_{\text{ref}}}\mathcal{M}, \quad (4.6)$$

for $i = 0, \dots, 2n + 1$. The control point x_i is finally obtained by $x_i = \exp_{d_{\text{ref}}}(\tilde{x}_i)$. To minimize distortions due to the lifting of the data on the tangent space $T_{d_{\text{ref}}}\mathcal{M}$ and back on the manifold, we choose d_{ref} as the closest data point from the control point x_i . Namely, $d_{\text{ref}} = d_j$, if $x_i = b_j^-$ or b_j^+ , $j = 1, \dots, n - 1$, and we choose $d_{\text{ref}} = d_0$ (resp. $d_{\text{ref}} = d_n$) for the case $x_i = p_0$ or $x_i = b_0^+$ (resp. b_n^- or p_n). Afterwards, one can retrieve p_i , $i = 1, \dots, n - 1$ with the differentiability constraints (3.20):

$$p_i = g(0.5; b_i^-, b_i^+), \quad i = 1, \dots, n - 1. \quad (4.7)$$

The associated composite Bézier curve $\mathbf{B}(t): [0, n] \rightarrow \mathcal{M}$ is finally reconstructed with the De Casteljau algorithm presented in Definition 3.12.

The previous reasoning leads to the following definition for a fitting

composite cubic Bézier curve on a manifold.

Definition 4.1 (Fitting C^1 composite cubic Bézier curve on manifold). For a given value of the parameter $\lambda > 0$, the composite cubic Bézier curve $\mathbf{B}: [0, n] \rightarrow \mathcal{M}$, fitting the manifold-valued data points d_0, \dots, d_n at parameter values $t_0 = 0, \dots, t_n = n$, is defined as

$$\mathbf{B}(t) = \beta_3(t - i; p_i, b_i^+, b_{i+1}^-, p_{i+1}), \quad i = \lfloor t \rfloor,$$

where β_3 is as in Definition 3.12. The control points $p_0, b_i^+, b_{i+1}^-, p_n, i = 0, \dots, n - 1$, are defined by (4.6) with $d_{\text{ref}} = d_i$, while the remaining control points $p_i, i = 1, \dots, n - 1$, are defined by (4.7).

The full algorithm to generate a fitting composite cubic Bézier curve on manifold is presented in Algorithm 2.

An example of a composite cubic Bézier curve fitting a set of data points on \mathbb{R}^2 is given in Figure 4.1a, and on the sphere S^2 in Figure 4.1b. Both results were obtained with the exact same code. The only difference was the definition of the exponential and logarithm maps. Exponential and logarithm maps on S^2 are given in Table E.1 (Appendix E).

4.1.2 Properties of the fitting composite cubic Bézier curve

We present here some properties of the fitting composite cubic Bézier curve of Definition 4.1.

Proposition 4.2 (Natural cubic spline). When $\mathcal{M} = \mathbb{R}^m$, the composite cubic Bézier curve $\mathbf{B}: [0, n] \rightarrow \mathbb{R}^m$ of Definition 4.1 is the cubic smoothing spline that minimizes (4.1) over the Sobolev space $H^2(0, n)$.

Proof. By construction. □

Theorem 4.3 (C^1 interpolation when $\lambda \rightarrow \infty$). Consider the data points $d_0, \dots, d_n \in \mathcal{M}$ associated with parameter values $t_i = i, i = 0, \dots, n$, and the control points $p_0, b_0^+, b_i^-, p_i, b_i^+, b_n^-,$ and $p_n, i = 1, \dots, n - 1$. Let $r = \inf_{a \in \mathcal{M}} r_a$ be the injectivity radius of \mathcal{M} . The composite cubic Bézier curve $\mathbf{B}(t)$ defined in Definition 4.1 satisfies the following properties:

- (i) If $\lambda \rightarrow \infty$ and $\|\tilde{x}_i\|_{\mathcal{M}} < \frac{r}{2}$, for \tilde{x}_i defined by (4.6) then $d_{\mathcal{M}}(\mathbf{B}(i), d_i) = d_{\mathcal{M}}(p_i, d_i) \rightarrow 0$;
- (ii) $\mathbf{B}(t)$ is differentiable at $t \in [0, n]$.

Fitting with a
composite
Bézier curve

natural cubic
spline

differentiability
conditions

Algorithm 2 Fitting C^1 composite cubic Bézier curve approaching the solution of (4.1).

Require: $d_0, \dots, d_n, \lambda > 0, A_0, A_1$ and C (Appendix C).

Init: $s_0 = \dots = s_n = 0$.

$W \leftarrow (A_0 + \lambda A_1)^{-1} C$ % matrix of weights

for $i = 0, \dots, n$ **do**

$d_{\text{ref}} \leftarrow d_i$ % reference point

for $j = 0, \dots, n$ **do**

$s_j \leftarrow \log_{d_{\text{ref}}}(d_j)$ % mapping to $T_{d_{\text{ref}}}\mathcal{M}$

end for

$\tilde{x} \leftarrow \sum_{k=0}^n w_{(2i)k} s_k$ % cp generation

$\tilde{y} \leftarrow \sum_{k=0}^n w_{(2i+1)k} s_k$

$x \leftarrow \exp_{d_{\text{ref}}}(\tilde{x})$ % mapping to \mathcal{M}

$y \leftarrow \exp_{d_{\text{ref}}}(\tilde{y})$

if $i \neq 0, i \neq n$ **then**

$b_i^- \leftarrow x$

$b_i^+ \leftarrow y$

$p_i \leftarrow g(0.5; x, y)$ % C^1 condition (3.20)

else if $i = 0$ **then**

$p_0 \leftarrow x$

$b_0^+ \leftarrow y$

else $\{i = n\}$

$b_n^- \leftarrow x$

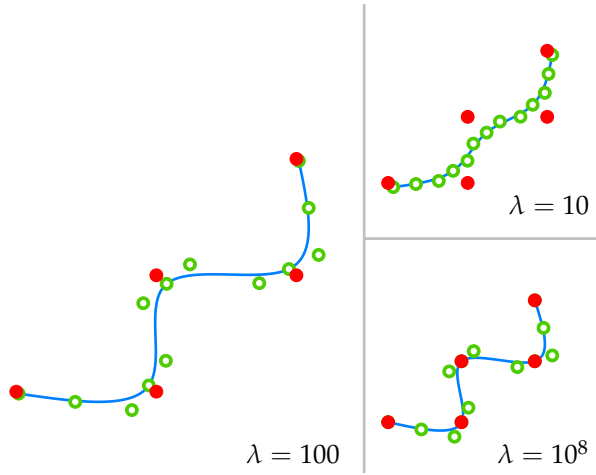
$p_n \leftarrow y$

end if

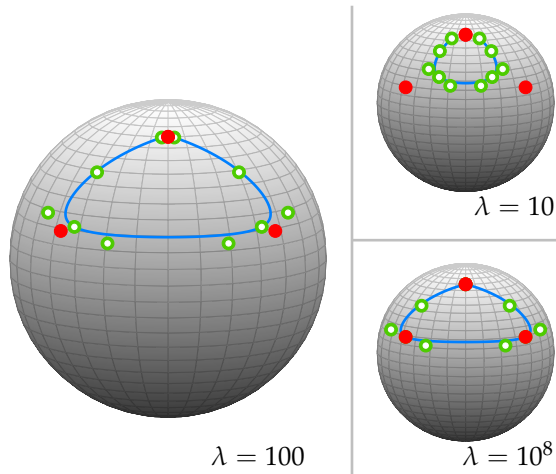
end for

% De Casteljau algorithm (Definition 3.12)

$\mathbf{B}(t) = \beta_3(t - i; p_i, b_i^+, b_{i+1}^-, p_{i+1}), \quad i = \lfloor t \rfloor$.



(a) On the Euclidean space \mathbb{R}^2



(b) On the sphere S^2 .

Fig. 4.1 Composite cubic Bézier curve fitting a set of data points on a manifold \mathcal{M} for different values of the regularization parameter λ . **Top:** on the Euclidean space \mathbb{R}^2 . **Bottom:** on the sphere S^2 . The (solid red) data points are fitted by the (blue) curve defined by the (circled green) control points. Fitting becomes interpolation when $\lambda \rightarrow \infty$.

Proof. By the standing assumption from the introduction, the data points are such that the exponential and the logarithm are well defined. We show property (ii) by construction as $\log_{p_i}(b_i^+) = -\log_{p_i}(b_i^-)$. Thus, we just have to verify property (i). By hypothesis, b_i^- and b_i^+ both lie in the set $\mathcal{D}_{d_i}(\frac{r}{2}) := \{y \in \mathcal{M} : d_{\mathcal{M}}(d_i, y) < \frac{r}{2}\}$. Due to the triangle inequality, $d_{\mathcal{M}}(b_i^-, b_i^+) \leq d_{\mathcal{M}}(d_i, b_i^-) + d_{\mathcal{M}}(d_i, b_i^+) < r$. Hence $(b_i^-, b_i^+) \mapsto \exp_{b_i^-}(\frac{1}{2} \log_{b_i^-}(b_i^+))$ is continuous. Since interpolation holds on the Euclidean case when $\lambda \rightarrow \infty$, we have that $\lim_{\lambda \rightarrow \infty} \tilde{x}_i = -\lim_{\lambda \rightarrow \infty} \tilde{y}_i$ (see Algorithm 2). Hence, d_i is the midpoint of the minimizing geodesic between $x := \lim_{\lambda \rightarrow \infty} b_i^-$ and $y := \lim_{\lambda \rightarrow \infty} b_i^+$, i.e., $d_i = \exp_x(\frac{1}{2} \log_x(y))$. It follows that $\lim_{\lambda \rightarrow \infty} p_i = d_i$. \square

This theorem shows that, when $\lambda \rightarrow \infty$, the composite cubic Bézier curve \mathbf{B} of Definition 4.1 interpolates the data points *if they are not too spread out*, since $\|\tilde{x}_i\|_{\mathcal{M}}$ is then small for all i . Otherwise, interpolation as $\lambda \rightarrow \infty$ may not hold, and we show some examples in the next section.

4.1.3 Lack of interpolation when $\lambda \rightarrow \infty$

A first illustration of the limitation suggested by Theorem 4.3(i) can be observed on $\mathcal{S}_+(p, q)$, the manifold of positive semidefinite matrices of size p and rank q , equipped with the metric from [VAV09, §7.2]. A second illustration will be also given on the unit circle \mathbb{S}^1 .

Example 4.4 (*Lack of interpolation on $\mathcal{S}_+(p, q)$*). The $\mathcal{S}_+(p, q)$ manifold arises in the development of efficient and safe navigating tools for UAV's (unmanned aerial vehicles). These navigating tools rely on faithful models for the wind field.

For this example, we consider 33 data points $C_i, i = 1, \dots, 33$ on the manifold $\mathcal{S}_+(3024, 20)$, and associated to parameters $\theta_i = (i - 1)\pi/64$. The data points are covariance matrices that represent the wind field, extracted from [GMM⁺17]; the parameter corresponds to the orientation of the prevalent wind in an area of interest (see Section 4.5 for a detailed description of the application of $\mathcal{S}_+(p, q)$ to wind fields). The manifold $\mathcal{S}_+(p, q)$ is here seen as a quotient manifold $\mathbb{R}_*^{p \times q} / \mathcal{O}(q)$, where $\mathbb{R}_*^{p \times q}$ is the set of full rank matrices of size $p \times q$, and $\mathcal{O}(q)$ is the manifold of orthogonal matrices of size $q \times q$. This quotient manifold is endowed with the metric proposed in [VAV09, §7.2]; the geometric elements used for this ex-

ample are the result of the work of Dr. Estelle Massart [MA18, MHA19] (actually, the example itself was done in collaboration with her).

Applying Algorithm 2 with $\lambda = 10^8$ to those data, we observe in Figure 4.2 that the composite cubic Bézier curve $\mathbf{B}(\theta)$ does not interpolate one of the data points, the interpolation error at that point being several orders of magnitude higher than at the other points. The reason of this in-

interpolation

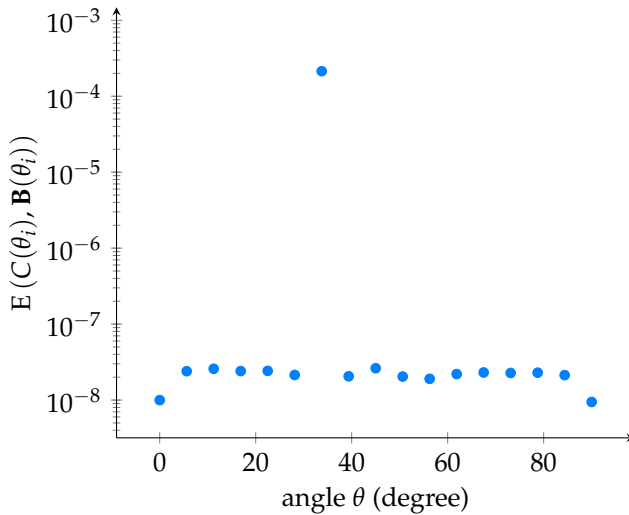


Fig. 4.2 Interpolation error between the curve $\mathbf{B}(\theta)$ and the data points $C_1, C_3, C_5, \dots, C_{33} \in \mathcal{S}_+(3024, 20)$ extracted from [GMM⁺17]. The parameter λ is set to 10^8 . We observe that the error made on the data point C_{13} is several orders of magnitude higher than the error made on the others.

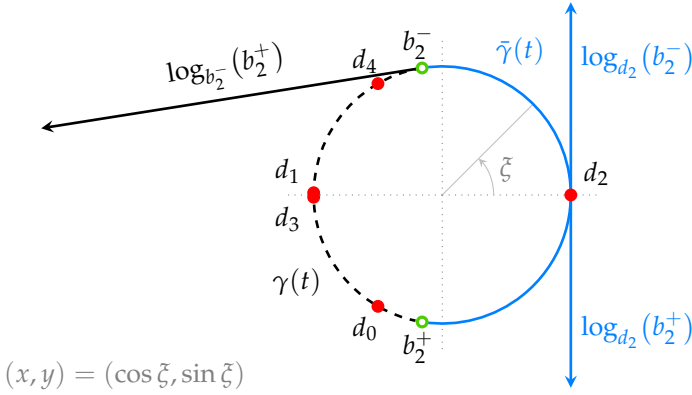
terpolation error is directly related to Theorem 4.3 whose hypotheses are not met. Indeed, there is no guarantee that $\|\tilde{x}_i\|_{\mathcal{M}} < \frac{r}{2}$. Actually, we can even show that, for the metric considered here, the local injectivity radius around a point $a \in \mathcal{S}_+(p, q)$ is equal to the square root of the smallest non-zero eigenvalue of a , which means that it tends towards zero as a tends towards the border of the manifold.

injectivity radius

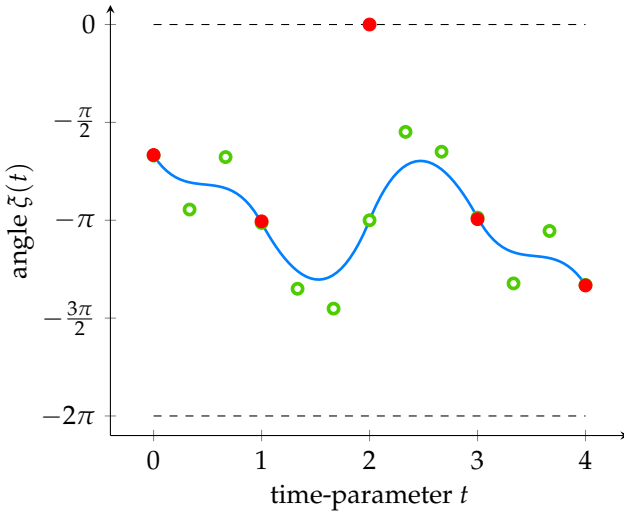
The next example deals with the circle S^1 , a manifold whose injectivity radius is equal to π . We provide a configuration of data points that violates the assumptions of Theorem 4.3, and for which interpolation is indeed not achieved.

Example 4.5 (*Lack of interpolation in S^1*). Consider the set of data points

sphere S^1



(a) Representation of the data set used for the counter-example on the circle, as well as the two curves γ and $\tilde{\gamma}$, for $i = 2$. The injectivity radius of the circle S^1 is π . As the geodesic length $L_{\tilde{\gamma}} = |\log_{d_2}(b_2^-)| + |\log_{d_2}(b_2^+)| > \pi$, the geodesic $\tilde{\gamma}$ is not minimizing, while $\gamma = g(t; b_i^-, b_i^+)$ is. The consequence of this is that $p_i \neq d_i$.



(b) The curve $\xi(t) = \tan^{-1}(\mathbf{B}(t))$ (solid blue) obtained by applying Algorithm 2 to the (solid red) data points, with $\lambda = 10^8$, does not interpolate all data points. Specifically, $\xi(2) \neq \tan^{-1}(d_2)$.

Fig. 4.3 Second counter-example for interpolation by the fitting curve of Definition 4.1, when $\lambda \rightarrow \infty$.

d_0, \dots, d_4 represented on Figure 4.3a. The position of the data points d_0 and d_4 has been chosen respectively on the lower half circle and the upper half circle (here, at $\zeta_0 = -120^\circ$ and $\zeta_4 = 120^\circ$). The data point d_2 corresponds to an angle $\zeta_2 = 0^\circ$, while data points d_1 and d_3 correspond respectively to angles $\zeta_1 = 179^\circ$ and $\zeta_3 = 181^\circ$.

On Figure 4.3b, we represent the curve obtained when we try to interpolate the set of data points from Figure 4.3a with Algorithm 2. At time $t = 2$, the angle ζ is equal to $-\pi$, instead of the desired value 0. Indeed, following (3.20), $p_2 = -\pi$ is the midpoint of the shortest endpoint geodesic between the control points b_2^- and b_2^+ . Unfortunately, as indicated on Figure 4.3a, $d_2 = 0$ is the midpoint of a non-minimizing geodesic $\tilde{\gamma}$ between b_2^- and b_2^+ (because $\|\log_{d_2}(b_2^-)\| > \frac{r}{2}$, as well as $\|\log_{d_2}(b_2^+)\|$), while the midpoint of the shortest geodesic γ between the two data points is given by $\zeta = -\pi$. Therefore, d_2 is not interpolated by $\mathbf{B}(t)$.

cut locus

interpolation

4.1.4 Alternative definition of the control points to ensure interpolation while losing differentiability

To overcome this limitation for any general manifold, we propose now a solution that enforces the fitting curve to interpolate the data points when $\lambda \rightarrow \infty$, regardless of the curvature of the manifold. This new condition will fix the interpolation problem, but we will see that this conditions yields to a loss of the differentiability condition that we had before. As a consequence of that, it will be necessary to modify the De Casteljau algorithm (usually used at the reconstruction step) to propose a new definition of Bézier curves, compatible with the new C^1 -condition (see Sections 4.2 and 4.3).

The solution to have interpolation modifies the definition of the intermediary control points $p_i, i = 1, \dots, n - 1$, as follows.

Proposition 4.6 (Interpolation conditions). *Let us consider the tangent vectors \tilde{b}_i^+ and $\tilde{b}_i^- \in T_{d_i}\mathcal{M}, i = 1, \dots, n - 1$, computed by (4.6). To interpolate the data points d_i when $\lambda \rightarrow \infty$, the control points p_i , can be computed as (see Figure 4.4)*

$$p_i = \exp_{d_i} \left(\frac{\tilde{b}_i^- + \tilde{b}_i^+}{2} \right), i = 1, \dots, n - 1. \quad (4.8)$$

Proposition 4.6 is nothing more than another way to generalize the Eu-

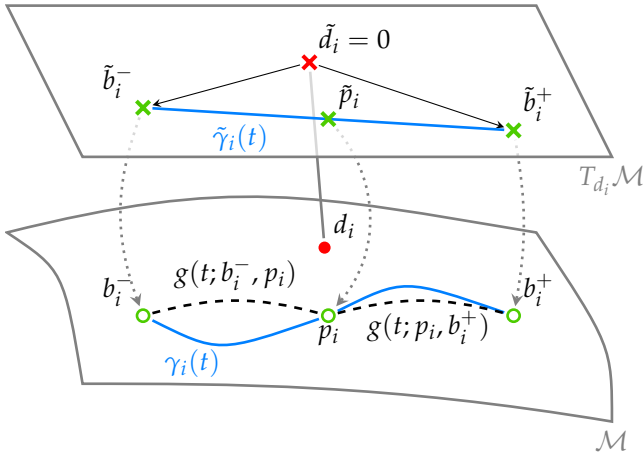


Fig. 4.4 Choice of the point p_i with the interpolation conditions (4.8). As $\tilde{p}_i \in T_{d_i}\mathcal{M}$ is in the middle of the straight line $\tilde{\gamma}_i(t)$ joining \tilde{b}_i^- and \tilde{b}_i^+ , $\lim_{\lambda \rightarrow \infty} p_i = d_i$. Furthermore, $\gamma_i(t) = \exp_{d_i}(\tilde{\gamma}_i(t))$ is differentiable at p_i even if b_i^-, p_i and b_i^+ are not aligned on the same geodesic.

clidean conditions of differentiability (3.12). The two resulting definitions for the control points $p_i, i = 1, \dots, n - 1$, are equivalent on the Euclidean space, while they are not on a general Riemannian manifold. The difference compared to (4.7) is that here the C^1 condition is directly computed in the tangent space at d_i , and not at b_i^- .

Applying these new interpolation conditions results in the following definition for the fitting composite cubic Bézier curve.

Definition 4.7 (Fitting composite cubic Bézier curve on manifold-II). For a given value of the parameter $\lambda > 0$, the composite cubic Bézier curve $\mathbf{B}: [0, n] \rightarrow \mathcal{M}$, fitting the manifold-valued data points d_0, \dots, d_n at parameter values $t_0 = 0, \dots, t_n = n$, is defined as

$$\mathbf{B}(t) = \beta_3(t - i; p_i, b_i^+, b_{i+1}^-, p_{i+1}), \quad i = \lfloor t \rfloor.$$

The control points $p_0, b_i^+, b_{i+1}^-, p_n, i = 0, \dots, n - 1$, are defined by (4.6) with $d_{\text{ref}} = d_i$. The remaining control points $p_i, i = 1, \dots, n - 1$, are computed as in Proposition 4.6. The Bézier curve β_3 is reconstructed according to Definition 3.12.

We now list a set of properties that are verified by the fitting curve of Definition 4.7.

Proposition 4.8. *The composite cubic Bézier curve $\mathbf{B}(t)$ of Definition 4.7 interpolates the data points when $\lambda \rightarrow \infty$.*

interpolation

Proof. We show here that $\mathbf{B}(i) \rightarrow d_i, i = 0, \dots, n$, when $\lambda \rightarrow \infty$. Let $i \in \{1, \dots, n - 1\}$. When $\lambda \rightarrow \infty$, the weights w_{ij} in equation (4.6) are such that $\lim_{\lambda \rightarrow \infty} \tilde{b}_i^+ = -\lim_{\lambda \rightarrow \infty} \tilde{b}_i^-$. Therefore, $\lim_{\lambda \rightarrow \infty} \tilde{p}_i = 0 = \tilde{d}_i$, and thus $\lim_{\lambda \rightarrow \infty} p_i = d_i$. For $i \in \{0, n\}$, interpolation is ensured by construction. \square

Proposition 4.9 (Minimal representation of the fitting curve). *The fitting curve of Definition 4.7 is uniquely represented by $2(n + 1)$ tangent vectors.*

minimal representation

Proof. The proof is similar to the one of Proposition 3.24. The control points are obtained from the $2(n + 1)$ tangent vectors $\tilde{x}, \tilde{y} \in T_{d_i}\mathcal{M}, i = 0, \dots, n$, as stated in the relevant parts of Algorithm 2. \square

Proposition 4.10 (Exponential and logarithm maps required). *The number of exponential and logarithm maps required by the fitting curve of Definition 4.7 is*

Exp-Log complexity

- $n(n + 1)$ logarithms for the construction of the minimal representation of the curve of Proposition 4.9
- 10 exponential maps and 6 logarithm maps to reconstruct the curve $\mathbf{B}(t)$, for a given parameter value t , given that minimal representation.

Proof. The $n(n + 1)$ logarithms maps are required for the evaluation of (4.6). The reconstruction of the curve at a given value of t require 4 exponential maps (to obtain the representation on the manifold of the 4 control points associated to the segment $i = \lfloor t \rfloor$), and the De Casteljaun algorithm requires 6 additional exponential and logarithm maps. \square

Proposition 4.11. *The composite cubic Bézier curve of Definition 4.7 is in general not differentiable at p_i .*

Proof. Consider a set of data points $d_i \in \mathbb{R}^2, i = 0, \dots, n$, and the corresponding parameter values $t_0 = 0, \dots, t_n = n$. Let \mathbf{B} be the fitting curve (Definition 4.7), for a given value of λ . Without loss of generality, we fix i , and consider that the associated data point d_i is equal to $(0, 0)$. Assume now that the data points belong actually to a manifold \mathcal{M} similar to \mathbb{R}^2 . More precisely, let \mathcal{M} be the Euclidean space \mathbb{R}^2 except in a small

region between b_i^- and b_i^+ where the metric is smoothly reduced such that $d_{\mathcal{M}}(b_i^-, b_i^+) < d_{\mathbb{R}^2}(b_i^-, b_i^+) = \|b_i^+ - b_i^-\|$, as represented in Figure 4.5. Finally, let us denote by \tilde{x} the representation in $T_{d_i}\mathcal{M} = \mathbb{R}^2$ of a point $x \in \mathcal{M}$.

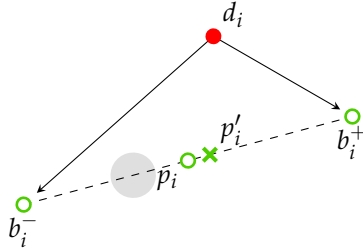


Fig. 4.5 Proposition 4.6 is not compatible with the differentiability constraints, as the middle of the geodesic between b_i^- and b_i^+ is p_i' and not p_i .

Proposition 4.6 states that $\tilde{p}_i := \log_{d_i}(p_i)$ is obtained as $\tilde{p}_i = 0.5(\tilde{b}_i^- + \tilde{b}_i^+)$. As $d_i = (0,0)$, we observe immediately that $b_i^+ = \tilde{b}_i^+$, as well as $b_i^- = \tilde{b}_i^-$. However, $p_i = 0.5(b_i^- + b_i^+)$ does not correspond to the midpoint $p_i' = g(0.5; b_i^-, b_i^+)$ of a (possibly non-minimizing) geodesic g between b_i^- and b_i^+ . This is due to the shrinking of the metric in the above-mentioned area. Therefore, the two velocities $\lim_{\epsilon \rightarrow 0} \frac{d}{dt} \mathbf{B}(i - \epsilon) = -\log_{p_i}(b_i^-)$ and $\lim_{\epsilon \rightarrow 0} \frac{d}{dt} \mathbf{B}(i + \epsilon) = \log_{p_i}(b_i^+)$ are in general different in $T_{p_i}\mathcal{M}$, which results in the non-differentiability of the curve. \square

On the one hand, the new fitting curve of Definition 4.7 recovers the interpolation property as $\lambda \rightarrow \infty$, but on the other hand, losing differentiability is a major drawback.

In the next section, we propose a new curve definition, obtained using a so-called blending process, in order to overcome this drawback.

4.2 Fitting with blended Bézier curves

We now present a way to construct a *differentiable* curve $\mathbf{B}(t)$ that satisfies the six properties mentioned at the beginning of this chapter. The method is also less costly than those of Section 4.1, as it requires fewer exp and

log evaluations to evaluate the curve at a given parameter value t . We still resort to Bézier curves, but here it is merely for convenience in order to reuse (4.6); actually all the Bézier curve are now computed in (Euclidean) tangent spaces and are thus nothing else than polynomials expressed in a specific form.

4.2.1 Blending technique

The basic idea of this method is that the control points of the i^{th} curve of $\mathbf{B}(t)$ are computed in the tangent spaces of the data points d_i and d_{i+1} . A (Euclidean) Bézier curve is computed on each of these tangent spaces and then mapped to the manifold. These two solutions are finally blended together on \mathcal{M} , via a carefully chosen weighted mean that will be defined in a moment. This last step of this sketched algorithm gives its name to the reconstructed *blended* curve \mathbf{B} . We provide now a more formal description of this method.

Definition 4.12 (*Blended curve*). Let $d_{\text{ref},1}, d_{\text{ref},2} \in \mathcal{M}$ be two different reference points. Consider two sequences of tangent vectors $\tilde{b}_0, \dots, \tilde{b}_K \in T_{d_{\text{ref},1}}\mathcal{M}$ and $\hat{b}_0, \dots, \hat{b}_K \in T_{d_{\text{ref},2}}\mathcal{M}$ named *control vectors*. The *blended curve* $\beta_K: [0, 1] \rightarrow \mathcal{M}$ of degree K is defined as

$$\beta_K(t) = \text{av}[(L(t), R(t)), (1 - w(t), w(t))], \tag{4.9}$$

where $L(t)$ (resp. $R(t)$) is the *local curve* at the reference point $d_{\text{ref},1}$ (resp. $d_{\text{ref},2}$). They are expressed as

$$L(t) = \exp_{d_{\text{ref},1}}(\tilde{\beta}_K(t; \tilde{b}_0, \dots, \tilde{b}_K)) \tag{4.10}$$

$$R(t) = \exp_{d_{\text{ref},2}}(\hat{\beta}_K(t; \hat{b}_0, \dots, \hat{b}_K)) \tag{4.11}$$

and where $w(t) = 3t^2 - 2t^3$ is a given *weight function*. The curve $\tilde{\beta}_K$ (resp. $\hat{\beta}_K$) is called the *tangent curve* in $T_{d_{\text{ref},1}}\mathcal{M}$ (resp. $T_{d_{\text{ref},2}}\mathcal{M}$).

Remark 4.13. In order to obtain the differentiability property of Theorem 4.18, the weight function $w(t)$ has to be chosen such that $\beta_K(0) = L(0)$, $\beta_K(1) = R(1)$, $\dot{\beta}_K(0) = \dot{L}(0)$, and $\dot{\beta}_K(1) = \dot{R}(1)$. This is achieved when $w(0) = 0$, $w(1) = 1$, $w'(0) = 0$, and $w'(1) = 0$. Among all functions w that satisfy these conditions, we opted for the one that minimizes the mean square second derivative $\int_0^1 (w''(t))^2 dt$. This choice of weight can also be

reference point
control vector
blended curve

local curve

tangent curve
 $\tilde{\beta}_K$

weight
function $w(t)$

seen as the minimizing one when then pieces $L(t) \in \mathbb{R}$ and $R(t) \in \mathbb{R}$ are constant, such that the resulting curve \mathbf{B} now minimizes $\int_0^1 \|\mathbf{B}''(t)\|^2 dt$. Remark also that $w(t) \in [0, 1]$, for any $t \in [0, 1]$. The weighting function is represented in figure 4.6

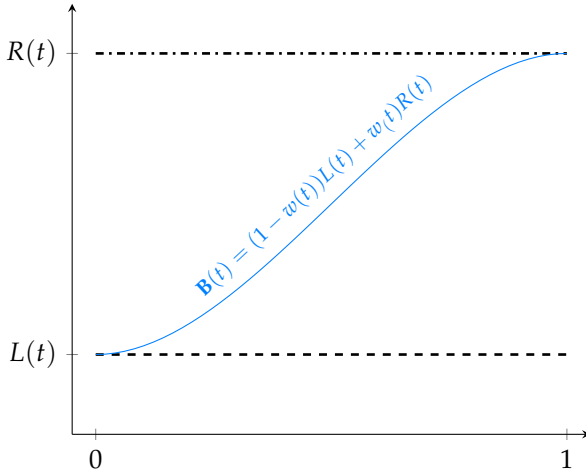


Fig. 4.6 Optimal blending of two constant curves $L(t), R(t)$ in \mathbb{R} .

Proposition 4.14 (Continuity of the blended curve). *The blended curve (Definition 4.12) is well defined and smooth under the assumption that the exponential and the logarithm map involved are well defined and smooth. This holds in particular if \mathcal{M} is geodesically complete and if the distance $d_{\mathcal{M}}(L(t), R(t)) < r^*(t)$, where $r^*(t) = \max(r_{R(t)}, r_{L(t)})$, for all $t \in [0, 1]$. Furthermore, it interpolates the point $b_0 = \exp_{d_{\text{ref},1}}(\tilde{b}_0)$ and the point $b_K = \exp_{d_{\text{ref},2}}(\hat{b}_K)$.*

Proof. The first claim is direct by composition. For the second claim, as $d_{\mathcal{M}}(L(t), R(t)) < r^*(t)$, the weighted average between $L(t)$ and $R(t)$ never crosses the cut loci of the points, so that the blended curve is never discontinuous. Interpolation is obtained as follows. As $w(0) = 0$, then $\beta_K(0) = \text{av}[(L(0), R(0)), (1, 0)] = L(0) = b_0$, and similarly for $\beta_K(1)$, as $w(1) = 1$. □

The natural next step is now to define the notion of *composite blended curve*.

Definition 4.15 (*Composite blended curve*). Consider now the sequence $(\beta_{K_i}^i)_{i=0}^{n-1}$ of n blended curves. The *composite blended curve* $\mathbf{B}(t)$ is the curve

$$\mathbf{B}: [0, n] \rightarrow \mathcal{M}, t \mapsto \beta_{K_i}^i(t - i), \quad i = \lfloor t \rfloor. \quad (4.12)$$

Consider now the data points $d_0, \dots, d_n \in \mathcal{M}$ corresponding to parameter values $t_i = i, i = 0, \dots, n$, and the regularization parameter $\lambda > 0$. Our objective is now to define the *blended cubic spline* $\mathbf{B}(t) = \beta_3^i(t - i), i = \lfloor t \rfloor$ minimizing (4.1) when \mathcal{M} is flat. To do so, one has to choose the two reference points and find the control vectors of each blended curve.

Without loss of generality, consider the i^{th} blended curve $\beta_3^i(t)$ of $\mathbf{B}(t)$. We define its intermediate functions $L(t)$ and $R(t)$ as

$$L(t) = \exp_{d_i} \left(\tilde{\beta}_K(t; \tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-, \tilde{p}_{i+1}) \right) \quad (4.13)$$

$$R(t) = \exp_{d_{i+1}} \left(\hat{\beta}_K(t; \hat{p}_i, \hat{b}_i^+, \hat{b}_{i+1}^-, \hat{p}_{i+1}) \right), \quad (4.14)$$

where we have chosen $d_{\text{ref},1} = d_i$ and $d_{\text{ref},2} = d_{i+1}$.

The control vectors of the Euclidean Bézier curves on $T_{d_i}\mathcal{M}$ and $T_{d_{i+1}}\mathcal{M}$ are obtained such that they are cubic smoothing splines on their respective tangent spaces. In other words, the control vectors $\tilde{x} \in \{\tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-, \tilde{p}_{i+1}\}$ (resp. \hat{x}) correspond to the tangent vectors computed using equation (4.6) with $d_{\text{ref}} = d_i$ (resp. $d_{\text{ref}} = d_{i+1}$) on $T_{d_i}\mathcal{M}$ (resp. $T_{d_{i+1}}\mathcal{M}$). Finally, in view of Proposition 3.9 and the C^1 (actually C^2) property of a natural cubic spline, the remaining control vectors are

$$\tilde{p}_i = \frac{\tilde{b}_i^- + \tilde{b}_i^+}{2}, \quad i = 1, \dots, n-1, \quad (4.15)$$

and accordingly for \hat{p}_i .

We can now define the *blended cubic spline*.

Definition 4.16 (*Blended cubic spline*). For a given value of the parameter $\lambda > 0$, the proposed *blended cubic spline* $\mathbf{B}: [0, n] \rightarrow \mathcal{M}$ fitting the manifold-valued data points $d_0, \dots, d_n \in \mathcal{M}$ at parameter values $t_0 = 0, \dots, t_n = n$, is defined as

$$\mathbf{B}(t) := \beta_3^i(t - i), \quad i = \lfloor t \rfloor,$$

where $\beta_3^i(t - i)$ is as in Definition 4.12. The intermediate functions $L(t)$ and $R(t)$ of β_3^i are defined by (4.13) and (4.14), and their control vectors are computed on $T_{d_i}\mathcal{M}$ and $T_{d_{i+1}}\mathcal{M}$ using (4.6) and (4.15).

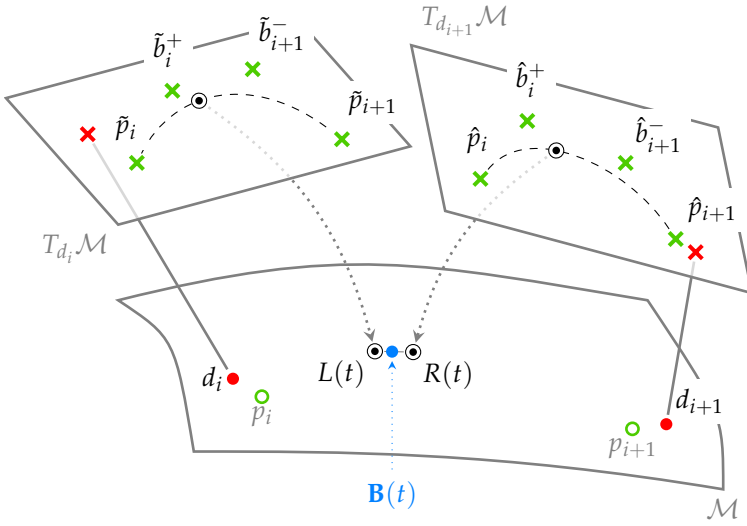


Fig. 4.7 Illustration of the reconstruction of the blended cubic spline (Algorithm 4). The points $(\tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-, \tilde{p}_{i+1})$ and $(\hat{p}_i, \hat{b}_i^+, \hat{b}_{i+1}^-, \hat{p}_{i+1})$ are computed respectively in the tangent space of d_i and d_{i+1} according to (4.6) and (4.15). The points $L(t)$ and $R(t)$ (black circles) are obtained as the mapping to \mathcal{M} of the Euclidean Bézier curve obtained on $T_{d_i}\mathcal{M}$ and $T_{d_{i+1}}\mathcal{M}$ respectively. Finally, these points are averaged on \mathcal{M} via equation (4.9) to obtain the value of the blended cubic spline $\mathbf{B}(t)$ (blue dot).

The whole method is summarized in Algorithms 3 and 4, and this last algorithm is represented on Figure 4.7.

4.2.2 Properties of the composite cubic blended curve

We now analyze the properties of the composite cubic blended curve.

Lemma 4.17. *Consider the tangent spaces $T_{d_i}\mathcal{M}$, $i = 1, \dots, n - 1$ and the control vectors \hat{p}_{i-1} , \hat{b}_{i-1}^+ , \hat{b}_i^- , $\hat{p}_i = \tilde{p}_i$, \tilde{b}_i^+ , \tilde{b}_{i+1}^- , and \tilde{p}_{i+1} obtained by (4.6) and (4.15). The curve $\tilde{\gamma}: [0, 2] \rightarrow T_{d_i}\mathcal{M}: t \mapsto \tilde{\gamma}(t)$ given by*

$$\tilde{\gamma}(t) = \begin{cases} \beta_3(t; \hat{p}_{i-1}, \hat{b}_{i-1}^+, \hat{b}_i^-, \hat{p}_i) & \text{for } t \in [0, 1] \\ \beta_3(t - 1; \tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-, \tilde{p}_{i+1}) & \text{for } t \in [1, 2], \end{cases}$$

is a natural cubic spline on $T_{d_i}\mathcal{M}$.

Algorithm 3 Control points generation for the blended cubic spline of Definition 4.16.

Require: $d_0, \dots, d_n, \lambda > 0, A_0, A_1$ and C (Appendix C).

Init: $s_0 = \dots = s_n = u_0 = \dots = u_n = 0$.

$W \leftarrow (A_0 + \lambda A_1)^{-1} C$ % matrix of weights

$d_{\text{ref},2} = d_0$.

for $j = 0, \dots, n$ **do**

$u_j = \log_{d_{\text{ref},2}}(d_j)$

end for

for $i = 0, \dots, n - 1$ **do**

$d_{\text{ref},1} \leftarrow d_i$ % reference points

$d_{\text{ref},2} \leftarrow d_{i+1}$

for $j = 0, \dots, n$ **do**

$s_j \leftarrow u_j$

% mapping to $T_{d_{\text{ref},*}} \mathcal{M}$

$u_j \leftarrow \log_{d_{\text{ref},2}}(d_j)$

end for

for $j = 0, \dots, 3$ **do**

$\tilde{x}_j \leftarrow \sum_{k=0}^n w_{(2i+j)k} s_k$ % cp generation

$\hat{x}_j \leftarrow \sum_{k=0}^n w_{(2i+j)k} u_k$

end for

% first point of the segment

if $i = 0$ **then**

$\check{p}_0 \leftarrow \tilde{x}_0$ **and** $\hat{p}_0 \leftarrow \hat{x}_0$

else

$\check{p}_i \leftarrow 0.5(\tilde{x}_0 + \tilde{x}_1)$ % C^1 -condition

$\hat{p}_i \leftarrow 0.5(\hat{x}_0 + \hat{x}_1)$

end if

% last point of the segment

if $i = n - 1$ **then**

$\check{p}_n \leftarrow \tilde{x}_3$ **and** $\hat{p}_n \leftarrow \hat{x}_3$

else

$\check{p}_{i+1} \leftarrow 0.5(\tilde{x}_2 + \tilde{x}_3)$ % C^1 -condition

$\hat{p}_{i+1} \leftarrow 0.5(\hat{x}_2 + \hat{x}_3)$

end if

% inner points of the segment

$\tilde{b}_i^+ \leftarrow \tilde{x}_1$ **and** $\hat{b}_i^+ \leftarrow \hat{x}_1$

$\tilde{b}_{i+1}^- \leftarrow \tilde{x}_2$ **and** $\hat{b}_{i+1}^- \leftarrow \hat{x}_2$

end for

Algorithm 4 Reconstruction of the C^1 blended cubic spline of Definition 4.16 at time t

Require:

$$\begin{aligned} i &\in \{0, \dots, n-1\}, t \in [i, i+1], \\ (\tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-, \tilde{p}_{i+1}) &\in T_{d_i} \mathcal{M}, \\ (\hat{p}_i, \hat{b}_i^+, \hat{b}_{i+1}^-, \hat{p}_{i+1}) &\in T_{d_{i+1}} \mathcal{M} \end{aligned}$$

$$\begin{aligned} w &\leftarrow 3t^2 - 2t^3 \\ \tilde{x} &\leftarrow \beta_3(t; \tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-, \tilde{p}_{i+1}), & x &\leftarrow \exp_{d_i}(\tilde{x}) \\ \hat{y} &\leftarrow \beta_3(t; \hat{p}_i, \hat{b}_i^+, \hat{b}_{i+1}^-, \hat{p}_{i+1}), & y &\leftarrow \exp_{d_{i+1}}(\hat{y}) \\ z &\leftarrow \text{av}[(x, y), (1-w, w)] \end{aligned}$$

return z

Proof. By construction of the control points, the curve $\tilde{\gamma}$ corresponds to two successive pieces of the optimal fitting curve in the Euclidean space $T_{d_i} \mathcal{M}$ for the data points $\log_{d_i}(d_j)$, $j = 0, \dots, n$. This optimal fitting curve is known to be a natural cubic spline; see, e.g., [GS93]. \square

Theorem 4.18. Consider the data points $d_0, \dots, d_n \in \mathcal{M}$ associated with the parameter-values $t_i = i$, $i = 0, \dots, n$. The blended cubic spline $\mathbf{B}(t): [0, n] \rightarrow \mathcal{M}$ of Definition 4.16 satisfies the following properties:

- (i) $\mathbf{B}(i) \rightarrow d_i$ when $\lambda \rightarrow \infty$, for $i = 0, \dots, n$;
- (ii) $\mathbf{B}(t)$ is well defined and smooth under the assumptions from the introduction, i.e., that the exponential and the logarithm map are well defined and smooth. This holds in particular if \mathcal{M} is complete and, on each piece, $d_{\mathcal{M}}(L(\tau), R(\tau)) < r^*(\tau)$, $r^*(\tau) = \max(r_{R(\tau)}, r_{L(\tau)})$, for all $\tau \in [0, 1]$.
- (iii) when \mathcal{M} is a Euclidean space, $\mathbf{B}(t)$ is the cubic smoothing spline that minimizes (4.1) over the Sobolev space $H^2(0, n)$.

Proof. By Proposition 4.14, $\beta_3^i(0) = p_i$ and $\beta_3^i(1) = p_{i+1}$, for $i = 0, \dots, n-1$. Therefore, $\mathbf{B}(i) = p_i$. When $\lambda \rightarrow \infty$, $\tilde{p}_i = 0 \in T_{d_i} \mathcal{M}$ by (4.6) and (4.15), so $p_i = d_i \in \mathcal{M}$, and property (i) is verified $\forall i$. The proof of property (iii) is direct from Lemma 4.17, as $T_{d_0} \mathbb{R}^m = T_{d_1} \mathbb{R}^m = \dots = T_{d_n} \mathbb{R}^m$. Let us now prove property (ii). Again by Proposition 4.14, \mathbf{B} is C^1 on $t \neq i$. There remains to show that $\mathbf{B}(t)$ is differentiable at $t = i$. For $i \in \{0, n\}$ differentiability is trivial. Let $i \in \{1, \dots, n-1\}$. Consider the curve $\tilde{\gamma}(t)$ of Lemma 4.17 and $\gamma(t) = \exp_{d_i}(\tilde{\gamma}(t))$. By definition, $w'(t) = 0$ at $t \in \{0, 1\}$.

Therefore,

$$\begin{aligned} \frac{d}{dt} \mathbf{B}(t)|_{t=i^+} &= \frac{d}{dt} \text{av}[(L_i(t), R_i(t)), (1-w(t), w(t))]|_{t=0^+} \\ &= \frac{d}{dt} L_i(t)|_{t=0^+} = \frac{d}{dt} \gamma(t)|_{t=1^+}, \end{aligned}$$

where $R_i(t)$ and $L_i(t)$ are the intermediate functions (4.13) and (4.14) of the i^{th} blended curve of \mathbf{B} . Similarly,

$$\frac{d}{dt} \mathbf{B}(t)|_{t=i^-} = \frac{d}{dt} R_{i-1}(t)|_{t=1^-} = \frac{d}{dt} \gamma(t)|_{t=1^-}.$$

As $\exp_{d_i}(\cdot)$ is a smooth mapping, $\gamma(t)$ is differentiable at $t = 1$, so $\mathbf{B}(t)$ is differentiable at $t = i$, $\forall i$. \square

Remark 4.19. Note that the condition of Theorem 4.18, (ii), is not easy to check in practice. Let us pose $\tilde{L}(t)$ and $\hat{R}(t)$, the smoothing splines computed in $T_{d_i} \mathcal{M}$ and $T_{d_{i+1}} \mathcal{M}$, respectively, evaluated at $t = s - i$, $s \in [i, i + 1]$. Let t be given. By the triangular inequality, one can say that

$$d_{\mathcal{M}}(L(t), R(t)) \leq \|\tilde{L}(t)\| + d_{\mathcal{M}}(d_i, d_{i+1}) + \|\hat{R}(t)\|.$$

By definition, the smoothing splines read

$$\tilde{L}(t) = \sum_{i=0}^3 \sum_{j=0}^n B_{i3}(t) D_{ij} \tilde{d}_j,$$

and accordingly for $\hat{R}(t)$. We pose $\Delta = \max_{ij} d_{\mathcal{M}}(d_i, d_j)$. As $B_{i3}(t) \geq 0$ and $\sum_{i=0}^3 B_{i3}(t) = 1$, one has

$$d_{\mathcal{M}}(L(t), R(t)) \leq (n + 1) \max_{ij} (D_{ij}) (2\Delta) + \Delta.$$

The condition of Theorem 4.18, (ii), is verified if

$$\Delta \leq \frac{r^*(t)}{1 + 2(n + 1) \max_{ij} (D_{ij})}.$$

This condition can be checked a priori. It is sufficient for arbitrary manifolds with nonzero injectivity radius, but it is by no means necessary. In practice, we observed that the property (ii) holds true in all experiments we

4 | Fitting with Bézier, blended, and Bézier-like curves

conducted, and we were not even able to choose the data points maliciously enough to make it wrong.

Proposition 4.20 (Minimal representation of the curve). *The blended cubic spline of Definition 4.16 is uniquely represented by $6(n - 1) + 8$ tangent vectors.*

Proof. The curve is represented by the following tangent vectors:

- 4 vectors $\tilde{p}_0, \tilde{b}_0^+, \tilde{b}_1^-, \tilde{b}_1^+ \in T_{d_0}\mathcal{M}$
- 6 vectors $\tilde{p}_0, \tilde{b}_0^+, \tilde{b}_1^-, \tilde{b}_1^+, \tilde{b}_2^-, \tilde{b}_2^+ \in T_{d_1}\mathcal{M}$
- $6(n - 3)$ vectors $\tilde{b}_{i-1}^-, \tilde{b}_{i-1}^+, \tilde{b}_i^-, \tilde{b}_i^+, \tilde{b}_{i+1}^-, \tilde{b}_{i+1}^+ \in T_{d_i}\mathcal{M}$, for $i = 2, \dots, n - 2$;
- 6 vectors $\tilde{b}_{n-2}^-, \tilde{b}_{n-2}^+, \tilde{b}_{n-1}^-, \tilde{b}_{n-1}^+, \tilde{b}_n^-, \tilde{p}_n \in T_{d_{n-1}}\mathcal{M}$
- 4 vectors $\tilde{b}_{n-1}^-, \tilde{b}_{n-1}^+, \tilde{b}_n^-, \tilde{p}_n \in T_{d_n}\mathcal{M}$.

□

Proposition 4.21 (Exponential and logarithm maps required). *The number of exponential and logarithm maps required by the blended cubic spline (Definition 4.16) is*

- $n(n + 1)$ logarithms for the construction of the minimal representation of the curve of Proposition 4.20
- 3 exponential maps and 1 logarithm map to reconstruct the curve $\mathbf{B}(t)$, for a given parameter value t , given that minimal representation.

Proof. The $n(n + 1)$ logarithms maps are required for the representation of the data points d_j , $j \neq i$, in the tangent space $T_{d_i}\mathcal{M}$. The reconstruction of the curve at a given value of t requires 2 exponential maps to map on the manifold the values of the Euclidean Bézier curves computed in the tangent spaces at d_i and d_{i+1} , with $i = \lfloor t \rfloor$, and one additional logarithm and exponential map, for the weighted average of those two values. □

4.3 Fitting with composite Bézier-like curves

This section is dedicated to present two other ideas fixing the differentiability issue identified in Proposition 4.11. The approaches are based on the

observation that the curve

$$\tilde{\gamma}_i: [0, 2] \rightarrow \mathcal{M}: \tilde{\gamma}_i(t) = \begin{cases} g(t; b_i^-, p_i) & \text{for } t \in [0, 1], \\ g(t; p_i, b_i^+) & \text{for } t \in [1, 2] \end{cases}$$

(see Figure 4.4) is not differentiable at $t = 1$. This curve, however, is used at the first step of the De Casteljau algorithm to reconstruct Bézier curves of \mathbf{B} afterwards (Definition 3.12). A natural (but naive) idea to fix this problem would be to replace these two geodesics (in the De Casteljau algorithm) by any differentiable curve $\gamma_i(t): [0, 2] \rightarrow \mathcal{M}$ satisfying $\gamma_i(0) = b_i^-$, $\gamma_i(1) = p_i$ and $\gamma_i(2) = b_i^+$ (for instance, $\gamma(t) = \exp_{d_i}(\tilde{\gamma}(t))$ from Lemma 4.17).

We explain here why this approach does not fix the differentiability problem identified in Proposition 4.11. Then we show how this approach can be modified in order to obtain C^1 fitting curves satisfying the interpolation property as $\lambda \rightarrow \infty$: the trick is to replace the De Casteljau method by a more general algorithm. However, the curve obtained will no longer reduce to a natural cubic spline in the case $\mathcal{M} = \mathbb{R}^m$.

This section relies on the notion of *blossom* of a recursive function. Blossoms can be viewed as a manner to generalize the De Casteljau algorithm. Indeed, at each step of the recursion, one can choose a different function and a different evaluation time, while the De Casteljau algorithm is limited to performing only geodesics at a given time t . The section is composed of two parts. We first recall the general theory and remarkable theorems about blossoms. Then, we present and analyse two new fitting methods based on that theory.

4.3.1 Blossom functions

Blossoms are conceptual mathematical objects described, *e.g.*, in [Far02, LW01]. We refer the reader to these documents for more details and extend here the results to a more general setting that will be useful to our developments.

Definition 4.22 (*Manifold-valued blossom*). Let $x_0, \dots, x_K \in \mathcal{M}$, $K \in \mathbb{N}$, be a set of points associated with the parameter values $t_0, \dots, t_K \in [0, 1]$. Consider a set of smooth functions $f_i(\cdot; x, y): [0, 1] \rightarrow \mathcal{M}$, $i = 1, \dots, K$ such that $f_i(0; x, y) = x$ and $f_i(1; x, y) = y$ for all i . We will later refer to such functions as *endpoint functions*. The associated recursive function

endpoint
function f_i

4 | Fitting with Bézier, blended, and Bézier-like curves

$h_K(\cdot; x_0, \dots, x_K): [0, 1] \rightarrow \mathcal{M}$ is defined as

$$h_i(t; x_0, \dots, x_i) = f_i(t; h_{i-1}(t; x_0, \dots, x_{i-1}), h_{i-1}(t; x_1, \dots, x_i)),$$

with $h_0(t; x_j) = x_j$, $j = 0, \dots, K$. The notion of *blossom* is a generalization of h_K , where different values of t can be used at each step of the recursion. The *blossom* of h_K is thus the map $\psi_K: \mathbb{R}^{K+1} \rightarrow \mathcal{M}$ given by

$$\begin{aligned} \psi_i(t_0, \dots, t_i; x_0, \dots, x_i) = \\ f_i(t_i; \psi_{i-1}(t_0, \dots, t_{i-1}; x_0, \dots, x_{i-1}), \\ \psi_{i-1}(t_0, \dots, t_{i-1}; x_1, \dots, x_i)), \end{aligned} \quad (4.16)$$

with $\psi_0(t_0; x_j) = x_j$ for all j . Observe therefore that

$$\psi_K(t, \dots, t; x_0, \dots, x_K) = h_K(t; x_0, \dots, x_K). \quad (4.17)$$

Example 4.23 (*De Casteljau algorithm*). One can of course define a blossom for the Bézier curves of degree K on manifolds, based on the De Casteljau algorithm (Definition 3.5). In that case, the curve $f_i(\cdot; x, y) = g(\cdot; x, y)$ is the geodesic between x and y , for all i .

Proposition 4.24 (Endpoint interpolation and velocity). *For any set of data points $x_0, \dots, x_K \in \mathcal{M}$, the following properties hold:*

- (i) $h_K(0; x_0, \dots, x_K) = x_0$,
- (ii) $h_K(1; x_0, \dots, x_K) = x_K$,
- (iii) $\dot{h}_K(0; x_0, \dots, x_K) = \sum_{i=1}^K \frac{d}{dt} \Big|_{t=0} f_i(t; x_0, x_1)$,
- (iv) $\dot{h}_K(1; x_0, \dots, x_K) = \sum_{i=1}^K \frac{d}{dt} \Big|_{t=1} f_i(t; x_{K-1}, x_K)$.
- (v) if $f_i(\cdot; x, y) \in C^1$ and is smooth in x, y , $h_K(\cdot; x_0, \cdot, x_K) \in C^1$ as well.

Proof. Properties (i) and (ii) follow directly from the definition of the functions f_i : indeed, one has $f_i(0; x, y) = x$ and $f_i(1; x, y) = y$, for all i . Properties (iii) and (iv) are proven in a similar way as [PN07, Theorem 1]. By (4.17) one has

$$\dot{h}_K(0; x_0, \dots, x_K) = \sum_{i=1}^K \frac{\partial}{\partial t_i} \Big|_{t_i=0} \psi_K(0, \dots, 0, t_i, 0, \dots, 0; x_0, \dots, x_K).$$

As $f_i(0; x, y) = x$ for $x, y \in \mathcal{M}$, one has, by [PN07, Lemma 3 (ii)],

$$\psi_K(0, \dots, 0, t_i, 0, \dots, 0; x_0, \dots, x_K) = f_i(t_i; x_0, x_1).$$

As a result, we obtain

$$\dot{h}_K(0; x_0, \dots, x_K) = \sum_{i=1}^K \frac{d}{dt} \Big|_{t=0} f_i(t; x_0, x_1).$$

(iv) is obtained symmetrically. Finally (v): the smoothness of h_K is preserved by composition. \square

4.3.2 Bézier-like fitting curves

In this section, we propose two modifications of the classical De Casteljau algorithm (Definition 3.12) in order to produce a fitting curve for a set of data points d_0, \dots, d_n associated with parameters $t_0 = 0, \dots, t_n = n$. The underlying idea is to modify the functions f_i used to construct the function h_K from Definition 4.22.

Definition 4.25 (*Bézier-like fitting curve – type I*). Let $\lambda > 0$. For $i \in \{0, \dots, n - 1\}$, we define $\tilde{p}_i, \tilde{b}_i^+ \in T_{d_i}\mathcal{M}$ and $\hat{b}_{i+1}^-, \hat{p}_{i+1} \in T_{d_{i+1}}\mathcal{M}$, the vectors computed with (4.6) and (4.15) in their respective tangent spaces. We consider the corresponding points p_i, b_i^+, b_{i+1}^- and $p_{i+1} \in \mathcal{M}$. The cubic Bézier-like curve $h_3(\cdot; p_i, b_i^+, b_{i+1}^-, p_{i+1}): [0, 1] \rightarrow \mathcal{M}$ is recursively computed, according to Definition 4.22, with the following endpoint functions:

Bézier-like curve

$$\begin{aligned} f_1(t; p_i, b_i^+) &:= \gamma_i(1 + t), \\ f_1(t; b_i^+, b_{i+1}^-) &:= g(t; b_i^+, b_{i+1}^-), \\ f_1(t; b_{i+1}^-, p_{i+1}) &:= \gamma_{i+1}(t), \\ f_k(t; x, y) &:= \text{av}[(x, y), (1 - w(t), w(t))], \quad k = 2, 3, \end{aligned}$$

where $w(t) = 3t^2 - 2t^3$ and $\gamma_i: [0, 2] \rightarrow \mathcal{M}$, $\gamma_i(t) := \exp_{d_i} \left(\frac{(2-t)}{2} \tilde{b}_i^- + \frac{t}{2} \tilde{b}_i^+ \right)$ is a straight line on $T_{d_i}\mathcal{M}$, mapped to \mathcal{M} . As illustrated in Figure 4.4, $\gamma(1) = p_i$. Observe that, in a mild abuse of notation, the definition of f_1 depends on the name of its arguments.

The composite cubic Bézier-like curve of type I $\mathbf{B}(t)$ is then defined according to Definition 3.15. This definition is illustrated in Figure 4.8.

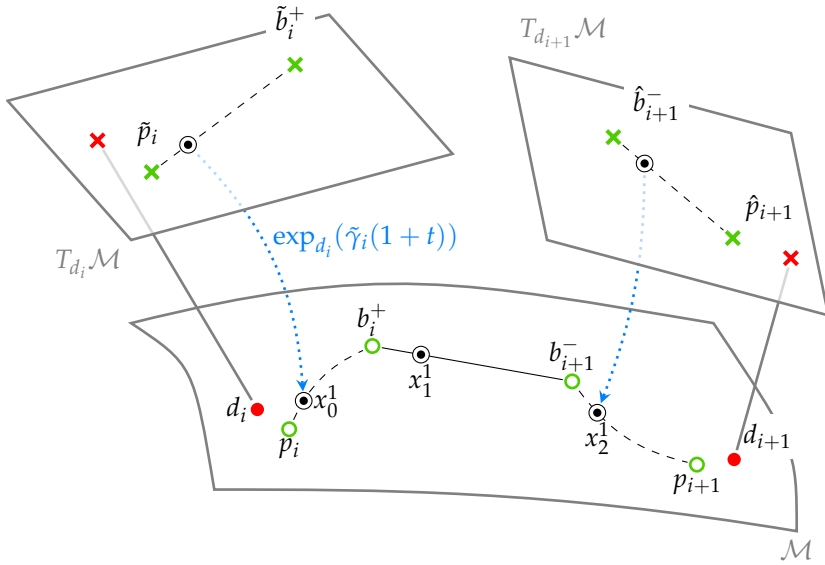


Fig. 4.8 Composite Bézier-like curve of type I. The first step of the algorithm is a hybrid method where the first (resp. last) geodesic is replaced by the curve $\gamma_i(1+t)$ (resp. $\gamma_{i+1}(t)$). The middle one is still a classical geodesic between b_i^+ and b_{i+1}^- . The next steps of the algorithm are classical weighted averagings.

Remark 4.26. This definition of Bézier-like curve is directly related to the interpolation conditions (4.15) but not restricted to cubic curves. Indeed, it can be easily generalized to curves of degree K , where $f_1(t; p_i, b_i^+)$ and $f_1(t; b_{i+1}^-, p_{i+1})$ would correspond to the Euclidean operation mapped to \mathcal{M} , while the “other” (endpoint) functions $f_1(t; x, y)$ would be classical geodesics between x and y .

Lemma 4.27. Let $x, y \in \mathcal{M}$ and $w(t) = 3t^2 - 2t^3$. Let $z := f(t; x, y) = \text{av}[(x, y), (1 - w(t), w(t))]$. Then

$$\left. \frac{d}{dt} \right|_{t=s} f(t; x, y) = 0, \text{ for } s \in \{0, 1\}.$$

Proof. The optimality condition of $z = f(t; x, y)$ are given by [Kar77, Thm 1.2]:

$$0 = \log_z(x) (1 - w(t)) + \log_z(y) w(t) := F(t, z).$$

By the implicit function theorem, one has

$$\frac{d}{dt}f(t; x, y) = -(D_z F(t, z))^{-1} D_t F(t, z),$$

where $D_t F(t, z) = (\log_z(y) - \log_z(x)) w'(t)$. As $w'(t) = 6t - 6t^2$, we see that $D_t F(t, z)|_{t=0} = D_t F(t, z)|_{t=1} = 0$, and $\frac{d}{dt}\Big|_{t=s} f(t; x, y) = 0$ for $s \in \{0, 1\}$. □

Implicit function theorem (summary): let $y = f(x)$ and $F(x, f(x)) = 0$, for f and F smooth. Hence, $\frac{\partial f}{\partial x} = -D_y F(x, y)^{-1} \frac{\partial F}{\partial x}$.

Theorem 4.28. Consider the data points $d_0, \dots, d_n \in \mathcal{M}$ associated with the parameter-values $t_0 = 0, \dots, t_n = n$. The composite cubic Bézier-like curve $\mathbf{B}(t)$ of type I (Definition 4.25) satisfies the following properties:

differentiability conditions

- (i) $\mathbf{B}(i) \rightarrow d_i$, when $\lambda \rightarrow \infty$;
- (ii) $\mathbf{B}(t)$ is differentiable for $t \in [0, n]$.

Proof. As $\text{av}[(x, y), (1, 0)] = x$ and $\text{av}[(x, y), (0, 1)] = y$, the cubic Bézier-like curves h_3 from Definition 4.25 are recursive functions as defined in Definition 4.22. Therefore, we can apply Proposition 4.24 (i–ii). By condition (4.15), we prove (i) because $\tilde{p}_i \rightarrow 0$ when $\lambda \rightarrow \infty$, so $p_i \rightarrow d_i$. For condition (ii), let $i = 1, \dots, n - 1$. $\mathbf{B}(t)$ is smooth for $t \neq i$, as h_3 is smooth (Proposition 4.24 (v)). For $t = i$, we have by Lemma 4.27 that $\frac{d}{dt}\Big|_{t=s} f_k(t; x, y) = 0$, for $s \in \{0, 1\}$. Therefore, by Proposition 4.24 (iii–iv), one has

$$\frac{d\mathbf{B}(t)}{dt}\Big|_{i^-} = \frac{d\gamma_i(t)}{dt}\Big|_{1^-}$$

and

$$\frac{d\mathbf{B}(t)}{dt}\Big|_{i^+} = \frac{d\gamma_i(t)}{dt}\Big|_{1^+}.$$

As $\gamma_i(t)$ is differentiable, so is $\mathbf{B}(t)$. □

Remark 4.29. In Definition 4.25, one could be tempted to simply use classical geodesics in place of $f_k(t; x, y)$, $k = 2, 3$. However, by Proposition 4.24 the left and right velocities of \mathbf{B} at $t = i$ will in general not be the same: indeed, as shown in Proposition 4.11, b_i^-, p_i and b_i^+ are not always aligned.

Proposition 4.30 (Minimal representation of the curve). *The fitting Bézier-like curve of Definition 4.25 is uniquely represented by $2(n + 1)$ tangent vectors.*

minimal representation

Proof. The proof is similar to the one of Proposition 4.9, the same tangent vectors can be used to represent the curve. □

4 | Fitting with Bézier, blended, and Bézier-like curves

Exp-Log
complexity

Proposition 4.31 (Exponential and logarithm maps required). *The number of exponential and logarithm maps required by the fitting Bézier-like curve of Definition 4.25 is*

- $n(n + 1)$ logarithms for the construction of the minimal representation of the curve;
- 8 exponential maps and 4 logarithm maps to reconstruct the curve $\mathbf{B}(t)$, for a given parameter value t , given that minimal representation.

Proof. The proof is here also similar to the one of Proposition 4.10. The differences between the two methods is that two geodesics are spared at the first step of the algorithm. \square

One strength of the method is that differentiability of the composite curve \mathbf{B} (Definition 4.25) at $t = i$ depends only on the differentiability of γ_i . The curve $\gamma_i(t)$ can be replaced by any differentiable curve between b_i^- and b_i^+ and such that $\gamma(0.5) = p_i$. This property is possible because $\frac{d}{dt} \text{av}[(x, y), (1 - w, w)]|_{t=s} = 0$, for $s \in \{0, 1\}$ and $w(t) = 3t^2 - 2t^3$. Then, most of the computation of the pieces of $\mathbf{B}(t)$ can be transferred to the tangent spaces $T_{d_i}\mathcal{M}$ and $T_{d_{i+1}}\mathcal{M}$, $i = 0, \dots, n - 1$. For cubic curves, for instance, the curve $\gamma_i: [0, 2] \rightarrow \mathcal{M}$ can be the mapping to \mathcal{M} of $\tilde{\gamma}_i(t)$, composed of two C^1 -patched quadratic Bézier curves computed on the tangent space of d_i , as

$$\tilde{\gamma}_i(t) = \begin{cases} \beta_2(t; \tilde{b}_{i-1}^+, \tilde{b}_i^-, \tilde{p}_i) & \text{for } t \in [0, 1] \\ \beta_2(t - 1; \tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-) & \text{for } t \in [1, 2], \end{cases} \quad (4.18)$$

where \tilde{x} is the point $x \in \mathcal{M}$ represented in $T_{d_i}\mathcal{M}$. The resulting curve h_3 would thus be an averaging of the two curves $\gamma_i(t)$ and $\gamma_{i+1}(t)$, as represented on Figure 4.9.

This leads us to the following definition.

Definition 4.32 (Bézier-like fitting curve – type II). Let $\lambda > 0$. For $i \in \{0, \dots, n - 1\}$, let $p_i, b_i^+, b_{i+1}^-, p_{i+1}$ be the control points computed with (4.6) and (4.8), and let $\tilde{x} = \log_{d_i}(x) \in T_{d_i}\mathcal{M}$ and $\hat{x} = \log_{d_{i+1}}(x) \in T_{d_{i+1}}\mathcal{M}$, the representation of these control points in the corresponding tangent spaces. Let also $\tilde{b}_{i+1}^- = \log_{d_i}(\exp_{d_{i+1}}(\hat{b}_{i+1}^-))$, $i = 0, \dots, n - 1$, and $\hat{b}_i^+ = \log_{d_{i+1}}(\exp_{d_i}(\tilde{b}_i^+))$, $i = 0, \dots, n - 1$. The cubic Bézier-like curve (type II)

Bézier-like
curve

$y(t): [0, 1] \rightarrow \mathcal{M}$ is computed with the following iterative procedure:

$$\begin{aligned} \tilde{x}_i^L &:= \beta_2(t; \tilde{p}_i, \tilde{b}_i^+, \tilde{b}_{i+1}^-) \\ \hat{x}_i^R &:= \beta_2(t; \hat{b}_i^+, \hat{b}_{i+1}^-, \hat{p}_{i+1}) \\ x_i^L &:= \exp_{d_i}(\tilde{x}_i^L) \\ x_i^R &:= \exp_{d_{i+1}}(\hat{x}_i^R) \\ y(t) &:= \text{av}[(x_i^L, x_i^R), (1 - w(t), w(t))] \end{aligned}$$

where $w(t) = 3t^2 - 2t^3$. The composite cubic Bézier-like curve (type II) $\mathbf{B}(t)$ is then defined as (3.11). This definition is illustrated at Figure 4.9.

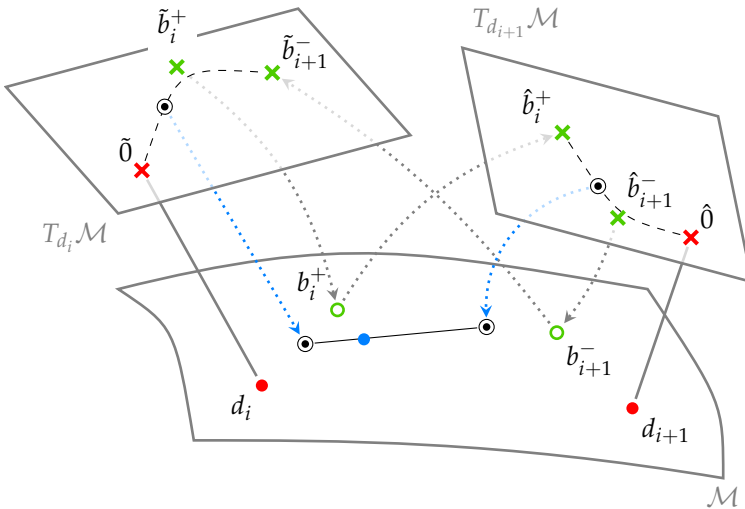


Fig. 4.9 Composite Bézier-like curve of type II. The first step of the algorithm consists in mapping on the manifold the quadratic Bézier curves computed on the tangent spaces $T_{d_i}\mathcal{M}$ and $T_{d_{i+1}}\mathcal{M}$. The value of the composite Bézier-like curve is then obtained by a classical weighted averaging of these two points (black circles).

Remark 4.33. The exponential and logarithm map evaluations represent usually the major part of the computation effort. Indeed, these maps are not always closed form and might require lengthy iterative procedures to be evaluated (e.g., on the space of shapes [KDL18]). A direct way to spare $2n - 2$ Exp-Log evaluations is to compute the control points directly on

4 | Fitting with Bézier, blended, and Bézier-like curves

the dedicated tangent space, to avoid transfers from one tangent space to another, as in Algorithm 3. The curve will remain C^1 by Properties 4.24, (iii–iv).

Theorem 4.34. *Consider the data points $d_0, \dots, d_n \in \mathcal{M}$ associated with the parameter-values $t_0 = 0, \dots, t_n = n$. The composite cubic Bézier-like curve $\mathbf{B}(t)$ of type II (Definition 4.32) satisfies the following properties:*

- (i) $\mathbf{B}(i) \rightarrow d_i$, when $\lambda \rightarrow \infty$;
- (ii) $\mathbf{B}(t)$ is differentiable for $t \in [0, n]$.

Proof. The proof is similar to the proof of Theorem 4.28. The condition (i) is reached via Proposition 4.24 (i–ii) and condition (4.8). Let $i = 1, \dots, n - 1$. Differentiability for $t \neq i$ is trivial. For $t = i$, condition (ii) follows from Lemma 4.27: as $\gamma_i(t) = \exp_{d_i}(\tilde{\gamma}_i(t))$ (Equation (4.18)) is differentiable, so is $\mathbf{B}(t)$. \square

Proposition 4.35 (Minimal representation of the curve). *The fitting Bézier-like curve of Definition 4.32 is uniquely represented by $2(n + 1)$ tangent vectors.*

Proof. The proof is similar to the one of Proposition 4.9, the same tangent vectors can be used to represent the curve. \square

Remark 4.36. There is no need to store the tangent vectors $\tilde{b}_{i+1}^- = \log_{d_i} b_{i+1}^-$ and $\hat{b}_i^+ = \log_{d_{i+1}} b_i^+$, as they can be recovered afterwards. Indeed, $\tilde{b}_{i+1}^- = \log_{d_i} \left(\exp_{d_{i+1}} \hat{b}_{i+1}^- \right)$, and $\hat{b}_i^+ = \log_{d_{i+1}} \left(\exp_{d_i} \tilde{b}_i^+ \right)$.

Proposition 4.37 (Exponential and logarithm maps required). *The number of exponential and logarithm maps required by the fitting Bézier-like curve of Definition 4.32 is*

- $n(n + 1)$ logarithms for the construction of the minimal representation of the curve of Proposition 4.35
- 5 exponential maps and 3 logarithm maps to reconstruct the curve $\mathbf{B}(t)$, for a given parameter value t , given that minimal representation.

Proof. The $n(n + 1)$ logarithm maps are required to represent the data points d_j in the tangent space $T_{d_i} \mathcal{M}$, $j \neq i$. At the reconstruction step, 2 exponentials and 2 logarithms are required to compute \hat{b}_i^+ and \tilde{b}_{i+1}^- , as mentioned in Remark 4.36. Finally, 2 exponential maps are required to map \tilde{x}_i^L and \hat{x}_i^R to \mathcal{M} , and the averaging of these two points costs 1 exponential map and 1 logarithm map. \square

Remark 4.38. A last thing to remark about Definitions 4.25 and 4.32 is that the reconstructed composite cubic Bézier-like curves no longer reduce to natural cubic splines when $\mathcal{M} = \mathbb{R}^m$. This will be shown numerically in the next section.

4.4 Numerical examples

In this section, we compare the four fitting methods described in this chapter. That is, the Bézier fitting curve from Definition 4.7 (Bézier), the blended fitting curve from Definition 4.16 (Blended), and the Bézier-like fitting curves defined in Section 4.3, *i.e.*, Definitions 4.25 (BL-I) and 4.32 (BL-II). The goals of the tests we have performed are twofold. The first goal is to validate numerically some of the target properties from the introduction. We consider here two such properties: property (i) (interpolation of the data as $\lambda \rightarrow \infty$), see Section 4.4.1, and property (iii) (recovering of the natural cubic spline in the Euclidean case), see Section 4.4.2. The second goal of this section is to compare the curves regarding the value of the optimization problem (4.1). This is done in Section 4.4.3, for two different manifolds: the sphere S^2 and the special orthogonal group $SO(3)$.

As a comparison, we also consider the simpler approach in which the fitting curve is entirely computed in a unique tangent space $T_{d_{\text{ref}}}\mathcal{M}$. One of the drawbacks of this approach is that the result usually depends on the tangent space chosen. In short, we compute the optimal (Euclidean) smoothing spline in $T_{d_{\text{ref}}}\mathcal{M}$ using the control points (4.6) and the condition (4.15), with a unique $d_{\text{ref}} = d_{\text{mid}}$, where $d_{\text{mid}} = d_{n/2}$ if n is even, and

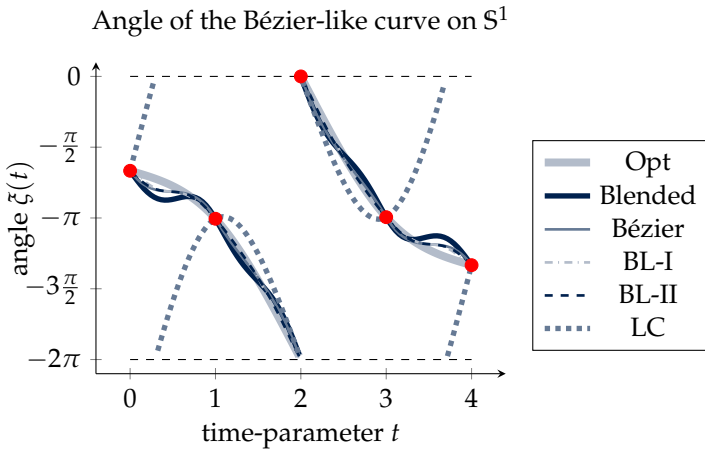
$$d_{\text{mid}} = g(0.5, d_{(n-1)/2}, d_{(n+1)/2})$$

if n is odd. This (Euclidean) fitting curve is then mapped back to \mathcal{M} . The latter method will be referred to as LC (for **L**ocal **C**urve) in our results.

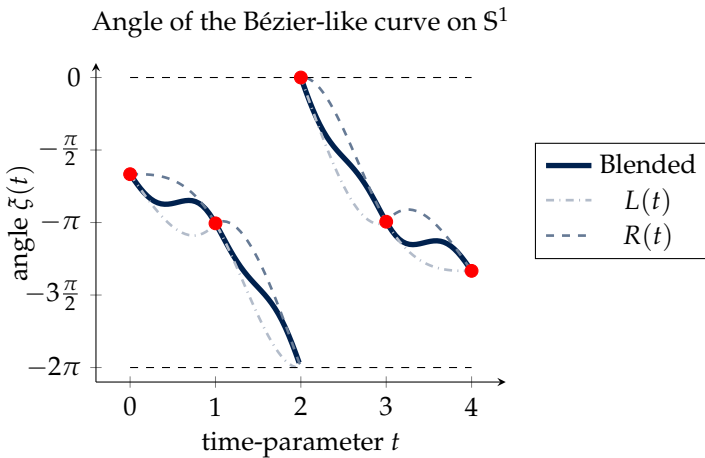
4.4.1 Validation of property (i): interpolation as $\lambda \rightarrow \infty$

We consider the two data sets from Section 4.1.3 (namely, the unit circle S^1 represented on Figure 4.3a and the wind field data, lying on $S_+(3024, 20)$). We compute the different fitting curves for $\lambda \rightarrow \infty$ on these datasets. Similarly to Section 4.1.3, the parameter λ is set to 10^8 .

sphere S^1
Wind field,
 $S_+(n, p)$



(a) The five methods interpolate the data points.



(b) Explanation of the behavior of the blended cubic spline.

Fig. 4.10 Comparison of the five methods on the counter-example of Section 4.1.3, on S^1 . **Top:** the fitting parameter is set to $\lambda = 10^8$. In accordance with Proposition 4.6, the data points are almost interpolated by all methods. We see that the local curve LC can lead to a drastically different result. **Bottom:** The blended cubic spline is computed, at each time, as a weighted average of $L(t)$ and $R(t)$ (see Definition 4.12). The fact that the two curves are computed on two different tangent spaces results here in strong differences between them after projection on the manifold.

The resulting curves are represented in Figure 4.10a and Figure 4.11. These two figures indicate that all the methods satisfy the interpolation property, for both datasets. Moreover, Figure 4.10a indicates that all the methods tend to behave in a similar way, except the local curve LC, which leads to a considerably different result. This can be explained by the fact that the data points are spread out on S^1 . Therefore, the mappings $\log_{d_{\text{mid}}}(d_i)$, $i = 0, \dots, 4$, are in general very distorted representations of the data points $d_i \in S^1$. It can also be observed on Figure 4.11 that the errors for the local curve are larger on the boundaries of the interval, *i.e.*, in the region in which the distortions expected from the mapping into the tangent space $T_{d_{\text{mid}}}\mathcal{M}$ are the largest.

As a comparison, we computed the optimal solution on the circle (Opt). Indeed, on that particular manifold, is it possible to compute the optimal solution to (4.1). The only particularity of the circle, with respect to the Euclidean space, is that two points distant of an angle 2π are equivalent. So, we can solve the problem (4.1) in two steps: first, compute the optimal representative for the angles characterizing the data, which is a combinatorial problem, and then, compute the composite Bézier curve in \mathbb{R} that fits those values.

By comparing the mean squared accelerations of the different curves (Bézier: 52.2, Blend: 66.7, BL-I: 56.3, BL-II: 34.2, LC: 71.9, Opt: 17.8), we observe that BL-II performs surprisingly well. We see that the blended cubic spline performs a bit less good, but as we show in Section 4.4.3, this is generally not observable when working with other (randomly generated) datasets. Actually, we expect this acceleration to be due to the choice of the data points, that are in our case far away from each other. Indeed, this causes here a large difference between the two blended curves $L(t)$ and $R(t)$, as represented in Figure 4.10b (see for instance between the points at $t = 0$ and $t = 1$). This difference highlights here the importance of the reference points chosen in all proposed methods. How to tackle this problem while keeping the properties (*i-vi*) is still an open question.

4.4.2 Validation of property (iii): natural cubic splines in the Euclidean case

Figure 4.12 illustrates the curves obtained for a fitting task ($\lambda = 10^2$), on the bidimensional Euclidean space \mathbb{R}^2 . We verify here that the blended curve, the local curve and the Bézier curve are the natural cubic spline, and that

natural cubic
spline

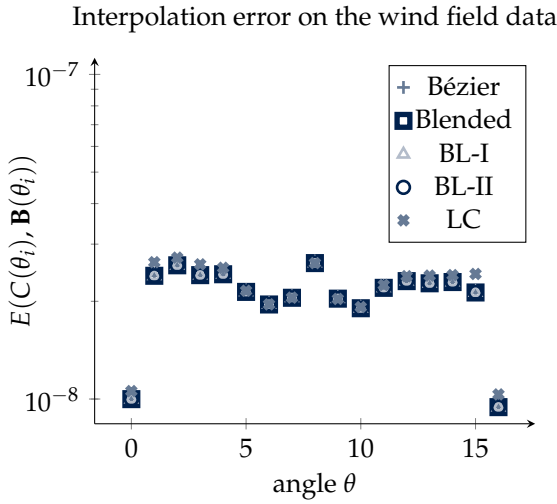


Fig. 4.11 The five reconstruction methods interpolate the data points of the counter-example on $\mathcal{S}_+(3024, 20)$ of Section 4.1.3, when condition (4.8) is respected. As a comparison, Figure 4.2 shows the situation when condition (4.8) is not respected.

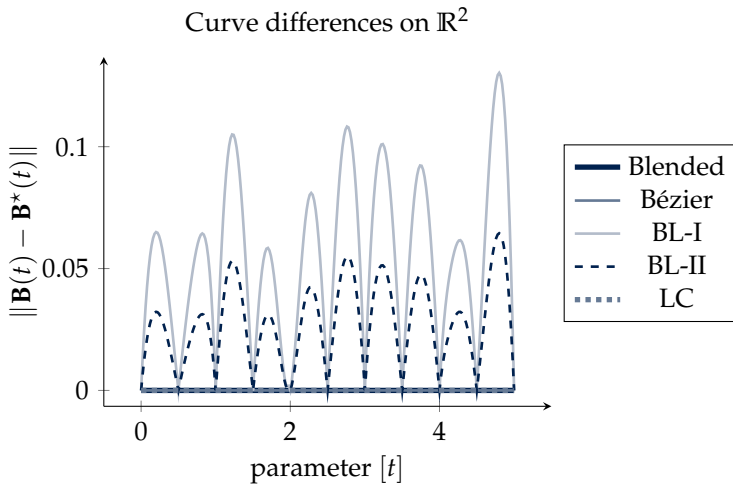
the curves obtained via BL-I and BL-II differ from the optimal solution. Each component of the data points $d_i = (d_{i,x}, d_{i,y}) \in \mathbb{R}^2, i = 0, \dots, 5$, was chosen randomly as $d_{i,x}, d_{i,y} \sim \mathcal{N}(0, 1)$.

We know by Theorem 4.18 and Proposition 4.2 that $\mathbf{B}(t)$ is the natural cubic spline $\mathbf{B}^*(t)$ when it is reconstructed as a composite blended cubic curve (Blended), or as a classical Bézier curve (Bézier and LC), as shown in Figure 4.12a. Figure 4.12b shows us that the curves reconstructed by the methods from Section 4.3 (BL-I and BL-II) are not the natural spline $\mathbf{B}^*(t)$ and that their speed and their path differ strongly.

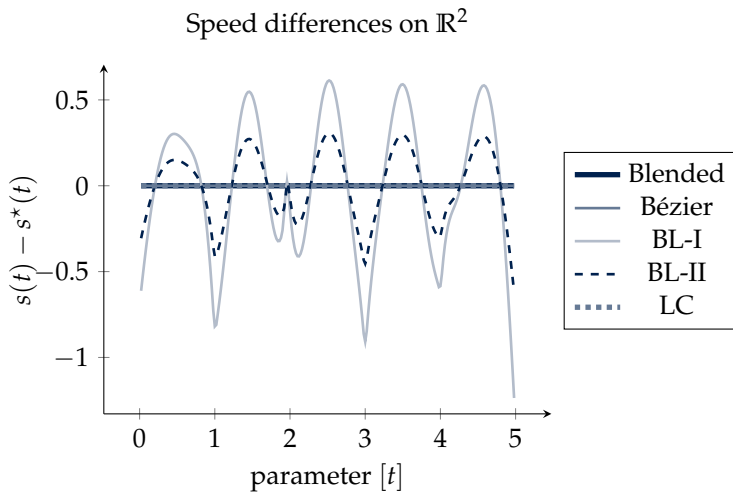
Figure 4.12a also suggests that all methods would return the same curve position at $t = \frac{k}{2}, k \in \mathbb{Z}$. Indeed, as $w(0) = 0, w(0.5) = 0.5$ and $w(1) = 1$, the curves reconstructed by all methods are identical there.

4.4.3 Mean acceleration of the curves

We compare here the acceleration of the different curves, on two manifolds: the sphere \mathcal{S}^2 and the special orthogonal group $\text{SO}(3)$. For each manifold, we generated randomly $N = 1000$ geodesics $\gamma_k: [0, 1] \rightarrow \mathcal{M}$. From these



(a) Differences between the curves on the Euclidean space \mathbb{R}^2 .



(b) Differences between the speed of the curves on the Euclidean space \mathbb{R}^2 .

Fig. 4.12 All methods do not satisfy property (iii), *i.e.*, not all of them reduce to the natural cubic spline when \mathcal{M} is a Euclidean space (here, $\mathcal{M} = \mathbb{R}^2$). **Top:** the curves $\mathbf{B}(t)$ computed by BL-I and BL-II are different from the natural cubic spline $\mathbf{B}^*(t)$. **Bottom:** naturally, their speed $s(t)$ then differ from the speed $s^*(t)$ of the natural cubic spline

4 | Fitting with Bézier, blended, and Bézier-like curves

geodesics, we extracted 6 points $x_{p,k} = \exp_{\gamma_k(p/5)}(v)$, $p = 0, \dots, 5$, where $v \in T_{\gamma_k(p/5)}\mathcal{M}$ is a random vector whose components are distributed according to a classical normal law $\mathcal{N}(0, 0.1^2)$ of mean 0 and standard deviation 0.1. On each data set $(x_{p,k})_{p=0}^5$, with $k = 1, \dots, N$, we built the five fitting curves at 1000 equispaced times $t \in [0, 5]$, and evaluated the acceleration $\ddot{\mathbf{B}}_k(t)$ of each of them using finite differences (see Definition 2.60):

$$\ddot{\mathbf{B}}_k(t_i) = \frac{\log_{\mathbf{B}_k(t_i)}(\mathbf{B}_k(t_{i+1})) + \log_{\mathbf{B}_k(t_i)}(\mathbf{B}_k(t_{i-1}))}{\Delta\tau^2}, \quad \Delta\tau = t_i - t_{i-1}.$$

Figure 4.13a displays, for the sphere S^2 , the mean acceleration of the curves $\mathbb{E}[\|\ddot{\mathbf{B}}(t)\|]$, estimated by averaging the results on the N datasets given by

$$\mathbb{E}[\|\ddot{\mathbf{B}}(t)\|] \simeq \frac{1}{N} \sum_{k=1}^N \|\ddot{\mathbf{B}}_k(t)\|.$$

This figure indicates that the local surface LC results on average in a curve with a considerably larger acceleration than the four other methods.

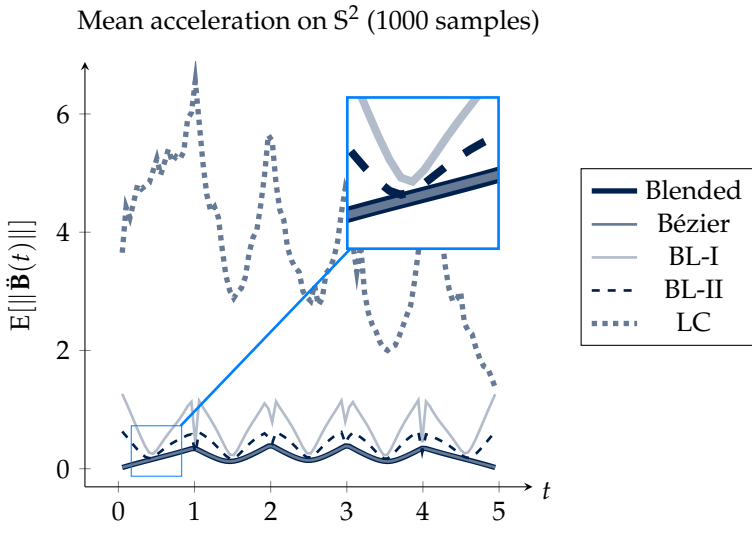
Figure 4.13b presents the results of the same tests, but on the special orthogonal group $SO(3)$. We observe here similar results as on the sphere.

4.5 An application to wind fields estimation

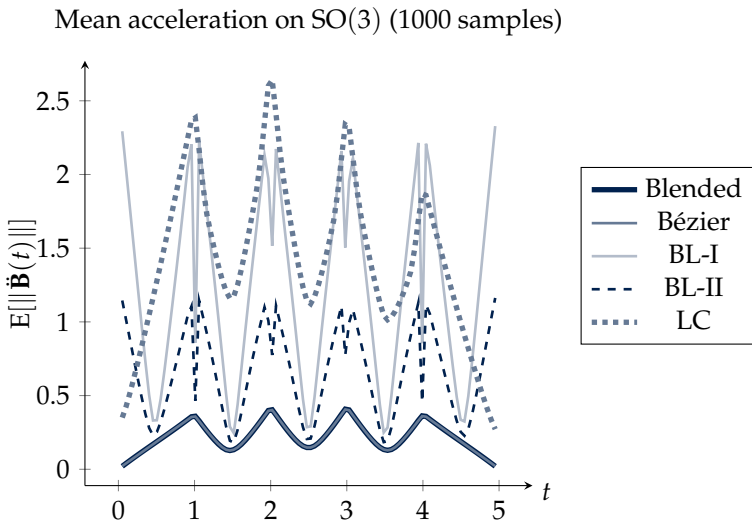
Let us now consider a real-life fitting application to compare the five methods presented in this chapter. The application concerns wind field fitting, as already mentioned in Section 4.1.3.

In this application, the data points are lying on the set of positive semi-definite matrices of size p and rank q , noted $\mathcal{S}_+(p, q)$. More specifically, $p = 3024$ and $q = 20$. The data points represent covariance matrices C_0, \dots, C_n corresponding to wind fields in a certain area of interest. They are associated to the parameters $\theta_0, \dots, \theta_n$ representing the prevailing wind orientation in the area.

This section is the result from a close collaboration with Estelle Massart (UCLouvain, Belgium) and Antoni Musolas (MIT, USA). Estelle provided the elements of differential geometry on $\mathcal{S}_+(p, q)$ while Antoni provided the dataset of covariance matrices.



(a) Mean acceleration on the sphere S^2 .



(b) Mean acceleration on $SO(3)$.

Fig. 4.13 Average acceleration of the curves, computed on 1000 random datasets, with $\lambda = 100$. The blending cubic spline outperforms the methods BL-I, BL-II and LC.

4.5.1 Some words of motivation

The wind fields estimation problem is motivated by applications related to the use of unmanned aerial vehicles (UAV). It appears that the current market proposes more and more uses of drones. To name a few, one could think to military use of surveillance drones, commercial use of delivery UAV for big companies present on the Internet and many more. Hopefully, UAV can also be used for health and care, as it has been demonstrated in Rwanda, where drones deliver medical supplies to hospitals in the whole country, reducing drastically the need of on-site storage and, at the same time, waste and money.

In order to make UAV fly safely and reliably, it is of tremendous importance to take the surrounding environment into account. The external wind conditions, in particular, have an obvious impact on the control of the wings of drones, for instance.

The wind field around the UAV can be modeled as a Gaussian process characterized by a covariance matrix. The covariance matrices depend on external meteorological parameters. In this case, the orientation of the prevailing wind in the area of interest is considered, but other parameters can enter in line, like the amplitude (or the force) of the prevailing wind, the temperature, etc. For each value of the parameters, computationally expensive unsteady CFD (Computational Fluid Dynamics) simulations must be run to estimate the corresponding covariance matrix. The cost of this computation is so high that, even for powerful processors, it is barely possible to obtain a solution in less than one hour.

The goal of this application is to reduce the computational time needed to estimate a covariance matrix. To do so, the strategy is quite simple: the CFD simulations are run only for a few values of the parameter, and then a curve is fitted to the obtained covariance matrices. That way, covariance matrices can be deduced for other values of the parameter. If the fitting method is computationally non-invasive and the estimation acceptable, the goal is met.

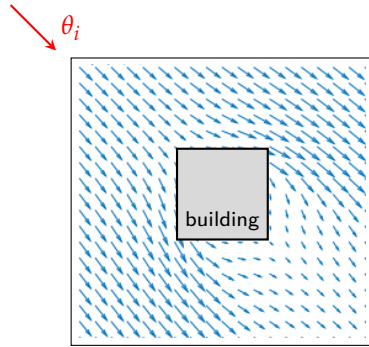


Fig. 4.14 Given a prevailing wind θ_i , local wind orientation might change among the domain, specially when an object (here a building) perturbs it.

For instance, drones are used by the company Zipline to deliver blood: <https://flyzipline.com/company/>

CFD

Note that this strategy is actually the core business of Parametric Model Order Reduction (PMOR) in general (see Section 4.6). This task arises mainly in problems of fluid dynamics and applied to Galerkin methods. In that case, the data points are projectors from the full state space to the reduced state space and hence belong to the Grassmann manifold [PA16].

4.5.2 The manifold $\mathcal{S}_+(p, q)$

The set of $p \times p$ positive semi-definite matrices of rank q admits a manifold structure and is noted $\mathcal{S}_+(p, q)$. As mentioned in Example 2.18, several geometries have been proposed to define $\mathcal{S}_+(p, q)$. We refer to [Van13, §7] for a survey, but also to [VAV09, BS09] for detailed approaches. However, none of those approaches allows to turn $\mathcal{S}_+(p, q)$ into a complete metric space with closed-form expressions for geodesics.

In their work, Massart *et al.* [MA18] resort to the quotient space geometry described in [Van13, §7.2] (i.e., $\mathcal{S}_+(p, q) \simeq \mathbb{R}_*^{p \times q} / \mathcal{O}_p$) but endow the tangent spaces with a very simple metric (the trace of the product of the two tangent vectors). This allows to compute easily the elements of Riemannian geometry; for instance, the geodesic can be computed in closed form. The main advantage of this representation is the low computational cost associated to each operation on the manifold. Indeed, each matrix $S \in \mathcal{S}_+(p, q)$ can be factorized as $S = YY^\top$, where Y belongs to $\mathbb{R}_*^{p \times q}$ (the set of full rank $p \times q$ matrices). This means that every operation is performed on a Y instead of S , which reduces drastically the dimension of the space where the calculus is done, specially when $q \ll p$. With this approach, the exponential map reduces to a sum of two $p \times q$ matrices, while the logarithm map requires, in addition, a polar decomposition of a $q \times q$ matrix.

The geometric elements are obtained as follows. Let $S = YY^\top$ for $S \in \mathcal{S}_+(p, q)$ and $Y \in \mathbb{R}_*^{p \times q}$. The quotient structure is highlighted by the following property: for Q orthogonal of size q , all matrices YQ are then equivalent because $(YQ)(YQ)^\top = YY^\top = S$. For two tangent vectors $\eta, \xi \in T_S \mathcal{S}_+(p, q)$, the metric is defined as $\langle \eta, \xi \rangle_Y = \text{tr}(\eta^\top \xi)$. The end-point geodesic $g(\cdot; S_0, S_1) : [0, 1] \rightarrow \mathcal{S}_+(p, q) : t \mapsto g(t; S_0, S_1)$, with $g(0; S_0, S_1) = S_0 = Y_0 Y_0^\top$ and $g(1; S_0, S_1) = S_1 = Y_1 Y_1^\top$, is given by $g(t; S_0, S_1) = g_Y(t) g_Y(t)^\top$, where

$$g_Y(t) = (1 - t)Y_0 + tY_1Q^\top,$$

 $\mathcal{S}_+(n, p)$

 metric
geodesic

4 | Fitting with Bézier, blended, and Bézier-like curves

with Q the orthogonal factor of the polar decomposition $Y_0^\top Y_1 = HQ$. In the Y -representation, the Riemannian exponential and logarithm are thus

$$\exp_Y(\eta) = Y + \eta$$

and

$$\log_Y(Z) = ZQ'^\top - Y,$$

where η is restricted to the horizontal space $\mathcal{H}_Y = \{\eta \in \mathbb{R}^{p \times q} : \eta^\top Y = Y^\top \eta\}$ and Q' comes from the polar decomposition $Y^\top Z = H'Q'$.

A last element to define is the projection operator

$$\Pi : \mathcal{S}_+(p, k) \rightarrow \mathcal{S}_+(p, q), \quad (4.19)$$

that returns, for a PSD matrix of rank $k \geq q$, the closest PSD matrix of rank q . This last tool will be particularly useful in Section 4.6.

The identification of $\mathcal{S}_+(p, q)$ as the quotient manifold $\mathbb{R}_*^{p \times q} / \mathcal{O}_q$ has already been used in several works in the past. For instance, Bonnabel *et al.* [BMS10] and Meyer *et al.* [MBS11] used this description to perform regression model learning via first-order optimization algorithms on $\mathcal{S}_+(p, q)$; the work of Mishra *et al.* [MMS11] focuses on gradient descent and trust regions methods to tackle the low-rank matrix completion problem ; sparse principal component analysis is studied by Journée *et al.* [JBAS10] using optimization on $\mathcal{S}_+(p, q)$.

4.5.3 Results

The dataset used here is the one of [GMM⁺17], already presented in Section 4.1.3. It is made of 33 covariance matrices $C(\theta_i)$ of size 3024×3024 , $i = 1, \dots, 33$, corresponding to 33 different orientations $\theta_i = (i - 1)\pi/64$. All of them are obtained from unsteady CFD simulations. Note that, for now, the magnitude of the wind field remains fixed. If it weren't, bidimensional methods would be needed, and that will be the point of Chapters 6 and 7. Using a singular value decomposition, the rank of $C(\theta_i)$ is reduced to $q = 20$ by factorizing it as

$$C(\theta_i) \simeq C_i = Y(\theta_i)Y(\theta_i)^\top \in \mathcal{S}_+(3024, 20).$$

Hence $Y_i = Y(\theta_i) \in \mathbb{R}_*^{3024 \times 20}$ (note that it is not even necessary to build $C(\theta_i)$, as $Y(\theta_i)$ can instead be directly obtained from the simulations).

The composite curves $\mathbf{B}(\theta) \in \mathcal{S}_+(3024, 20)$ are computed based on one out of three data points from the data set. This forms the training set

$$S_T = \{C(\theta_i)\}_{i \in I_T},$$

where $I_T = \{1, 4, 7, \dots, 33\}$. The rest of the dataset is used as a validation set

$$S_V = \{C(\theta_i)\}_{i \in I_V},$$

with $I_V = \{2, 3, 5, 6, \dots, 31, 32\}$. The fitting error of $\mathbf{B}(\theta)$ is compared to data from S_Ω ($\Omega \in \{T, V\}$) as a relative mean squared error (MSE) expressed in dB:

$$\text{MSE}(\mathbf{B}(\theta)) = 10 \log \left(\frac{\sum_{i \in I_\Omega} \|C_i - \mathbf{B}(\theta_i)\|_F^2}{\sum_{i \in I_\Omega} \|C_i\|_F^2} \right). \quad (4.20)$$

The evolution of this error with respect to the parameter λ is illustrated in Figure 4.15a. Not surprisingly, the MSE computed on the training set decreases when λ grows, as problem (4.1) is closer and closer to interpolation. Correspondingly, the MSE computed on the validation set at the limit $\lambda \rightarrow \infty$ measures the model error, *i.e.*, the inability of the composite curve to recover the hidden data.

The main advantage of the fitting methods is their robustness to corrupted data. To illustrate this, artificial noise is added to the data. Consider a new matrix $\tilde{C}_i = C(\theta_i) + 0.05N(\theta_i)$ with $N(\theta_i)_{lm} \stackrel{iid}{\sim} \mathcal{N}(0, 1)$ for $l, m = 1, \dots, 3024$. The corrupted matrices \tilde{C}_i are then factorized into $\tilde{C}_i \simeq \tilde{Y}_i \tilde{Y}_i^\top$ similarly as above. This artificial noise results in an $\text{MSE}(\tilde{\mathbf{C}}(\theta))$ of about -9 dB compared the (not corrupted) data points.

The composite curves $\tilde{\mathbf{B}}(\theta)$ are now computed based on the corrupted data from the set $\tilde{S}_T := \{\tilde{C}_i\}_{i \in I_T}$. The MSE of $\tilde{\mathbf{B}}(\theta)$ is computed accordingly and compared to the original data from S_Ω (see Figure 4.15b). One can observe an optimal balance λ_{opt} between data fitting and curve smoothing with about 5 dB of MSE reduction compared to the noise level.

Note also that, as expected, when the training set is made of more data points (like one out of two), the methods perform better, as one can see in Figures 4.16a and 4.16b (here, the result is obtained only with the blended curve). However, the gain is approximately 1 dB only, for a lot of data added to the training set. One can also see that the optimal parameter λ do not change so much from one case to the other.

4 | Fitting with Bézier, blended, and Bézier-like curves

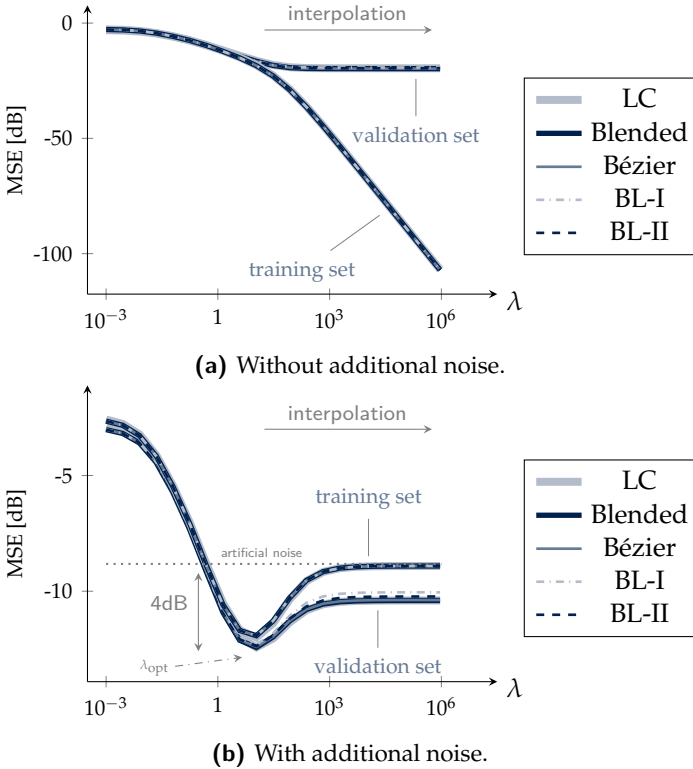
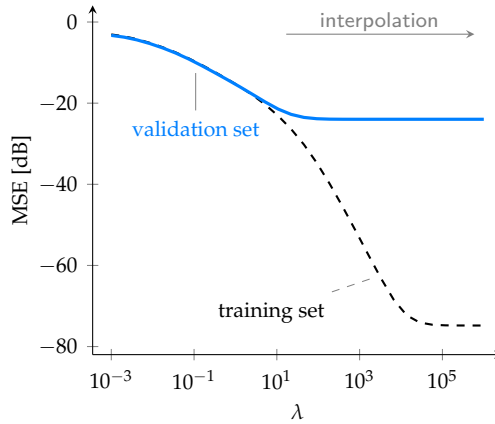
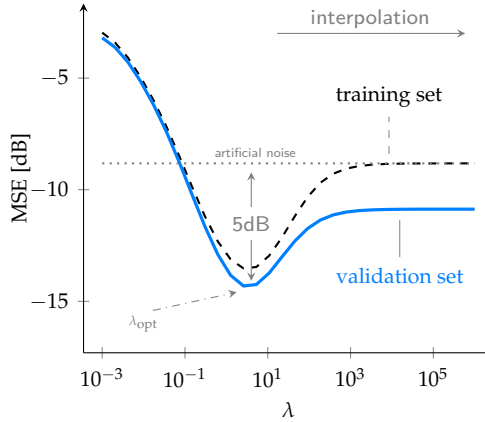


Fig. 4.15 Fitting error of the five methods applied to the wind field problem from [GMM⁺17]. The training set is made of one point out of three from the full dataset; the validation set is made of all remaining data points. The mean square error (MSE) is obtained on the training and validation sets, depending on the value of the regularization parameter λ . All the fitting methods considered behave similarly on this fitting problem. **(a)**: The error on the training set decreases as λ increases (the problem becomes an interpolation problem), while the error on the validation set reaches a constant level (due to the inability of the model to recover the left out data). **(b)**: Data have been corrupted with a Gaussian perturbation applied to the covariance matrices: $C(\theta_i) \leftarrow C(\theta_i) + 0.05N(\theta_i)$, with $N(\theta_i)_{lm} \sim \mathcal{N}(0, 1)$. This artificial noise amounts to adding an MSE of about -9 dB on the training set. The error displayed is computed with respect to the original (non-corrupted) data. As $\lambda \rightarrow \infty$, the error on the training set reaches a constant level (noisy data are interpolated). For $\lambda > 1$, the error on the validation set reaches a smaller value, with an optimal denoising parameter at $\lambda \simeq 10$.



(a) Without additional noise.



(b) With artificial noise.

Fig. 4.16 Fitting error of the blending method applied to the wind field problem from [GMM⁺17]. The difference with Figure 4.15 is that the training set is now made of one point out of **two** from the full dataset. In (a), no corruption is added to the dataset. In (b), as in Figure 4.15, data have been corrupted with a Gaussian perturbation applied to the covariance matrices: $C(\theta_i) \leftarrow C(\theta_i) + 0.05N(\theta_i)$, with $N(\theta_i)_{lm} \sim \mathcal{N}(0,1)$. The artificial noise amounts to adding an MSE of about -9 dB on the training set. The denoising reaches here better values (-5 dB instead of 4 dB), as expected, but the behavior of the method remains similar. The optimal lambda moves to $\lambda \simeq 1$.

4.6 Another application to parametric model order reduction

As a second real-life fitting application, we consider parametric model order reduction. In this application, the goal is to replace very large dynamical system by a drastically smaller one while still representing sufficiently well the physics.

In this application, we consider again data points lying on $\mathcal{S}_+(p, q)$ (see Section 4.5.2). Here, the data points represent positive semidefinite solutions of Lyapunov equations, truncated to a certain rank q , and stored in matrices $P_0, \dots, P_n \in \mathcal{S}_+(p, q)$ and $Q_0, \dots, Q_n \in \mathcal{S}_+(p, q)$. They serve to build projector matrices to reduce the dynamical system to a low dimensional space. They are associated to the abstract parameters μ_0, \dots, μ_n , corresponding to a parameter driving the dynamics of the system.

This application was proposed by Dr. Nguyen Thanh Son (UCLouvain, Belgium) who also provided the dataset used here, and the geometric tools on $\mathcal{S}_+(p, q)$ were provided by Dr. Estelle Massart (UCLouvain, Belgium).

4.6.1 Some words of motivation

MOR Model order reduction (MOR) is a well known tool to simulate large-scale systems [Ant05]. Its purpose is to compute a model of smaller order (by design, faster to simulate) that represents accurately enough the behavior of the full-order system. Indeed, working with the full-order system is often computationally untrackable [LW02]. The reduced-order model is usually obtained via projection-based methods such as balanced truncation (further described in Section 4.6.2), proper orthogonal decomposition, Krylov subspace-based moments matching or \mathcal{H}_2 -norm optimization [Ant05]. Those methods provide two matrices V_{Proj} and W_{Proj} that project the large-scale system into a smaller state space.

PMOR In many cases, the system depends on parameters representing, *e.g.*, physical, material or environmental properties [Wik]. Parametric model order reduction (PMOR) computes a parameterized reduced model that approximates the behavior of the full-order system in a given parameter range. The example of Section 4.5 follows a logic close to parametric model order reduction, except that, here, no projectors are computed: only low-rank matrices are manipulated.

A classical technique of PMOR is made of two steps: an offline step,

which can take as much time as necessary, and an online step, which is supposed to run fast. In the offline step, the reduced-order models are obtained for a subset of parameter values. These models are afterwards used to recover the reduced models for other parameter values by interpolation during the online step. The interpolation is done on representatives of the models, *e.g.*, the projection matrices V_{Proj} and W_{Proj} , the reduced-system matrices, or the reduced transfer functions. For more details, see [BGW15].

In this application, the step of interest is of course the online step, as the main advantages of the methods presented in this chapter is their computational efficiency. For the offline step, the chosen reduction method is balanced truncation. The most computationally expensive part of this algorithm is the evaluation of the solutions P and Q to a pair of Lyapunov equations. Those solutions can, in fact, be approximated by low-rank positive semi-definite matrices, *i.e.*, points on $\mathcal{S}_+(p, q)$. Those low-rank approximations are thus computed for a few values of the parameters (in the offline step). Then, for other values of the parameters, the pre-computed solutions are interpolated (during the online step) on $\mathcal{S}_+(p, q)$.

 $\mathcal{S}_+(n, p)$

4.6.2 Theory on parametric model order reduction

The theory on PMOR and balanced truncation can be found in details in the work of Antoulas [Ant05]. This section is just a quick summary.

Consider an asymptotically stable linear parameterized system:

$$\begin{aligned} E(\mu)\dot{x}(t, \mu) &= A(\mu)x(t, \mu) + B(\mu)u(t), \\ y(t, \mu) &= C(\mu)x(t, \mu), \end{aligned} \quad (4.21)$$

where $\mu \in [\alpha, \beta]$ is a parameter representing, for instance, physical, material or environmental properties; $E(\mu) \in \mathbb{R}^{p \times p}$ is nonsingular, $A(\mu) \in \mathbb{R}^{p \times p}$, $B(\mu) \in \mathbb{R}^{p \times m}$, $C(\mu) \in \mathbb{R}^{s \times p}$, for $s, m \ll p$. The vectors $x(t, \mu) \in \mathbb{R}^p$, $u(t) \in \mathbb{R}^m$ and $y(t, \mu) \in \mathbb{R}^s$ are respectively the state, the input and the output vectors of the system. The goal of PMOR is to approximate system (4.21) with a parametric reduced-order model

$$\begin{aligned} \tilde{E}(\mu)\dot{\tilde{x}}(t, \mu) &= \tilde{A}(\mu)\tilde{x}(t, \mu) + \tilde{B}(\mu)u(t), \\ \tilde{y}(t, \mu) &= \tilde{C}(\mu)\tilde{x}(t, \mu), \end{aligned} \quad (4.22)$$

where $\tilde{x}(t, \mu) \in \mathbb{R}^r$ is the reduced state vector, $\tilde{y}(t, \mu) \in \mathbb{R}^s$ is the approximated output, $\tilde{E}(\mu)$ and $\tilde{A}(\mu)$ are now instances of $\mathbb{R}^{r \times r}$, $\tilde{B}(\mu) \in \mathbb{R}^{r \times m}$,

4 | Fitting with Bézier, blended, and Bézier-like curves

$\tilde{C}(\mu) \in \mathbb{R}^{s \times r}$. Of course, $r \ll p$. In the following, the μ -dependency will be omitted for readability.

Balanced truncation is a three-step method to compute the reduced-order model. First, one has to find the low-rank approximate solutions $P = XX^\top$ and $Q = YY^\top$ of the Lyapunov equations

$$\begin{aligned} EPA^\top + APE^\top &= -BB^\top, \\ E^\top QA + A^\top QE &= -C^\top C, \end{aligned} \quad (4.23)$$

where $X \in \mathbb{R}^{p \times k_X}$, $Y \in \mathbb{R}^{p \times k_Y}$, in which k_X and k_Y depend on μ as well. For this application, equations (4.23) are solved with a low-rank ADI solver detailed in [LW02]. Secondly, the singular value decomposition $Y^\top EX = U\Sigma V^\top$ has to be computed. This SVD is truncated to a given rank r such that $\tilde{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_r)$ is composed of the r first largest singular values $\{\sigma_i\}_{i=1}^r$, and \tilde{U} and \tilde{V} are respectively the truncation of U and V to their r first columns. The projection matrices of the reduced-order model are then given by

$$\begin{aligned} W_{\text{Proj}} &= Y\tilde{U}\tilde{\Sigma}^{-1/2}, \\ V_{\text{Proj}} &= X\tilde{V}\tilde{\Sigma}^{-1/2}. \end{aligned} \quad (4.24)$$

Finally, the reduced model (4.22) is obtained by projection: $\tilde{E} = W_{\text{Proj}}^\top EV_{\text{Proj}}$, $\tilde{A} = W_{\text{Proj}}^\top AV_{\text{Proj}}$, $\tilde{B} = W_{\text{Proj}}^\top B$, and $\tilde{C} = CV_{\text{Proj}}$.

The truncation rank r is generally not chosen randomly, as it is related to the error between the (Laplace) transfer function $\tilde{H}(s, \mu)$ of the reduced model (4.22) and $H(s, \mu)$, the one of the full-order model (4.21). The Laplace transfer function $H(s)$ of a system gives the relation between $X(s)$, the Laplace transform of the state vector, and $Y(s)$, the Laplace transform of the output vector. The relation is given by $Y(s) = H(s)X(s)$. In this case, the transfer function of the full-order system is

$$H(s, \mu) = C(\mu)(sE(\mu) - A(\mu))^{-1}B(\mu),$$

and accordingly for \tilde{H} . That way, it can be chosen relatively to a given tolerance ϵ as

$$\|H - \tilde{H}\|_{\mathcal{H}_\infty} \leq 2(\sigma_{r+1} + \dots + \sigma_{\min(k_X, k_Y)}) < \epsilon.$$

For a transfer function G , $\|G\|_{\mathcal{H}_\infty}$ is the supremum, over the frequencies, of the magnitude of G .

Instead of computing the solutions P and Q of (4.23) for each value of

μ , some solutions P_i and Q_i are precomputed; they are associated to some parameter values μ_i , $i = 0, \dots, n$. Then, the solutions P (respectively Q) associated to other parameter values are estimated by interpolation of the P_i (respectively the Q_i). The precomputed solutions P_i and Q_i are positive semi-definite and have a rank k_{X_i} and k_{Y_i} , respectively. As the set of all positive semi-definite matrices is not a manifold [MA18], the rank of the matrices must be truncated to a given rank q , such that the data points will all belong to $\mathcal{S}_+(p, q)$. For instance, to interpolate the P_i , one chooses $q = \min(\{k_{X_i}\}_{i=0}^n)$. This step induces of course a loss of information, but which turns out to be mild in practice. Hence, the interpolated points are $\tilde{P}_i := \tilde{X}_i \tilde{X}_i^\top$, where \tilde{X}_i is made of the q first columns of X_i . By the design of the low-rank ADI method [LW02], these columns contain the dominant information of the low-rank solution. To interpolate the Q_i , the rank q is chosen as $q := \min(\{k_{Y_i}\}_{i=0}^n)$.

Finally, once the interpolant \tilde{P} and \tilde{Q} , associated to μ , are known, the construction of the reduced system (4.22) is considerably cheap, compared to the resolution of the Lyapunov equations.

4.6.3 Methods used for comparison

For this application, the composite cubic blended curve will be compared to four other interpolation methods presented hereunder. Consider that the points to interpolate are for now points $d_0, \dots, d_n \in \mathcal{M}$ and associated to times $t_0 < t_1 < \dots < t_n \in \mathbb{R}$.

Interpolation in the ambient space. This first approach consists in running Euclidean interpolation algorithms in the ambient space in which \mathcal{M} is embedded (i.e., $\mathbb{R}^{p \times p}$ for this application). This means that points are interpolated disregarding their geometry. The result is then projected back on \mathcal{M} , using a projection operator $\Pi_{\mathcal{M}}$ (in this case, the projector (4.19)). The curve will be noted $\mathbf{P}_{\text{AS}}^{\text{lin}}(t)$ and corresponds to a piecewise **linear** interpolation in the **Ambient Space**.

Interpolation with a local curve. A second approach consists in classical a three-step procedure, also known as the exp-log algorithm: (i) all the data points are mapped to a tangent space based at an arbitrary point d_{ref} , using the logarithm map $\log_{d_{\text{ref}}}$; (ii) Euclidean interpolation is performed on those points (for instance, with a cubic spline); (iii) the result is finally

4 | Fitting with Bézier, blended, and Bézier-like curves

mapped back on the manifold using the exponential map $\exp_{d_{\text{ref}}}$. The curve $\mathbf{P}_{\text{LC}}^{\text{cub}}(t)$ is obtained with this strategy. It is noted like this because interpolation is done with a **cubic Local Bézier Curve**, computed on a given tangent space $T_{d_{\text{ref}}}\mathcal{M}$. Two possible choices are considered for the tangent space: $d_{\text{ref}} = d_0$ or $d_{\text{ref}} = d_{\lfloor (n)/2 \rfloor}$. Note that this method is a degenerate version of the composite cubic blended curve, where the same tangent space is considered for all pieces.

Piecewise geodesic splines and blended curves. Just like blended curves do not perform all the computations on one single tangent space (which can generate significant deviations when some points are far from the root of the tangent space), the **Piecewise Geodesic spline** $\mathbf{P}_{\text{PC}}(t)$ consists in a concatenation of geodesics between two consecutive data points. The composite cubic blended curve will be noted $\mathbf{P}_{\text{Blend}}(t)$ and is run for $\lambda = 10^8$ (*i.e.*, in its interpolating version).

4.6.4 Results

In this experiment, the full-order model (4.21) is the one-parameter anemometer [Wik] where $p = 29008$, and $\mu \in [0, 1]$ corresponds to the fluid velocity. The low-rank ADI method [LW02] is run to solve the Lyapunov equations (4.23) (with tolerance $\epsilon = 10^{-8}$) for 21 values of the parameter μ given by $\{\mu_1, \dots, \mu_{21}\} = \{0, 0.05, \dots, 1\}$. This provides the training set $\{P_1^{\text{ADI}}, \dots, P_{21}^{\text{ADI}}\}$. The rank of the solutions returned by the solver varies from 25 to 39, so that the common rank is fixed to $q = 25$. The results compare the ability of the five interpolation methods to recover P^{ADI} at test values for μ .

Figure 4.17a shows the relative error $E_{\text{rel}}(\mu)$ between the predicted matrices $P(\mu)$ and the ADI solutions $P^{\text{ADI}}(\mu)$, truncated to a rank $q = 25$:

$$E_{\text{rel}}(\mu) = \frac{\|P(\mu) - P^{\text{ADI}}(\mu)\|_{\text{F}}}{\|P^{\text{ADI}}(\mu)\|_{\text{F}}}.$$

The test set is made of 40 points, and has no intersection with the training set, for which E_{rel} is zero. The best trade-off between computation time and accuracy seems to be the cubic interpolation in the tangent space $\mathbf{P}_{\text{LC}}^{\text{cub}}(\mu)$, choosing P_{ref} as the midpoint of the data set (LC3-M). However, this method is sensitive to the choice of the tangent space: when P_{ref} is set to P_1 this leads to significantly larger errors (LC3-1). The blended curve,

which intrinsically combines several tangent spaces, does not present this drawback, and reaches the same accuracy as LC3-M. The fact that $\mathbf{P}_{LC}^{cub}(\mu)$ and $\mathbf{P}_{Blend}(\mu)$ are similar when the tangent space is based at the midpoint of the data set indicates that the curvature of the manifold is small around the data points considered. For the same reason, the piecewise linear interpolation in the ambient space $\mathbf{P}_{AS}^{lin}(\mu)$ (LinP) is almost as accurate as the piecewise geodesic spline $\mathbf{P}_{PG}(\mu)$ (PG).

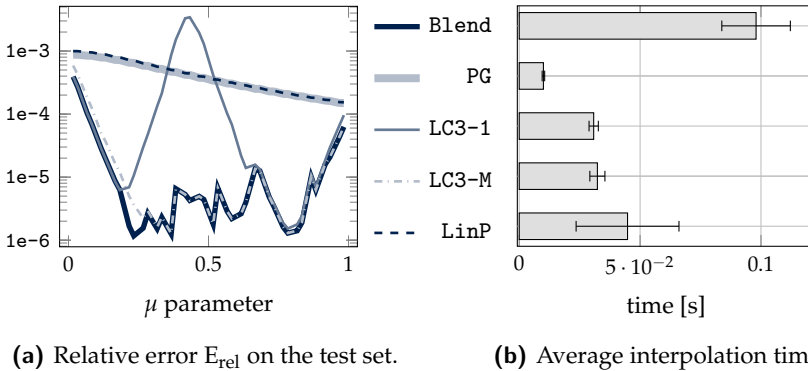


Fig. 4.17 Comparison of the five interpolation methods considered. The training set is made of 21 values of the parameter μ given by $\{\mu_1, \dots, \mu_{21}\} = \{0, 0.05, \dots, 1\}$. The test set is made of 40 values of μ and has no intersection with the training set for which the relative error E_{rel} is zero. The composite cubic blended curve performs best but requires the highest effort of computation; however, its result are not sensitive to the choice of the reference tangent space on which data are projected. In comparison, when the reference point is well chosen, the exp-log solution is extremely fast and reliable, while extremely bad when it is badly chosen.

Figure 4.17b compares the computation times required to obtain a value of the different curves at one arbitrary value of μ and during the *online* phase. That is, all the computations independent of μ were done offline and not considered in those timings. The curve $\mathbf{P}_{Blend}(\mu)$ is the most expensive one. However, those values have to be compared with the average time to compute the matrix $P^{ADI}(\mu)$ using the ADI solver (*i.e.*, without interpolation), which is here around 10s. Hence, the blending procedure represents an improvement of approximately a factor 100 compared to the ADI solution; all other methods take half of the time needed by the blended curve.

This chapter is mainly based on the paper [GMA18c] for the theoretical parts, and on [GMM⁺17] and [MGS⁺19] for applications in Sections 4.5 and 4.6 respectively. They are sometimes cited verbatim.

The references of these papers are

[GMA18c] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Data fitting on manifolds with composite Bézier-like curves and blended cubic splines. *Journal of Mathematical Imaging and Vision*, 61(5):645–671, 2018. doi:10.1007/s10851-018-0865-2

[GMM⁺17] Pierre-Yves Gousenbourger, Estelle Massart, Antoni Musolas, P.-A. Absil, Laurent Jacques, Julien M Hendrickx, and Youssef Marzouk. Piecewise-Bézier C^1 smoothing on manifolds with application to wind field estimation. In *ESANN2017*, pages 305–310. Springer, 2017

and

[MGS⁺19] Estelle Massart, Pierre-Yves Gousenbourger, Nguyen Thanh Son, Tatjana Stykel, and P.-A. Absil. Interpolation on the manifold of fixed-rank positive-semidefinite matrices for parametric model order reduction: preliminary results. In *ESANN2019*, pages 281–286. Springer, 2019

The figures can be reproduced based on the code provided in the toolbox available at this link address:

<https://github.com/pgousenbourg/manint>

5

Optimality of the Bézier fitting curve

IN THE TWO PREVIOUS CHAPTERS, techniques for fitting and interpolation have been presented to approximate the acceleration-minimizing curve passing close to the data points. The focus was oriented to the computational efficiency of the methods rather than to the quality of the minimizing curve.

This chapter intends to evaluate the quality of the fitting (or interpolating) composite Bézier curves from Definitions 3.20 and 4.1, compared to the manifold-valued optimization problem

$$\min_{\mathbf{B}} \int_0^n \left\| \frac{D^2 \mathbf{B}(t)}{dt^2} \right\|_{\mathbf{B}(t)}^2 dt + \frac{\lambda}{2} \sum_{i=0}^n d^2(\mathbf{B}(i), d_i). \quad (5.1)$$

In this chapter, each data point d_i is associated to an integer time i , for simplicity. This is the next contribution of this thesis.

The principal result of the chapter is the derivation of the gradient of the differentiable composite Bézier curve $\mathbf{B}: [t_0, t_n] \rightarrow \mathcal{M}$ that satisfies (5.1), while fitting the set of manifold-valued data points at their associated time-parameters. By extension, the gradient of (5.1) itself is derived.

The approximating model to (5.1) is obtained in the following way. The Levi-Civita second covariant derivative is approximated by a discretized

(squared) second order absolute differences introduced in [BBSW16]; the quality of the approximation depends only on the number of sampling points. The gradient is then built as a recursion of Jacobi fields that, for numerical reasons, are implemented as a concatenation of so-called *adjoint Jacobi fields*. At the end, the philosophy of the previous chapters is preserved, as the optimization algorithm is only based on three tools on the manifold: the exponential map, the logarithmic map, and a Jacobi field along geodesics.

The first section of the chapter (Section 5.1) defines quickly the *discrete mean squared acceleration* (MSA) that approximates the regularizer of (5.1); it also gives some additional mathematical elements to complete Chapter 2. Then, Section 5.2 is dedicated to the derivation of the gradient of the MSA of a composite Bézier curve with respect to its control points.

In Section 5.3, we present the corresponding gradient descent algorithm, as well as an efficient gradient evaluation method, to solve (5.1) for different values of λ . The limit case where $\lambda \rightarrow \infty$ (interpolation) is studied as well. Finally, in Section 5.4, we validate, analyze and illustrate the performances of the algorithm for several numerical examples on the sphere S^2 and on the special orthogonal group $SO(3)$. We also compare the solution to the curves obtained via Definitions 3.20 and 4.1.

This chapter is the result of a collaboration with Dr. Ronny Bergmann (Technische Universität Chemnitz, Germany).

5.1 Some additional mathematical elements

Evaluating the Levi-Civita second covariant derivative of a curve $\gamma: [0, 1] \rightarrow \mathcal{M}$ is still a complicated task, as already mentioned in the previous chapters. The idea here is to avoid this difficulty by discretizing $\gamma(t)$ and approximating its MSA

$$\int_0^1 \left\| \frac{D^2 \gamma(t)}{dt^2} \right\|_{\gamma(t)}^2 dt, \quad (5.2)$$

with the second order absolute finite differences introduced by Bačák *et al.* [BBSW16].

Consider three points $x, y, z \in \mathcal{M}$ and the set of mid-points of x and z

$$\mathcal{C}_{x,z} := \left\{ c \text{ such that } c = g\left(\frac{1}{2}; x, z\right) \right\},$$

for all (not necessarily shortest) geodesics $g(\cdot; x, z)$ connecting x and z . The manifold-valued second order absolute finite differences are defined by

$$d_2[x, y, z] = \min_{c \in \mathcal{C}_{x,z}} 2d_{\mathcal{M}}(c, y). \tag{5.3}$$

It is equivalent, on the Euclidean space, to $\|x - 2y + z\| = 2\|\frac{1}{2}(x + z) - y\|$. This definition is illustrated in Figure 5.1.

finite differences

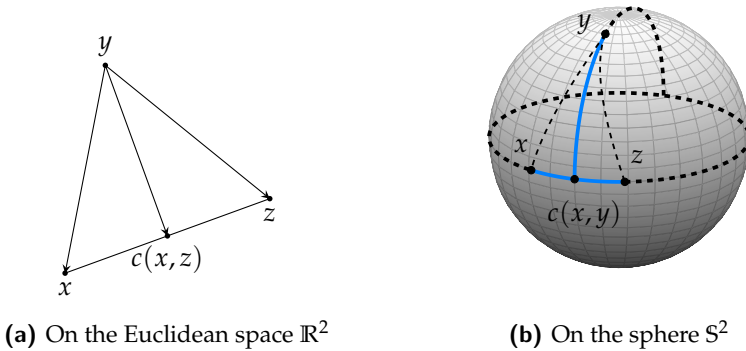


Fig. 5.1 Geometric interpretation of the second order finite differences.

The time domain $[0, n]$ is discretized by $N \in \mathbb{N}$ equispaced points t_0, \dots, t_N , with step size $\Delta_t = t_1 - t_0 = \frac{1}{N}$. The normed acceleration is approximated by

$$\left\| \frac{D^2 \gamma(t_i)}{dt^2} \right\|_{\gamma(t_i)} \approx \frac{1}{\Delta_t^2} d_2[\gamma(t_{i-1}), \gamma(t_i), \gamma(t_{i+1})], \quad i = 1, \dots, N - 1.$$

The discretization times t_k must not be confused with the parameters $t_i = i$ associated with the data points.

By the trapezoidal rule, the MSA is given by

$$\int_0^1 \left\| \frac{D^2 \gamma(t)}{dt^2} \right\|_{\gamma(t)}^2 dt \approx \sum_{i=1}^{N-1} \frac{\Delta_t d_2^2[\gamma(t_{i-1}), \gamma(t_i), \gamma(t_{i+1})]}{\Delta_t^4}.$$

For (composite) Bézier curves $\gamma(t) = \mathbf{B}(t)$, the regularizer of (5.1) depends on the control points \mathbf{b} . The discretized MSA $A(\mathbf{b})$ finally reads

discretized MSA

$$A(\mathbf{b}) := \sum_{i=1}^{N-1} \frac{d_2^2[\mathbf{B}(t_{i-1}), \mathbf{B}(t_i), \mathbf{B}(t_{i+1})]}{\Delta_t^3}. \tag{5.4}$$

This chapter will extensively use the notions of directional derivatives and gradients (defined in Chapter 2). However, one has to introduce several short-hand notations that are specific to this chapter. Let $f(x, y) : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$ be a multivariate functions. In that case, the variable along which the directional derivative is computed must be specified. We denote by $D_x f[\xi](x, y) \in T_{f(x,y)}\mathcal{M}$ the directional derivative of f evaluated at (x, y) , with respect to its variable x and in the direction $\xi \in T_x\mathcal{M}$. We use the short hand $D_x f[\xi] = D_x f[\xi](x, y)$ whenever this directional derivative is evaluated afterwards again at x . Identically, the x -component of the gradient of f at (x, y) will be denoted by $\nabla_{\mathcal{M},x} f(x, y)$. We shorten this notation as $\nabla_{\mathcal{M},x} f = \nabla_{\mathcal{M},x} f(x, y)$ when this gradient is seen as a function of x and y .

The notion of chain rule on manifolds will also be a core element. The following definition is extracted from [AMS08, p. 195].

Definition 5.1 (*Chain rule*). Let $f: \mathcal{M} \rightarrow \mathcal{M}, h: \mathcal{M} \rightarrow \mathcal{M}$ be two functions on a manifold \mathcal{M} and $F: \mathcal{M} \rightarrow \mathcal{M}, x \mapsto F(x) = (f \circ h)(x) = f(h(x))$, their composition. Let $x \in \mathcal{M}$ and $\eta \in T_x\mathcal{M}$. The directional derivative $D_x F[\eta]$ of F with respect to x in the direction η is given by

$$D_x F[\eta] = D_{h(x)} f [D_x h[\eta]], \tag{5.5}$$

where $D_x h[\eta] \in T_{h(x)}\mathcal{M}$ and $D_x F[\eta] \in T_{F(x)}\mathcal{M}$.

5.2 Gradient of the discretized mean squared acceleration

In order to minimize the discretized MSA $A(\mathbf{b})$, we aim to employ a gradient descent algorithm on the product manifold \mathcal{M}^M , where M is the number of elements in \mathbf{b} . In the following, we derive a closed form of the gradient $\nabla_{\mathcal{M},b_i} A(\mathbf{b})$ of the discretized MSA (5.4). This gradient is obtained by means of a recursion and the chain rule. In fact, the derivative of (5.3) is already known [BBSW16], such that it only remains to compute the derivative of the composite Bézier curve.

The section is organized in four parts. We first recall the theory on Jacobi fields in Section 5.2.1 and their relation to the differential of geodesics (with respect to start and end point). As geodesics are the principal ingredient of the De Casteljau algorithm, Jacobi fields become the principal ingredient in the derivation of a Bézier curve. In Section 5.2.2, we apply the

chain rule to the composition of two geodesics, which appears within the De Casteljau algorithm. We use this result to build an algorithmic derivation of the differential of a general Bézier curve on manifolds with respect to its control points (Section 5.2.3). We extend the result to composite Bézier curves in Section 5.2.4, including their constraints on junction points p_i to enforce the C^1 conditions from Property 3.16, and finally gather these results to state the gradient $\nabla_{\mathcal{M}}A(\mathbf{b})$ of the discretized MSA (5.4) with respect to the control points.

5.2.1 Jacobi fields as derivative of a geodesic

The differential $D_x g(t; \cdot, y)$ of a geodesic $g(t; x, y), t \in [0, 1]$, with respect to its start point $x \in \mathcal{M}$, can be expressed using the notion of Jacobi fields. Conversely, the differential with respect to the end point $y \in \mathcal{M}$ of the geodesic is simply obtained by taking the reversed geodesic $g(t, y, x) = g(1 - t; x, y)$.

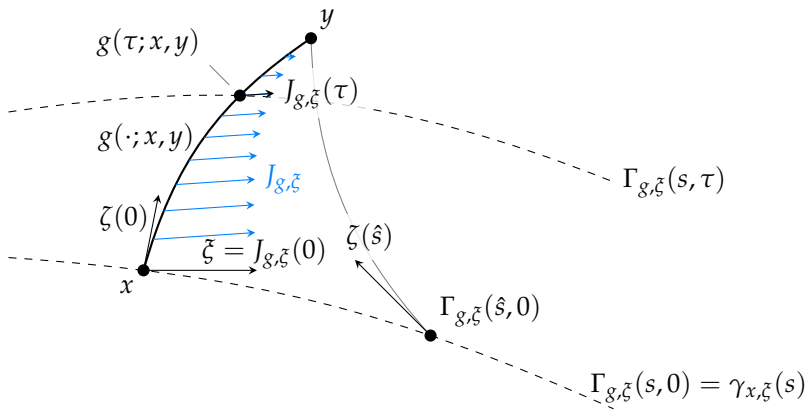


Fig. 5.2 Schematic representation of the variation $\Gamma_{g, \zeta}(s, t)$ of a geodesic g w.r.t. the direction $\zeta \in T_x \mathcal{M}$. The corresponding Jacobi field along g and in the direction ζ is the vector field $J_{g, \zeta}(t) = \frac{\partial}{\partial s} \Gamma_{g, \zeta}(s, t)|_{s=0}$.

We denote by $\gamma_{x, \zeta}$, the geodesic starting in $\gamma_{x, \zeta}(0) = x$ and with direction $\frac{D}{dt} \gamma_{x, \zeta}(0) = \zeta \in T_x \mathcal{M}$. We introduce $\zeta(s) := \log_{\gamma_{x, \zeta}(s)} y$, the tangential vector in $T_{\gamma_{x, \zeta}(s)} \mathcal{M}$ pointing towards y . Then, the *geodesic variation* $\Gamma_{g, \zeta}(s, t)$ of the geodesic $g(\cdot; x, y)$ with respect to the tangential direction $\zeta \in T_x \mathcal{M}$

geodesic
variation $\Gamma_{g, \zeta}$

5 | Optimality of the Bézier fitting curve

is given by

$$\Gamma_{g,\xi}(s,t) := \exp_{\gamma_{x,\xi}(s)}(t\zeta(s)), \quad s \in (-\varepsilon, \varepsilon), t \in [0,1],$$

where $\varepsilon > 0$. The corresponding Jacobi field $J_{g,\xi}$ along g , represented in Figure 5.2, is then given by the vector field

$$J_{g,\xi}(t) := \left. \frac{D}{ds} \Gamma_{g,\xi}(s,t) \right|_{s=0}$$

that represents the direction of the displacement of g if x is perturbed in a direction ξ .

We directly obtain $J_{g,\xi}(0) = \xi$, and $J_{g,\xi}(1) = 0$ as well as $J_{g,\xi}(t) \in T_{g(t;x,y)}\mathcal{M}$. Furthermore, since $\Gamma_{g,\xi}(s,t) = g(t; \gamma_{x,\xi}(s), y)$ the chain rule gives

$$D_x g(t, \cdot, y)[\xi] = \left. \frac{D}{ds} g(t; \gamma_{x,\xi}(s), y) \right|_{s=0} = \left. \frac{D}{ds} \Gamma_{g,\xi}(s,t) \right|_{s=0} = J_{g,\xi}(t). \quad (5.6)$$

In symmetric spaces, Jacobi fields can be computed in closed form, as summarized in the next Lemma. This gives a convenient way to compute derivatives of geodesics in such spaces.

Lemma 5.2. [BBSW16, Prop. 3.5] *Let \mathcal{M} be a m -dimensional Riemannian manifold. Let $g(t; x, y)$, $t \in [0, 1]$, be a geodesic between $x, y \in \mathcal{M}$, $\eta \in T_x\mathcal{M}$ be a tangent vector and $\{\xi_1, \dots, \xi_m\}$ be an orthonormal basis (ONB) of $T_x\mathcal{M}$ that diagonalizes the curvature operator of \mathcal{M} with eigenvalues κ_ℓ , $\ell = 1, \dots, m$. For details, see [dC92, Chap. 4.2 and 5, (Ex. 5)]. Let further denote by $\{\Xi_1(t), \dots, \Xi_m(t)\}$ the parallel transported frame of $\{\xi_1, \dots, \xi_m\}$ along g .*

Decomposing $\eta = \sum_{\ell=1}^m \eta_\ell \xi_\ell \in T_x\mathcal{M}$, the derivative $D_x g[\eta]$ becomes

$$D_x g(t; x, y)[\eta] = J_{g,\eta}(t) = \sum_{\ell=1}^m \eta_\ell J_{g,\xi_\ell}(t),$$

where the Jacobi field $J_{g,\xi_\ell} : \mathbb{R} \rightarrow T_{g(t;x,y)}\mathcal{M}$ along g and in the direction ξ_ℓ is given by

$$J_{g,\xi_\ell}(t) = \begin{cases} \frac{\sinh(d_g(1-t)\sqrt{-\kappa_\ell})}{\sinh(d_g\sqrt{-\kappa_\ell})} \Xi_\ell(t) & \text{if } \kappa_\ell < 0, \\ \frac{\sin(d_g(1-t)\sqrt{\kappa_\ell})}{\sin(d_g\sqrt{\kappa_\ell})} \Xi_\ell(t) & \text{if } \kappa_\ell > 0, \\ (1-t)\Xi_\ell(t) & \text{if } \kappa_\ell = 0, \end{cases} \quad (5.7)$$

with $d_g = d_{\mathcal{M}}(x, y)$ denoting the length of the geodesic $g(t; x, y)$, $t \in [0, 1]$.

The Jacobi field of the reversed geodesic $\bar{g}(t) := g(t; y, x) = g(1 - t; x, y)$ is obtained using the same orthonormal basis and transported frame but evaluated at $s = 1 - t$. We thus obtain $D_y g(t; x, y)[\xi_\ell] = D_y g(1 - t; y, x)[\xi_\ell] = J_{\bar{g}, \xi_\ell}(1 - t)$, where

$$J_{\bar{g}, \xi_\ell}(1 - t) = \begin{cases} \frac{\sinh(d_g t \sqrt{-\kappa_\ell})}{\sinh(d_g \sqrt{-\kappa_\ell})} \Xi_\ell(t) & \text{if } \kappa_\ell < 0, \\ \frac{\sin(d_g t \sqrt{\kappa_\ell})}{\sin(d_g \sqrt{\kappa_\ell})} \Xi_\ell(t) & \text{if } \kappa_\ell > 0, \\ t \Xi_\ell(t) & \text{if } \kappa_\ell = 0. \end{cases}$$

Note that $T_{g(t)}\mathcal{M} = T_{\bar{g}(1-t)}\mathcal{M}$. Therefore $\Xi_\ell(t) \in T_{g(t)}\mathcal{M}$, $\ell = 1, \dots, m$, is an orthonormal basis for this tangent space.

5.2.2 Derivative of coupled geodesics

Let \mathcal{M} be a symmetric Riemannian manifold. The result of Lemma 5.2 is used to directly compute the derivative of *coupled geodesics*, i.e., a function composed of $g_1(t) := g(t; x, y)$ and $g_2(t) := g(t; g_1(t), z)$. By Definition 5.1, we have

$$D_x g_2(t)[\eta] = D_{g_1(t)} g(t; \cdot, z) [D_x g_1(t)[\eta]]$$

and by (5.6), we obtain

$$D_x g_2(t)[\eta] = J_{g_2, D_x g_1(t)[\eta]}(t),$$

where the direction of variation used in the Jacobi field is now the derivative of $g_1(t)$ in direction η . Similarly, we compute the derivative of a reversed coupled geodesic $g_3(t) := g(t; z, g_1(t))$ as

$$D_x g_3(t)[\eta] = D_{g_1(t)} g(t; z, \cdot) [D_x g_1(t)[\eta]] = J_{\bar{g}_3, D_x g_1(t)[\eta]}(1 - t).$$

Note that the Jacobi field is here reversed, but that its direction of variation is the same as the one of the Jacobi field introduced for $g_2(t)$. In a computational perspective, it means that we can use the same ONB for the derivatives of both g_3 and \bar{g}_3 . Furthermore, in this case, the direction of variation is also computed by a Jacobi field since $D_x g_1(t)[\eta] = J_{g_1, \eta}(t)$.

To summarize, the derivative of g_2 (resp. g_3) on symmetric spaces is obtained as follows. Let $\{\tilde{\zeta}_1^{[1]}, \dots, \tilde{\zeta}_m^{[1]}\}$ be an ONB of $T_x \mathcal{M}$ for the inner Ja-

cobi field along g_1 , and $\{\zeta_1^{[2]}, \dots, \zeta_m^{[2]}\}$ be an ONB of $T_{g_1(t)}\mathcal{M}$ for the outer Jacobi field along g_2 (resp. g_3). As $\eta = \sum_{\ell=1}^m \eta_\ell \zeta_\ell^{[1]} \in T_x\mathcal{M}$, and stating $J_{g_1, \zeta_\ell^{[1]}}(t) = \sum_{l=1}^m \mu_l \zeta_l^{[2]} \in T_{g_1(t)}\mathcal{M}$, the derivative of g_2 (resp. g_3) with respect to x in the direction $\eta \in T_x\mathcal{M}$ reads

$$D_x g_2(t)[\eta] = \sum_{l=1}^m \sum_{\ell=1}^m J_{g_2, \zeta_l^{[2]}}(t) \mu_l \eta_\ell, \tag{5.8}$$

and accordingly for g_3 .

5.2.3 Derivative of a Bézier curve

Sections 5.2.1 and 5.2.2 introduced the necessary concepts to compute the derivative of a general Bézier curve $\beta_K(t; b_0, \dots, b_K)$ (see Definition 3.12) with respect to its control points b_j . For readability of the recursive structure investigated in the following, we introduce a slightly simpler notation and the following setting.

Let K be the degree of the Bézier curve $\beta_K(t; b_0, \dots, b_K)$. We fix $k \in \{1, \dots, K\}$, $i \in \{0, \dots, K - k\}$ and $t \in [0, 1]$. We introduce

$$g_i^{[k]}(t) := g(t; g_i^{[k-1]}(t), g_{i+1}^{[k-1]}(t)) = \beta_i^{[k]}(t; b_i, \dots, b_{i+k}), \tag{5.9}$$

for the i^{th} Bézier curve of degree k in the De Casteljau algorithm, and $g_i^{[0]}(t) = b_i$. Furthermore, given $x \in \{b_i, \dots, b_{i+k}\}$, we denote by

$$\eta_i^{[k]} := D_x g_i^{[k]}(t)[\eta], \tag{5.10}$$

its derivative with respect to one of its control points x in the direction $\eta \in T_x\mathcal{M}$.

Remark 5.3. Clearly any other derivative of $g_i^{[k]}$ with respect to $x = b_j$, $j < i$ or $j > i + k$ is zero. In addition we have $\eta_i^{[0]} = D_x g_i^{[0]}[\eta] = \eta$ for $x = b_i$ and zero otherwise.

Theorem 5.4 (Derivative of a Bézier curve). *Let $k \in \{1, \dots, K\}$ and $i \in \{0, \dots, K - k\}$ be given. The derivative $\eta_i^{[k]} = D_x g_i^{[k]}(t)[\eta]$ of $g_i^{[k]}$ with respect to its control point $x := b_j$, $i \leq j \leq i + k$, and in the direction $\eta \in T_x\mathcal{M}$ is given*

by

$$\eta_i^{[k]} := D_x g_i^{[k]}(t)[\eta] = \begin{cases} J_{g_i^{[k]}, \eta_i^{[k-1]}}(t) & \text{if } j = i, \\ J_{g_i^{[k]}, \eta_i^{[k-1]}}(t) + J_{\bar{g}_i^{[k]}, \eta_{i+1}^{[k-1]}}(1-t) & \text{if } i < j < i+k, \\ J_{\bar{g}_i^{[k]}, \eta_{i+1}^{[k-1]}}(1-t) & \text{if } j = i+k. \end{cases}$$

Proof. Let $t \in [0, 1]$ and $x = b_j$, $i \leq j \leq i+k$. For readability we set $a := g_i^{[k-1]}(t)$, $b := g_{i+1}^{[k-1]}(t)$, and $f := g_i^{[k]}(t) = g(t; a, b)$. Note that while f depends on the control points b_i, \dots, b_{i+k} and is a Bézier curve of degree k , both a and b are Bézier curves of degree $k-1$. The former does not depend on b_{i+k} , and the latter is independent of b_i .

We prove the claim by induction. For $k = 1$ the function $g_i^{[1]}$ is just a geodesic. The case $i < j < i+1$ does not occur and the remaining first and third cases follow by the notation introduced for $k = 0$ and Lemma 5.2.

For $k > 1$ we apply the chain rule (5.5) to $D_x f[\eta]$ and obtain

$$D_x f[\eta] = D_a f[D_x a[\eta]] + D_b f[D_x b[\eta]].$$

Consider the first term $D_a f[D_x a[\eta]]$ and $j < i+k$. By (5.6) and the notation from (5.10), one directly has

$$D_a f[\eta_i^{[k-1]}] = J_{f, \eta_i^{[k-1]}}(t).$$

For $j = i+k$, clearly $D_x a[\eta] = D_a f[D_x a[\eta]] = 0$, as a does not depend on b_{i+k} .

We prove the second term similarly. For $j > i$, by applying the chain rule and using the reversed Jacobi field formulation of Lemma 5.2 for the derivative of a geodesic with respect to its end point, we obtain

$$D_b f[\eta_{i+1}^{[k-1]}] = J_{\bar{f}, \eta_{i+1}^{[k-1]}}(1-t).$$

Finally, as $D_x b[\eta] = D_b f[D_x b[\eta]] = 0$ for $x = b_i$, the assumption follows. \square

Figure 5.3 represents one level of the schematic propagation tree to compute the derivative of a Bézier curve.

5 | Optimality of the Bézier fitting curve

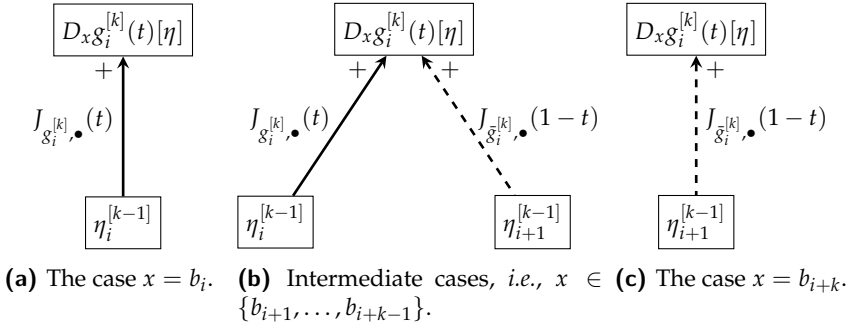


Fig. 5.3 Schematic representation of the cases where elements compose the chained derivative of the i th Bézier curve of order k in the De Castel-jau algorithm. The solid line represents a Jacobi field along $g_i^{[k]}$, while the dashed one represents a *reversed* Jacobi field.

Example 5.5 (*Quadratic Bézier curve*). Consider the quadratic Bézier curve $\beta_2: [0, 1] \rightarrow \mathcal{M}$ defined as

$$\beta_2(t; b_0, b_1, b_2) = g(t; g(t; b_0, b_1), g(t; b_1, b_2)).$$

Using the notations (5.9), we have

$$\begin{aligned} g_0^{[1]}(t) &:= g(t; b_0, b_1), & g_1^{[1]}(t) &:= g(t; b_1, b_2), \\ g_0^{[2]}(t) &:= g(t; g_0^{[1]}, g_1^{[1]}). \end{aligned}$$

The derivative of β_2 at t with respect to b_0 in the direction $\eta \in T_{b_0}\mathcal{M}$, is given by

$$D_{b_0}\beta_2[\eta] = J_{g_0^{[2]}, \eta_0^{[1]}}(t), \text{ with } \eta_0^{[1]} := J_{g_0^{[1]}, \eta}(t).$$

The derivative of β_2 at t with respect to b_2 in the direction $\eta \in T_{b_2}\mathcal{M}$ can be seen as deriving by the first point after inverting the Bézier curve, i.e., looking at $\bar{\beta}_2(t) = \beta_2(1-t)$. Hence we have analogously to the first term

$$D_{b_2}\beta_2[\eta] = J_{g_0^{[2]}, \eta_1^{[1]}}(1-t), \text{ with } \eta_1^{[1]} := J_{g_1^{[1]}, \eta}(1-t).$$

The case $D_{b_1}\beta_2[\eta]$, $\eta \in T_{b_1}\mathcal{M}$, involves a chain rule where b_1 appears in both $g_0^{[1]}$ (as its end point) and $g_1^{[1]}$ (as its starting point). Using the two

intermediate results (or Jacobi fields of geodesics)

$$\eta_0^{[1]} := J_{\bar{g}_0^{[1]}, \eta} (1-t) \quad \text{and} \quad \eta_1^{[1]} := J_{g_1^{[1]}, \eta} (t),$$

we obtain

$$D_{b_1} \beta_2[\eta] = J_{\bar{g}_0^{[2]}, \eta_0^{[1]}} (t) + J_{\bar{g}_0^{[2]}, \eta_1^{[1]}} (1-t).$$

Example 5.6 (*Cubic Bézier curve*). Consider a cubic Bézier curve $\beta_3: [0, 1] \rightarrow \mathcal{M}$ defined as

$$\beta_3(t; b_0, b_1, b_2, b_3) = g(t; \beta_2(t; b_0, b_1, b_2), \beta_2(t; b_1, b_2, b_3)).$$

As in Example 5.5, we use the notations (5.9) and define

$$\begin{aligned} g_j^{[1]}(t) &:= g(t; b_j, b_{j+1}), & j = 0, 1, 2, \\ g_j^{[2]}(t) &:= g(t; g_j^{[1]}, g_{j+1}^{[1]}), & j = 0, 1, \text{ and} \\ g_0^{[3]}(t) &:= g(t; g_0^{[2]}, g_1^{[2]}). \end{aligned}$$

The derivation of β_3 with respect to b_0 or b_3 follows the same structure as in Example 5.5. The case of $D_{b_1} \beta_3[\eta]$, however, requires two chain rules. The needed Jacobi fields follow the tree structure shown in Figure 5.4b: given $\eta \in T_{b_1} \mathcal{M}$, we define at the first recursion step

$$\eta_0^{[1]} := J_{\bar{g}_0^{[1]}, \eta} (1-t), \quad \eta_1^{[1]} := J_{g_1^{[1]}, \eta} (t),$$

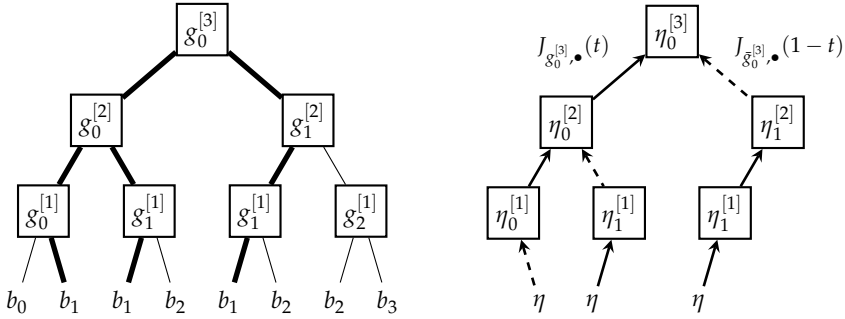
and at the second recursion step

$$\eta_0^{[2]} := J_{\bar{g}_0^{[2]}, \eta_0^{[1]}} (t) + J_{\bar{g}_0^{[2]}, \eta_1^{[1]}} (1-t), \quad \eta_1^{[2]} := J_{g_1^{[2]}, \eta_1^{[1]}} (t).$$

Note that both $\eta_0^{[2]}$ and $\eta_1^{[2]}$ are actually the derivatives of $\beta_2(t; b_0, b_1, b_2)$ and $\beta_2(t; b_1, b_2, b_3)$, respectively, with respect to b_1 and in the direction $\eta \in T_{b_1} \mathcal{M}$. Finally we have

$$D_{b_1} \beta_3[\eta] = J_{\bar{g}_0^{[3]}, \eta_0^{[2]}} (t) + J_{\bar{g}_0^{[3]}, \eta_1^{[2]}} (1-t).$$

The case of $D_{b_2} \beta_3[\eta]$ is obtained symmetrically.



(a) Tree-representation of the construction of a cubic Bézier curve. The thick line tracks the propagation of b_1 within the tree.

(b) Tree-representation of the recursive construction of $\eta_0^{[3]} := D_{b_1} \beta_3[\eta]$. The solid lines are Jacobi fields while dashed lines are *reversed* Jacobi fields.

Fig. 5.4 Construction and derivation tree of a Bézier curve $\beta_3(t; b_0, b_1, b_2, b_3)$. The derivative with respect to a variable b_i is obtained by a recursion of Jacobi fields added at each leaf of the tree.

5.2.4 Joining segments and deriving the gradient

In this subsection we express the derivative of a composite Bézier curve $\mathbf{B}(t)$ and take the C^1 conditions (3.19) into account. We simplify the general definition and fix the degree to $K_i = K$ for all segments, *e.g.*, $K = 3$ for a cubic composite Bézier curve. Then the control points are b_j^i , $j = 0, \dots, K$, $i = 0, \dots, n - 1$. We use the notations from Chapter 3 and denote by $p_i = b_{K-1}^{i-1} = b_0^i$, $i = 1, \dots, n - 1$ the common junction point of the segments and p_0 and p_n the start and end points, respectively; we denote by $b_i^- = b_{K-1}^{i-1}$ and $b_i^+ = b_1^i$, $i = 1, \dots, n - 1$, the two points needed for differentiability condition investigation (see Figure 3.4 for an illustration of the framework on $\mathcal{M} = \mathbb{R}$, with $K = 3$).

One possibility to enforce the C^1 condition (3.19) is to include it into the composite Bézier curve by replacing b_i^+ with

$$b_i^+ = g(2; b_i^-, p_i), \quad i = 1, \dots, n - 1. \tag{5.11}$$

This way both the directional derivatives of $\mathbf{B}(t)$ with respect to b_i^- and p_i change due to a further (most inner) chain rule.

Lemma 5.7 (Derivative of a composite Bézier curve with C^1 condition). *Let \mathbf{B} be a composite Bézier curve and p_i, b_i^+, b_i^- introduced as above. Replacing $b_i^+ =$*

$g(2; b_i^-, p_i)$ eliminates that variable from the composite Bézier curve and keeps the remaining derivatives unchanged, except the following which now reads

$$D_{b_i^-} \mathbf{B}(t)[\eta] = \begin{cases} D_{b_i^-} \beta_K(t - i + 1; p_{i-1}, b_{i-1}^+, \dots, \cdot, p_i)[\eta] & t \in (i - 1, i], \\ D_{b_i^+} \beta_K(t - i; p_i, b_i^+, \dots, \cdot, p_{i+1}) [D_{b_i^-} g(2; \cdot, p_i)[\eta]] & t \in (i, i + 1], \end{cases}$$

and

$$D_{p_i} \mathbf{B}(t)[\eta] = \begin{cases} D_{p_i} \beta_K(t - i + 1; p_{i-1}, b_{i-1}^+, \dots, b_i^-, \cdot)[\eta] & t \in (i - 1, i], \\ D_{p_i} \beta_K(t - i; \cdot, b_i^+, \dots, b_{i+1}^-, p_{i+1})[\eta] \\ \quad + D_{b_i^+} \beta_K(t - i; p_i, \cdot, \dots, b_{i+1}^-, p_{i+1}) [D_{p_i} g(2; b_i^-, \cdot)[\eta]] & t \in (i, i + 1]. \end{cases}$$

In both cases, the first interval includes $i - 1 = 0$, when $i = 1$.

Proof. Both first cases are simply the derivation of a Bézier curve as before; for both second cases, replacing $b_i^+ = g(2; b_i^-, p_i)$ yields one additional geodesic derivation. \square

We now derive the gradient of the objective function (5.4). We introduce the abbreviation $\mathbf{B}_i = \mathbf{B}(t_i) \in \mathcal{M}$, and $d_{2,i}^2 = d_{\mathcal{M}}^2(\mathbf{B}_{i-1}, \mathbf{B}_i, \mathbf{B}_{i+1})$.

Theorem 5.8. Let \mathcal{M} be a m -dimensional manifold, x be one of the control points of a composite Bézier curve \mathbf{B} , and $\{\tilde{\xi}_1, \dots, \tilde{\xi}_m\}$ be a corresponding orthonormal basis (ONB) of $T_x \mathcal{M}$. The gradient $\nabla_{\mathcal{M},x} A(\mathbf{b})$ of the discretized mean squared acceleration $A(\mathbf{b})$ of \mathbf{B} , w.r.t. x , discretized at $N + 1$ equispaced times t_0, \dots, t_N is given by

$$\nabla_{\mathcal{M},x} A(\mathbf{b}) = \frac{1}{\Delta t^3} \sum_{\ell=1}^m \sum_{i=1}^{N-1} \sum_{j=i-1}^{i+1} \langle \nabla_{\mathcal{M}} d_{2,i}^2, D_x \mathbf{B}_j[\tilde{\xi}_\ell] \rangle_{\mathbf{B}_i} \tilde{\xi}_\ell.$$

Proof. As $\nabla_{\mathcal{M},x} A(\mathbf{b}) \in T_x \mathcal{M}$, we seek for the coefficients $a_\ell := a_\ell(x)$ such that

$$\nabla_{\mathcal{M},x} A(\mathbf{b}) = \sum_{\ell=1}^m a_\ell \tilde{\xi}_\ell. \quad (5.12)$$

gradient
 $\nabla_{\mathcal{M},x} A(\mathbf{b})$ of
the MSA

5 | Optimality of the Bézier fitting curve

Therefore, for any tangential vector $\eta := \sum_{\ell=1}^m \eta_\ell \xi_\ell \in T_x \mathcal{M}$, we have,

$$\langle \nabla_{\mathcal{M},x} A(\mathbf{b}), \eta \rangle_x = \sum_{\ell=1}^m a_\ell \eta_\ell. \quad (5.13)$$

By definition of $A(\mathbf{b})$ and Definition 2.36 this yields

$$\langle \nabla_{\mathcal{M},x} A(\mathbf{b}), \eta \rangle_x = \frac{1}{\Delta t^3} \sum_{i=1}^{N-1} \langle \nabla_{\mathcal{M},x} d_{2,i}^2, \eta \rangle_x = \frac{1}{\Delta t^3} \sum_{i=1}^{N-1} D_x d_{2,i}^2[\eta]. \quad (5.14)$$

We compute $D_x d_{2,i}^2[\eta]$ using the chain rule (Definition 5.1) as

$$D_x d_{2,i}^2[\eta] = \sum_{j=i-1}^{i+1} D_{\mathbf{B}_j} d_{2,i}^2[D_x \mathbf{B}_j[\eta]], \quad (5.15)$$

which, by Definition 2.36, again becomes

$$D_{\mathbf{B}_j} d_{2,i}^2[D_x \mathbf{B}_j[\eta]] = \langle \nabla_{\mathcal{M}} d_{2,i}^2, D_x \mathbf{B}_j[\eta] \rangle_{\mathbf{B}_j}.$$

The term on the left of the inner product is given in [BBSW16, Sec. 3] and the right term is given in Section 5.2.3. The former can be computed using Jacobi fields and a logarithmic map, the latter is the iteratively coupling of Jacobi fields. Furthermore, the differential $D_x \mathbf{B}_j[\eta]$ can be written as

$$D_x \mathbf{B}_j[\eta] = \sum_{\ell=1}^m \eta_\ell D_x \mathbf{B}_j[\xi_\ell] \in T_{\mathbf{B}_j} \mathcal{M}.$$

Hence, we obtain

$$D_{\mathbf{B}_j} d_{2,i}^2[D_x \mathbf{B}_j[\eta]] = \sum_{\ell=1}^m \eta_\ell \langle \nabla_{\mathcal{M}} d_{2,i}^2, D_x \mathbf{B}_j[\xi_\ell] \rangle_{\mathbf{B}_j},$$

and by (5.13), (5.14) and (5.15), it follows

$$\langle \nabla_{\mathcal{M},x} A(\mathbf{b}), \eta \rangle_x = \frac{1}{\Delta t^3} \sum_{\ell=1}^m \eta_\ell \sum_{i=1}^{N-1} \sum_{j=i-1}^{i+1} \langle \nabla_{\mathcal{M}} d_{2,i}^2, D_x \mathbf{B}_j[\xi_\ell] \rangle_{\mathbf{B}_j},$$

which yields the assertion (5.12). \square

5.3 Numerical considerations

In order to apply the theory of Section 5.2 to the fitting problem (5.1), we briefly mention some numerical considerations, like the practical implementation methods.

Minimizing $A(\mathbf{b})$ alone leads to any geodesic or to the trivial solution, for any set of control points $\mathbf{b} = (x, \dots, x)$, $x \in \mathcal{M}$, and this is why the fitting term from (5.1) is important. As the variables to optimize are the control points, the fitting problem (5.1) becomes

$$\min_{\mathbf{b} \in \Gamma_{\mathbf{B}}} E_{\lambda}(\mathbf{b}) := \min_{\mathbf{b} \in \Gamma_{\mathbf{B}}} \int_{t_0}^{t_n} \left\| \frac{D^2 \mathbf{B}(t)}{dt^2} \right\|_{\mathbf{B}(t)}^2 dt + \frac{\lambda}{2} \sum_{i=0}^n d^2(p_i, d_i), \quad (5.16)$$

where $\Gamma_{\mathbf{B}} \in \mathcal{M}^M$ is the set of the M control points of \mathbf{B} . Remark that, compared to (5.1), the optimization is now on the product manifold \mathcal{M}^M , and not on an infinite space of curves \mathbf{B} .

The section is divided in three parts: the product manifold \mathcal{M}^M is defined in Section 5.3.1, where the contribution of the fitting term in the gradient of E is also presented. Then, we propose an efficient algorithm to compute the gradient of the discretized MSA, based on so-called *adjoint Jacobi fields*. We finally shortly mention the gradient descent algorithm as well as the involved Armijo rule.

5.3.1 Fitting and interpolation

Let us clarify the set $\Gamma_{\mathbf{B}} \in \mathcal{M}^M$ from (5.16). We will explicitly present the vector \mathbf{b} and state its size M . The set $\Gamma_{\mathbf{B}}$ is the set of the M remaining free control points to optimize, when the C^1 continuity constraints are imposed. We distinguish two cases: (i) the fitting case, and (ii) the interpolation case ($\lambda \rightarrow \infty$) where the constraint $d_i = p_i$ is imposed.

For a given composite Bézier curve $\mathbf{B}: [0, n] \rightarrow \mathcal{M}$ consisting of a Bézier curve of degree K on each segment, and given the C^1 conditions (3.19), the segments are determined by the points

$$\mathbf{b} = (p_0, b_0^+, b_2^0, b_3^0, \dots, b_{K-2}^0, b_1^-, p_1, b_1^+, \dots, b_n^-, p_n) \in \mathcal{M}^M. \quad (5.17)$$

We investigate the length of M . First, b_i^+ is given by p_i and b_i^- via (3.19). Secondly, the first segment contains the additional value of b_0^+ that is not

5 | Optimality of the Bézier fitting curve

fixed by C^1 constraints. The first segment is thus composed of $K + 1$ control points, while the $n - 1$ remaining segments are determined by $K - 1$ points. In total we obtain $M = n(K - 1) + 2$ control points to optimize.

Fitting ($0 < \lambda < \infty$). In the fitting scheme, (5.16) reads

$$\operatorname{argmin}_{\mathbf{b} \in \Gamma_{\mathbf{B}}} \tilde{A}(\mathbf{b}), \quad \tilde{A}(\mathbf{b}) := A(\mathbf{b}) + \frac{\lambda}{2} \sum_{i=0}^n d_{\mathcal{M}}^2(d_i, p_i), \quad (5.18)$$

where $\lambda \in \mathbb{R}^+$ sets the priority to either the data term (large λ) or the mean squared acceleration (small λ) within the minimization. The gradient of the data term is given in (2.5.4); thus, the gradient of \tilde{A} is given by

$$\nabla_{\mathcal{M}^M, x} \tilde{A}(\mathbf{b}) = \begin{cases} \nabla_{\mathcal{M}^M, x} A(\mathbf{b}) - \lambda \log_{p_i} d_i & \text{if } x = p_i, i = 0, \dots, n, \\ \nabla_{\mathcal{M}^M, x} A(\mathbf{b}) & \text{otherwise.} \end{cases}$$

Interpolation ($\lambda \rightarrow \infty$). For interpolation we assume that the start point p_{i-1} and end point p_i of the segments $i = 1, \dots, n$ are fixed to given data d_{i-1} and $d_i \in \mathcal{M}$, respectively. The optimization of the discrete mean squared acceleration $A(\mathbf{b})$ now reads

$$\operatorname{argmin}_{\mathbf{b} \in \Gamma_{\mathbf{B}}} A(\mathbf{b}) \text{ such that } p_i = d_i, i = 0, \dots, n. \quad (5.19)$$

Since the p_i are fixed by constraint, they can be omitted from the vector \mathbf{b} . We obtain

$$\mathbf{b} = (b_0^+, b_2^0, \dots, b_1^-, b_2^1, \dots, b_n^-) \in \mathcal{M}^{M'}$$

Since there are $n + 1$ additional constraints, the minimization is hence performed on the product manifold $\mathcal{M}^{M'}$, $M' = M - (n + 1) = n(K - 2) + 1$.

5.3.2 Adjoint Jacobi fields

In the Euclidean space \mathbb{R}^m , the adjoint operator T^* of a linear bounded operator $T: \mathbb{R}^m \rightarrow \mathbb{R}^q$ is the operator fulfilling

$$\langle T(x), y \rangle_{\mathbb{R}^q} = \langle x, T^*(y) \rangle_{\mathbb{R}^m}, \quad \text{for all } x \in \mathbb{R}^m, y \in \mathbb{R}^q.$$

The same can be defined for a linear operator $S: T_x \mathcal{M} \rightarrow T_y \mathcal{M}$, $x, y \in \mathcal{M}$, on a m -dimensional Riemannian manifold \mathcal{M} . The adjoint opera-

tor $S^*: T_y\mathcal{M} \rightarrow T_x\mathcal{M}$ satisfies

$$\langle S(\eta), \nu \rangle_y = \langle \eta, S^*(\nu) \rangle_x, \quad \text{for all } \eta \in T_x\mathcal{M}, \nu \in T_y\mathcal{M}.$$

In the case where S is the differential operator D_x of a geodesic $F(x) = g(t; x, y)$, the differential $D_x F: T_x\mathcal{M} \rightarrow T_{F(x)}\mathcal{M}$ can be written as

$$D_x F[\eta] = J_{g,\eta}(t) = \sum_{\ell=1}^m \langle \eta, \xi_\ell \rangle_x \alpha_\ell \Xi_\ell(t),$$

where α_ℓ are the coefficients of the Jacobi field (5.7), and $\xi_\ell, \Xi_\ell(t)$ are given as in Lemma 5.2. We observe that for any $\nu \in T_{F(x)}\mathcal{M}$ we have

$$\langle D_x F[\eta], \nu \rangle_{F(x)} = \sum_{\ell=1}^m \langle \eta, \xi_\ell \rangle_x \alpha_\ell \langle \Xi_\ell(t), \nu \rangle_{F(x)} = \left\langle \eta, \sum_{\ell=1}^m \langle \Xi_\ell(t), \nu \rangle_{F(x)} \alpha_\ell \xi_\ell \right\rangle_x.$$

Hence the *adjoint differential* $(D_x F)^*: T_{F(x)}\mathcal{M} \rightarrow T_x\mathcal{M}$ is given by

$$(D_x F)^*[v] = \sum_{\ell=1}^m \langle v, \Xi_\ell(t) \rangle_{F(x)} \alpha_\ell \xi_\ell, \quad v \in T_{F(x)}\mathcal{M}.$$

We introduce the *adjoint Jacobi field* $J_{F,\nu}^*: \mathbb{R} \rightarrow T_x\mathcal{M}$, $\nu \in T_{F(x)}\mathcal{M}$ as

$$J_{F,\nu}^*(t) = \sum_{\ell=1}^m \langle v, \Xi_\ell(t) \rangle_{F(x)} J_{F,\Xi_\ell(t)}^*(t) = \sum_{\ell=1}^m \langle v, \Xi_\ell(t) \rangle_{F(x)} \alpha_\ell \xi_\ell.$$

adjoint Jacobi
field

Note that evaluating the adjoint Jacobi field J^* involves the same transported frame $\{\Xi_1(t), \dots, \Xi_m(t)\}$ and the same coefficients α_ℓ as the Jacobi field J , which means that the evaluation of the adjoint is in no way inferior to the Jacobi field itself.

The adjoint D^* of the differential is useful in particular, when computing the gradient $\nabla_{\mathcal{M}}(h \circ F)$ of the composition of $F: \mathcal{M} \rightarrow \mathcal{M}$ with $h: \mathcal{M} \rightarrow \mathbb{R}$. Setting $y := F(x) \in \mathcal{M}$, we obtain for any $\eta \in T_x\mathcal{M}$ that

$$\begin{aligned} \langle \nabla_{\mathcal{M},x}(h \circ F)(x), \eta \rangle_x &= D_x(h \circ F)[\eta] \\ &= D_{F(x)}h[D_x F[\eta]] \\ &= \langle \nabla_{\mathcal{M},y}h(y), D_x F[\eta] \rangle_y \\ &= \left\langle (D_x F)^*[\nabla_{\mathcal{M},y}h(y)], \eta \right\rangle_x. \end{aligned}$$

5 | Optimality of the Bézier fitting curve

Especially for the evaluation of the gradient of the composite function $h \circ F$ we obtain

$$\nabla_{\mathcal{M},x}(h \circ F)(x) = (D_x F)^* [\nabla_{\mathcal{M},y} h(y)] = J_{F, \nabla_{\mathcal{M},y} h(y)}^*(t).$$

The main advantage of this technique appears in the case of composite functions, *i.e.*, of the form $h \circ F_1 \circ F_2$ (the generalization to composition with K functions is similar). The gradient $\nabla_{\mathcal{M},x}(h \circ F_1 \circ F_2)(x)$ now reads,

$$\nabla_{\mathcal{M},x}(h \circ F_1 \circ F_2)(x) = (D_x F_2)^* [\nabla_{\mathcal{M},y_2} h \circ F_1(y_2)] = J_{F_2, \nabla_{\mathcal{M},y_2} h \circ F_1(y_2)}^*(t).$$

The recursive computation of $\eta^{[3]} = \nabla_{\mathcal{M},x} h(x)$ is then given “from the top of the three” and not from the bottom, *i.e.*,

$$\begin{aligned} \eta^{[1]} &= \nabla_{\mathcal{M},y_1} h(y_1), \\ \eta^{[2]} &= J_{F_1, \eta^{[1]}}^*(t), \\ \eta^{[3]} &= J_{F_2, \eta^{[2]}}^*(t). \end{aligned}$$

Example 5.9. For $h = d_2^2: \mathcal{M}^3 \rightarrow \mathbb{R}$ we know $\nabla_{\mathcal{M}^3} h$ by Lemma 3.1 and Lemma 3.2 from [BBSW16]. Let $t_1, t_2, t_3 \in [0, 1]$ be time points, and $\mathbf{b} \in \mathcal{M}^M$ be a given (sub)set of the control points of a (composite) Bézier curve \mathbf{B} . We define $F: \mathcal{M}^M \rightarrow \mathcal{M}^3$, $\mathbf{b} \mapsto F(\mathbf{b}) = (\mathbf{B}(t_1), \mathbf{B}(t_2), \mathbf{B}(t_3))$ as the evaluations of \mathbf{B} at the three given time points. The composition $h \circ F$ hence consists of (in order of evaluation) the geodesic evaluations of the De Casteljau algorithm, the mid point function for the first and third time point and a distance function.

The recursive evaluation of the gradient starts with the gradient of the distance function. Then, for the first and third arguments, a mid point Jacobi field is applied. The result is plugged into the last geodesic evaluated within the De Casteljau “tree” of geodesics. At each geodesic, a tangent vector at a point $g(t_j; a, b)$ is the input for two adjoint Jacobi fields, one mapping to $T_a \mathcal{M}$, the other to $T_b \mathcal{M}$. This information is available throughout the recursion steps anyways. After traversing this tree backwards, one obtains the required gradient of $h \circ F$.

Note also that even the differentiability constraint (3.19) yields only two further (most outer) adjoint Jacobi fields, namely $J_{g(2, b_i^-, p_i), \nabla_{\mathcal{M}, b_i^+} \mathbf{B}(t_j)}^*(2)$ and $J_{\tilde{g}(2, b_i^-, p_i), \nabla_{\mathcal{M}, b_i^+} \mathbf{B}(t_j)}^*(2)$. They correspond to variation of the start point

Algorithm 5 Gradient descent algorithm on a manifold $\mathcal{N} = \mathcal{M}^M$

Input. $F: \mathcal{N} \rightarrow \mathbb{R}$, its gradient $\nabla_{\mathcal{N}}F$, $x^{(0)} \in \mathcal{N}$, step sizes $s_k > 0$, $k \in \mathbb{N}$.
Output: $\hat{x} \in \mathcal{N}$
 $k \leftarrow 0$
repeat
 Perform a gradient descent step $x^{(k+1)} := \exp_{x^{(k)}}(-s_k \nabla_{\mathcal{N}}F(x^{(k)}))$
 $k \leftarrow k + 1$
until a stopping criterion is reached
return $\hat{x} := x^{(k)}$

b_i^- and the end point p_i , respectively as stated in (5.7).

5.3.3 A gradient descent algorithm

To address (5.18) or (5.19), we use a gradient descent algorithm, as described in [AMS08, Ch. 4]. For completeness, the algorithm is given in Algorithm 5. The step sizes are given by the Armijo line search condition presented in [AMS08, Def. 4.2.2]. Let \mathcal{N} be a Riemannian manifold, $x = x^{(k)} \in \mathcal{N}$ be an iterate of the gradient descent, and $\beta, \sigma \in (0, 1)$, $\alpha > 0$. Let m be the smallest positive integer such that

$$F(x) - F(\exp_x(-\beta^m \alpha \nabla_{\mathcal{N}}F(x))) \geq \sigma \beta^m \alpha \|\nabla_{\mathcal{N}}F(x)\|_x. \quad (5.20)$$

We set the step size to $s_k := \beta^m \alpha$ in Algorithm 5.

As a stopping criterion we use a maximal number k_{\max} of iterations or a minimal change per iteration $d_{\mathcal{N}}(x^k, x^{k+1}) < \varepsilon$. In practice, this last criterion is matched first.

The gradient descent algorithm converges to a critical point if the function F is convex [AMS08, Sec. 4.3 and 4.4]. The mid-points model (5.3) possesses two advantages: (i) the complete discretized MSA (5.4) consists of (chained) evaluations of geodesics and a distance function, and (ii) it reduces to the classical second order differences on the Euclidean space. However, this model is not convex on general manifolds. An example is given in the arXiv preprint (version 3) of [BBSW16], Remark 4.6. Another possibility (also reducing to classical second order differences in Euclidean space) is the so-called Log model (see, e.g., [Bou13])

$$d_{2,\text{Log}}[x, y, z] = \|\log_y x + \log_y z\|_y.$$

The (merely technical) disadvantage of the Log model is that the computation of the gradient involves further Jacobi fields than the one presented above, namely to compute the differentials of the logarithmic map both with respect to its argument $D_x \log_y x$ as well as its base point $D_y \log_y x$. Still, these can be given in closed form for symmetric Riemannian manifolds [BFPS18, Th. 7.2][Per18, Lem.2.3]. To the best of our knowledge, the joint convexity of the Log model in x, y and z is still an open question.

5.4 Validation of the fitting methods

In this section, we provide several examples of the gradient descent algorithm applied to the fitting problem (5.16). We validate it first on the Euclidean space and verify that it retrieves the natural cubic smoothing spline. We then present examples on the sphere S^2 and the special orthogonal group $SO(3)$. We compare the results with the Bézier fitting algorithm from Section 4.1.

We will show in this section that the gradient descent approach performs as good as the fitting methods from Chapter 4 when the data lies on the Euclidean space (Section 5.4.1). This also means that all methods work equally well whenever the data points on the manifold are local enough. However, we will also show by the other examples (and this is not surprising) that the “fast” fitting methods fail to approach the optimal solution whenever the data points are spread out on the manifold. Those examples are given in Sections 5.4.2 and 5.4.3.

All examples were implemented in MVIRT [Ber17], and the comparison implementations from Chapter 4 use Manopt [BMAS14].

5.4.1 Validation on the Euclidean space

Let $\mathcal{M} = \mathbb{R}^3$. To verify that we recover the natural smoothing spline, we compute the comparison curve with the method from Section 4.1 (Bézier fitting) that we call here “fast Bézier”. We use the following data points

$$d_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad d_1 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \quad d_2 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \quad d_3 = \frac{1}{\sqrt{82}} \begin{bmatrix} 0 \\ -1 \\ -9 \end{bmatrix}, \quad (5.21)$$

and we initialize the control points as $p_i = d_i$ and

$$\begin{aligned} b_0^+ &= \exp_{p_0} \frac{\pi}{8\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, & b_1^+ &= \exp_{p_1} -\frac{\pi}{2\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, & b_1^- &= g(2; b_1^+, p_1), \\ b_2^+ &= \exp_{p_2} \frac{\pi}{2\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, & b_3^- &= \exp_{p_3} \frac{\pi}{8} \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, & b_2^- &= g(2; b_2^+, p_2), \end{aligned} \quad (5.22)$$

for the gradient descent algorithm. The exponential map and the geodesic on \mathbb{R}^3 are actually the addition and line segments, respectively. Note that, by construction, the initial curve is continuously differentiable but (obviously) does not minimize (5.1). The parameter λ is set to 50. The MSA of this initial curve $\tilde{A}(\mathbf{b})$ is approximately 18.8828. The result is shown in Figure 5.5a together with the first order differences along the curve in Figure 5.5b.

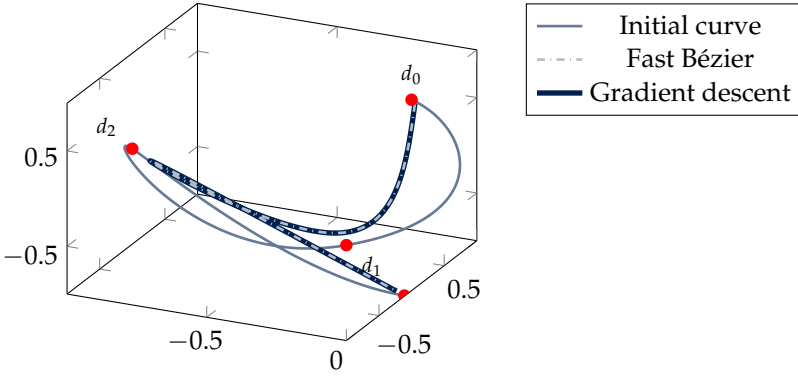
The set of control points $(p_0, b_0^+, \dots, b_3^-, p_3)$ is optimized with the gradient descent. We discretize the second order difference using $N = 1600$ points. The resulting curve and the first order difference plots are also obtained using these sampling values and a first order forward difference. The parameters of the Armijo rule (5.20) are set to $\beta = \frac{1}{2}$, $\sigma = 10^{-4}$, and $\alpha = 1$. The stopping criteria are $\sum_{i=1}^{1600} d_{\mathcal{M}}(x_i^{(k)}, x_i^{(k+1)}) < \epsilon = 10^{-15}$ or $\|\nabla_{\mathcal{M}^n} \tilde{A}\|_2 < 10^{-9}$, and the algorithm stops when one of the two is met. For this example, the first criterion stopped the algorithm, while the norm of the gradient was of magnitude 10^{-6} .

Both methods improve the initial functional value of $\tilde{A}(\mathbf{b}) \approx 18.8828$ to a value of $\tilde{A}(\mathbf{b}_{\min}) \approx 4.981218$. Both the fast Bézier approach and the gradient descent perform equally. The difference of objective value is 2.4524×10^{-11} smaller for the gradient descent, and the maximal distance between the sampling points of the resulting curves is of size 4.3×10^{-7} . Hence, in the Euclidean space, the proposed gradient descent yields the natural cubic spline, as one would expect.

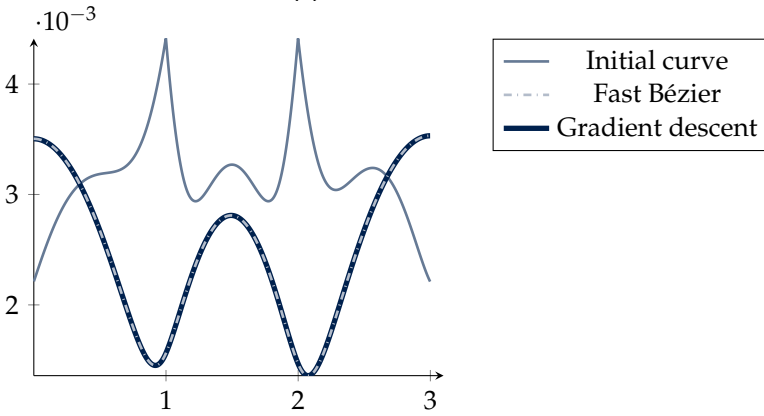
5.4.2 Examples on the sphere \mathbb{S}^2

Validation on a geodesic. As a second example with a known minimizer, we consider the manifold $\mathcal{M} = \mathbb{S}^2$, *i.e.*, the two-dimensional unit sphere

5 | Optimality of the Bézier fitting curve



(a) Bézier curves.



(b) First order differences.

Fig. 5.5 The initial interpolating Bézier curve in \mathbb{R}^3 (solid blue, a) with an MSA of 18.8828 is optimized by the Bézier fitting curve (fast Bézier) from Section 4.1 (dashed) and by the proposed gradient descent (solid fat darkblue). As expected, both curve coincide, with an MSA of 4.981218. The correspondence is further illustrated with their first order differences in (b).

embedded in \mathbb{R}^3 , where geodesics are great arcs. We use the data points

$$d_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad d_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad d_2 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

aligned on the geodesic connecting the north pole p_0 and the south pole p_2 , and running through a point p_1 on the equator. We define the control points of the cubic Bézier curve as follows:

$$x_0 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}, \quad b_0^+ = \exp_{p_0} \left(3 \log_{p_0} (x_0) \right), \quad b_1^- = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix},$$

$$x_2 = \frac{1}{\sqrt{6}} \begin{bmatrix} -1 \\ 1 \\ -2 \end{bmatrix}, \quad b_1^+ = \frac{1}{\sqrt{6}} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}, \quad b_2^- = \exp_{p_2} \left(\frac{1}{3} \log_{p_0} (x_2) \right),$$

where x_0 and x_2 are temporary points and $p_i = d_i$. We obtain two segments smoothly connected since $\log_{p_1} b_1^- = -\log_{p_1} b_1^+$. The original curve is shown in Figure 5.6a, where especially the tangent vectors illustrate the different speeds at p_i .

The control points are optimized with the interpolating model, *i.e.*, we fix the start and end points p_0, p_1, p_2 and minimize $A(\mathbf{b})$.

The curve, as well as the second and first order differences, is sampled with $N = 201$ equispaced points. The parameters of the Armijo rule are again set to $\beta = \frac{1}{2}$, $\sigma = 10^{-4}$, and $\alpha = 1$. The stopping criteria are slightly relaxed to 10^{-7} for the distance and 10^{-5} for the gradient, because of the sines and cosines involved in the exponential map.

The result is shown in Figure 5.6b. Since the minimizer is the geodesic running from p_0 to p_2 through p_1 , we measure the performance first by looking at the resulting first order difference, which is constant, as can be seen in Figure 5.6c. As a second validation, we observe that the maximal distance of the resulting curve to the geodesic is of 2.2357×10^{-6} . These evaluations again validate the quality of the gradient descent.

Effect of the data term. As a third example we investigate the effect of λ in the fitting model. We consider the data points

$$d_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad d_1 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \quad d_2 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \quad d_3 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix},$$

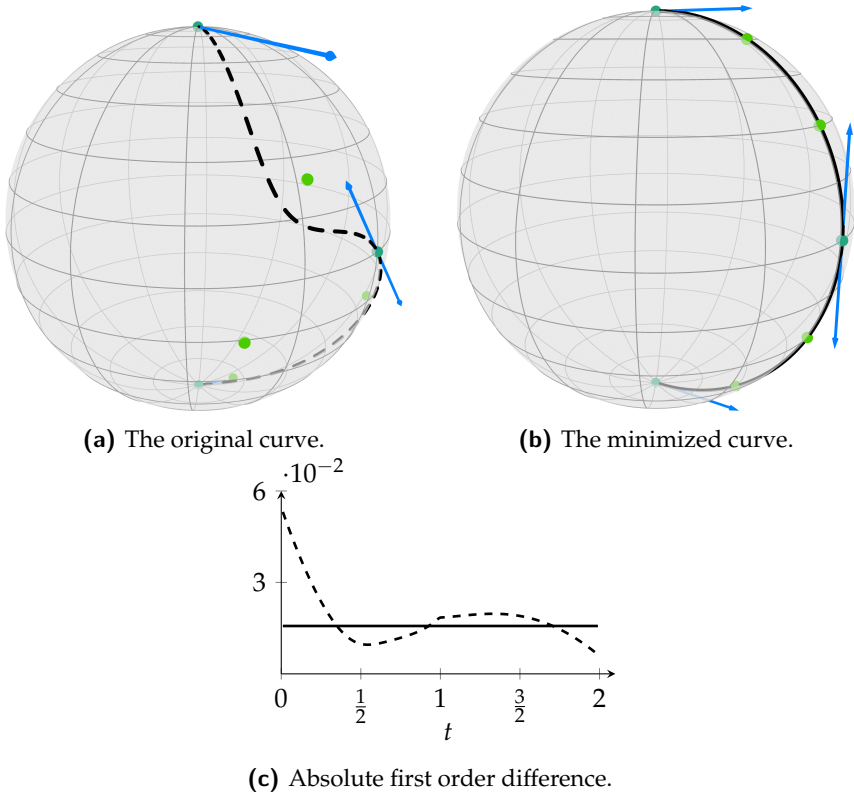


Fig. 5.6 The initial curve (dashed, a) results in a geodesic (solid, b) when minimizing the discrete acceleration. This can also be seen on the first order differences (c).

as well as the control points $p_i = d_i$, and

$$b_0^+ = \exp_{p_0} \left(\frac{\pi}{8\sqrt{2}} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \right), \quad b_1^+ = \exp_{p_1} \left(-\frac{\pi}{4\sqrt{2}} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \right),$$

$$b_2^+ = \exp_{p_2} \left(\frac{\pi}{4\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \right), \quad b_3^- = \exp_{p_3} \left(-\frac{\pi}{8\sqrt{2}} \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \right).$$

The remaining control points b_1^- and b_2^- are given by the C^1 conditions (3.19). The corresponding curve $\mathbf{B}(t)$, $t \in [0, 3]$, is shown in Figure 5.7 in dashed

black. When computing a minimal MSA curve that interpolates $\mathbf{B}(t)$ at the points p_i , $i = 0, \dots, 3$, the acceleration is not that much reduced, see the solid blue curve in Figure 5.7a.

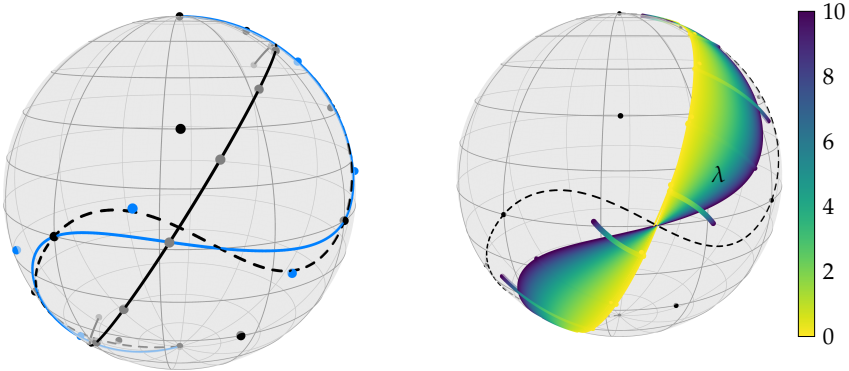
For fitting, we consider different values of λ and the same parameters as for the last example. The optimized curve fits the data points closer and closer as λ grows, and the limit $\lambda \rightarrow \infty$ yields the interpolation case. On the other hand smaller values of λ yield less fitting, but also a smaller value of the mean squared acceleration. In the limit case, *i.e.*, $\lambda \rightarrow 0$, the curve (more precisely the control points of the Bézier curve) just follows the gradient flow to a geodesic.

The results are collected in Figure 5.7. In Figure 5.7a, the original curve (dashed), is shown together with the solution of the interpolating model (solid blue) and the gradient flow result, *i.e.*, the solution of the fitting model (solid black) with $\lambda \rightarrow 0$. Figure 5.7b illustrates the effect of λ even further with a continuous variation of λ from a large value of $\lambda = 10$ to a small value of $\lambda = 0.01$. The image is generated by sampling this range with 1000 equidistant values colored in the colormap `viridis`. Furthermore, the control points are also shown in the same color.

Several corresponding functional values, see Table 5.1, further illustrate that with smaller values of λ the discretized MSA also reduces more and more to a geodesic. For $\lambda = 0$ there is no coupling to the data points and hence the algorithm does not restrict the position of the geodesic. In other words, any choice for the control points that yields a geodesic, is a solution. Note that the gradient flow still chooses a reasonably near geodesic to the initial data (Figure 5.7a, solid black).

λ	$\tilde{A}(\mathbf{b})$
orig	10.6122
∞	4.1339
10	1.6592
1	0.0733
0.1	0.0010
0.01	1.0814×10^{-5}
0.001	1.6240×10^{-7}
0	3.5988×10^{-9}

Table 5.1 MSA values of Figure 5.7 for different values of λ .



(a) Initial (dashed), interpolation (solid blue) and fitting with $\lambda = 0$ (solid black). (b) Reducing the data term from $\lambda = 10$ (violet) down to $\lambda = 0.01$ (yellow) in 1000 equidistant steps.

Fig. 5.7 The composite Bézier curves are composed of three segments. The initial curve (dashed, a) is optimized with the interpolating model (solid blue, a) as well as with the fitting model for a continuum of values of λ , from $\lambda = 10$ (violet, b) to $\lambda = 0.01$ (yellow, b). The limit case where $\lambda \rightarrow 0$ approaches an unconstrained geodesic (solid black, a).

Comparison with the tangential solution. In the Euclidean space, the method introduced Section 4.1 and the gradient descent proposed here yield the same curve, *i.e.*, the natural cubic spline. On Riemannian manifolds, however, all approaches approximate the optimal solution. Indeed, the method from Section 4.1 provides a solution by working in different tangent spaces, while the method presented here minimizes a discretization of the objective function.

In this example we take the same data points d_i as in (5.21) now interpreted as points on $\mathcal{M} = \mathbb{S}^2$, and $\lambda = 10$. Note that the control points constructed in (5.22) still fit to the sphere since each b_i^\pm is built with a vector in $T_{p_i}\mathcal{M}$. The result is shown in Figure 5.8a. The norm of the first order differences is given in Figure 5.8b. The initial curve (dashed) has an objective value of 10.9103. The local Bézier curve (thick light blue), computed with $d_{\text{ref}} = p_0$ (see page 107 for a reminder of the model), reduces this value to 7.3293. The Bézier curve (solid darkblue, Definition 4.1 lowers the MSA to 3.1863 and the gradient descent (solid orange) yields a value of 2.7908, regardless of whether the starting curve is the initial one, or one of the already computed one. Note that, when $p_3 = [0, 0, -1]^\top$, the lo-

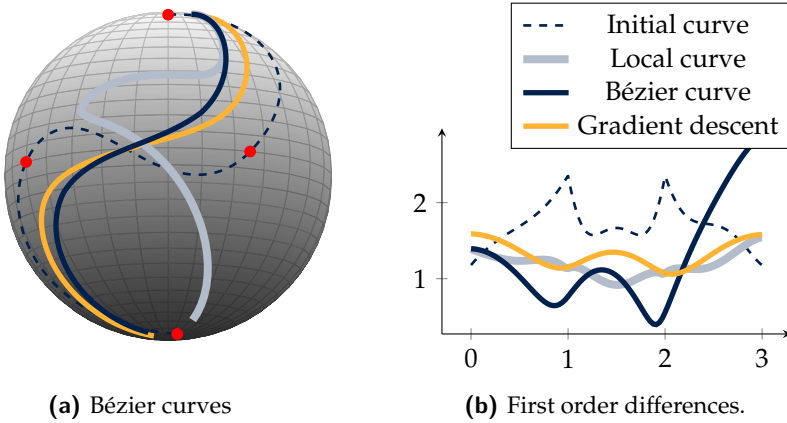


Fig. 5.8 The initial composite Bézier curve on S^2 (dashed, a) has an MSA of 10.9103. The curve obtained with a local curve (thick light blue, $d_{\text{ref}} = p_0$), has an MSA of 7.3293. The Bézier (solid dark blue) curve from Definition 4.1 reaches an MSA of 3.1863. The gradient descent (solid orange) lowers the MSA to 2.7908. The first order derivative in (b) shows that the gradient descent algorithm leads to solutions where the velocity varies less.

cal approach is not able to compute any result, since the construction is performed in the tangent space of p_0 . Thus, $\log_{p_0} p_3$ is not defined. We can observe that, despite the spreading of the data points, the method from Section 4.1 performs quite well. It is difficult to compare the quality of each curve only based on the value of the MSA, as the gradient descent technique minimizes this approximation while the Bézier fitting curve solves problem (1.1) on the tangent spaces. Logically, the gradient descent leads to a better result, but takes more time; on the other hand the improvement observed compared with the local Bézier surface shows that the methods from Chapter 4 permit to obtain an acceptable solution in short time.

5.4.3 An example of orientations

Finally we analyze the blended splines for orientations, *i.e.*, data

SO(3)

$$R_{xy}(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

5 | Optimality of the Bézier fitting curve

$$R_{xz}(\alpha) = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix},$$

$$R_{yz}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix}$$

denote the rotation matrices in the $x - y$, $x - z$, and $y - z$ planes, respectively. We introduce the three data points

$$d_0 = R_{xy} \left(\frac{4\pi}{9} \right) R_{yz} \left(-\frac{\pi}{2} \right),$$

$$d_1 = R_{xz} \left(-\frac{\pi}{8} \right) R_{xy} \left(\frac{\pi}{18} \right) R_{yz} \left(-\frac{\pi}{2} \right),$$

$$d_2 = R_{xy} \left(\frac{5\pi}{9} \right) R_{yz} \left(-\frac{\pi}{2} \right).$$

These data points are shown in the first line of Figure 5.9 in cyan.

We set $\lambda = 10$ and discretize (5.16) with $N = 401$ equispaced points. This sampling is also used to generate the first order finite differences. The parameters of the gradient descent algorithm are set to the same values as for the examples on the sphere.

We perform the blended spline fitting with two segments and cubic splines. The resulting control points are shown in the second row of Figure 5.9, in green. The objective value is $\tilde{A}(\mathbf{b}) = 0.6464$. We improve this solution with the minimization problem (5.18) on the product manifold $\text{SO}(3)^6$. We obtain the control points shown in the last line of Figure 5.9 and an objective value of $\tilde{A}(\hat{\mathbf{b}}) = 0.2909$ for the resulting minimizer $\hat{\mathbf{b}}$.

We further compare both curves by looking at their absolute first order differences. In the third line of Figure 5.9, we display 17 orientations along the initial curve with its first order difference as height. The line without samples is the absolute first order differences of the minimizer $\hat{\mathbf{b}}$, scaled to the same magnitude. The line is straightened especially for the first Bézier segment. Nevertheless, the line is still bent a little bit and hence not a geodesic. This can be seen in the fourth line which represents 17 samples of the minimizing curve compared to its control points in the last line, which are drawn on a straight line.

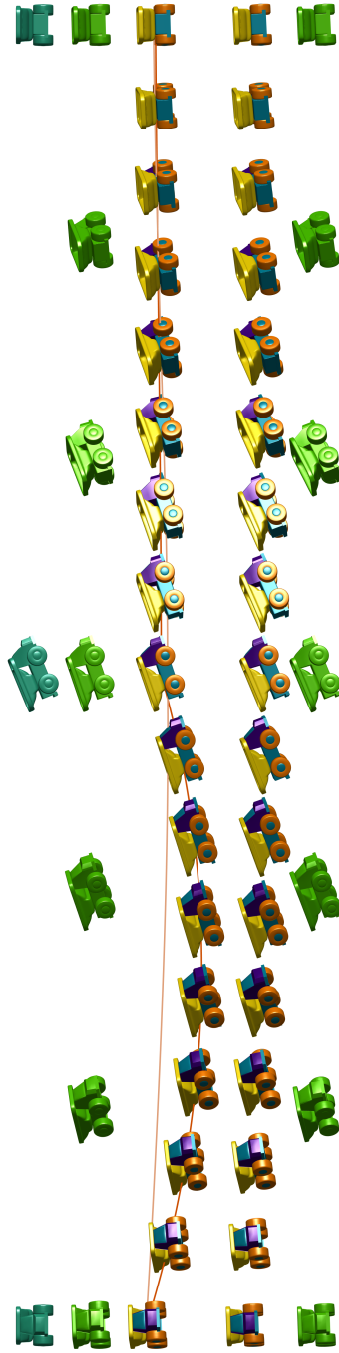


Fig. 5.9 From top row to bottom row: (1) the initial data points (cyan), (2) the control points computed by the blended curve, (3) 17 points on the corresponding curve, where the height represents the absolute first order difference, and the comparing curve is the first order differences from the curve obtained via the gradient descent, (4) 17 points along the resulting curve from gradient descent, and (5) the resulting control points of the curve in (4).

This section is based on the paper[BG18], often cited verbatim. Its reference is

[BG18] Ronny Bergmann and Pierre-Yves Gousenbourger. A variational model for data fitting on manifolds by minimizing the acceleration of a Bézier curve. *Frontiers in Applied Mathematics and Statistics*, 4(59):1–16, 2018. doi:10.3389/fams.2018.00059

6

Interpolation with Bézier surfaces

AFTER PRESENTING TECHNIQUES for univariate interpolation and fitting on manifolds, the present chapter aims at extending the scope to multivariate interpolation, *i.e.*, a framework where the data point now depend on multiple parameters.

The technique to perform multivariate manifold-valued interpolation is based on C^1 piecewise-cubic Bézier functions (see Figure 1.1 for an example). More precisely, given data points d_{i_1, \dots, i_D} on a manifold \mathcal{M} associated to nodes $(i_1, \dots, i_D) \in \mathbb{Z}^D$ of a Cartesian grid in \mathbb{R}^D , one seeks a C^1 function

$$\mathbf{S} : \mathbb{R}^D \rightarrow \mathcal{M} \text{ such that } \mathbf{S}(i_1, \dots, i_D) = d_{i_1, \dots, i_D}.$$

The technique can be viewed as a multivariate extension of the method presented in Section 3.4. For simplicity of the exposition, the focus is limited to 2D interpolation, but the transition to higher dimensions of the domain seems to be less intricate than the transition from 1D to 2D. This is the fourth contribution of this thesis.

The development of the bivariate manifold-valued technique requires more work than one might anticipate. The first task is to propose a bivariate extension of the manifold-valued Bézier curves (Definition 3.12). Three approaches are proposed in Section 6.2, leading to different results.

surface

Next comes the question of gluing these Bézier patches together in a smooth (C^1) way (Section 6.3). The problem is substantially more difficult than in Chapter 3 because the interfaces are no longer isolated points but instead regions of dimension 1. The C^0 conditions are not the core of the problem because they are the same as in Euclidean spaces; however, the classical conditions on the control points for C^1 -continuity in Euclidean spaces exhibit a linear dependence which is incompatible with general manifolds. In addition, those conditions are not sufficient to ensure C^1 -continuity on a manifold. This difficulty is resolved in Theorem 6.18 for two of the three Bézier surface definitions.

After generalizing composite curves to composite surfaces, Section 6.4 provides a technique to set control points in such a way that the resulting interpolating surface has minimal mean squared second derivative when \mathcal{M} is flat. Results are shown in Section 6.6, in the context of motion modeling and shape exploration in shape spaces.

This chapter is a collaboration with Prof. Benedikt Wirth and Dr. Paul Striowski (Universität Münster, Germany).

6.1 Euclidean Bézier surfaces

The idea of Bézier curves presented in Sections 3.1 and 3.2 extends also to higher dimensions; see, e.g., [Far02, Sec. 5.5]. In this chapter, the focus concerns Bézier surfaces, i.e., Bézier functions with a domain in \mathbb{R}^2 .

Definition 6.1 (*Euclidean Bézier surface*). For a family of points $(b_{ij})_{i,j=0,\dots,K} \subset \mathbb{R}^r$, the *Bézier surface* $\sigma_K : [0, 1]^2 \rightarrow \mathbb{R}^r$ of degree K is defined as

$$\sigma_K(t_1, t_2; (b_{ij})_{i,j=0,\dots,K}) = \sum_{i=0}^K \sum_{j=0}^K b_{ij} B_{iK}(t_1) B_{jK}(t_2), \quad (6.1)$$

where K is the order of the surface ($K = 3$ for cubic surfaces treated in this chapter), and $B_{jK}(t)$ are the same Bernstein polynomials from Definition 3.1.

Similarly to curves, for fixed t_1 and t_2 , the surface can be interpreted as a convex combination of all control points b_{ij} (Figure 6.1a). We directly see that the Bézier surface boundary consists of the four Bézier curves with control points $b_{0,j}, b_{K,j}, b_{j,0}, b_{j,K}, j = 0, \dots, K$, respectively.

We can also interpret a Bézier surface as a one-parameter family of Bézier curves, which allows an evaluation just based on the computation of Bézier curves (Figure 6.1b), in one direction at a time.

Definition 6.2 (*Tensorized Euclidean Bézier surface*). The Bézier surface can be interpreted “as a Bézier curve of a family of Bézier curves”, *i.e.*,

$$\begin{aligned}\sigma_K(t_1, t_2; (b_{ij})_{i,j=0,\dots,K}) &= \sum_{j=1}^K \left(\sum_{i=1}^K b_{ij} B_{iK}(t_1) \right) B_{jK}(t_2) \\ &= \beta_K(t_2; (\beta_K(t_1; (b_{ij})_{i=0,\dots,K}))_{j=0,\dots,K}),\end{aligned}\quad (6.2)$$

or alternatively by switching t_1 and t_2 and the corresponding control points.

A third alternative to compute the Bézier surface is to rely on the De Casteljau algorithm. In suggestive matrix notation, it takes the following form.

Definition 6.3 (*De Casteljau algorithm for surfaces*). The Bézier surface can be computed recursively as

$$\begin{aligned}x_{ij}^{[0]} &= b_{ij}, && \text{for } i, j = 0, \dots, K, \\ x_{ij}^{[k]} &= \begin{bmatrix} 1 - t_1 & t_1 \end{bmatrix} \begin{bmatrix} x_{ij}^{[k-1]} & x_{i,j+1}^{[k-1]} \\ x_{i+1,j}^{[k-1]} & x_{i+1,j+1}^{[k-1]} \end{bmatrix} \begin{bmatrix} 1 - t_2 \\ t_2 \end{bmatrix}, && \text{for } k = 1, \dots, K, \\ &&& \text{and } i, j = 0, \dots, K - k,\end{aligned}\quad (6.3)$$

and yields $x_{00}^{[K]} = \sigma_K(t_1, t_2; (b_{ij})_{i,j=0,\dots,K})$.

Here again, points $x_{ij}^{[k]}$ are obtained as convex combinations of $x_{ij}^{[k-1]}$, $x_{ij+1}^{[k-1]}$, $x_{i+1j}^{[k-1]}$ and $x_{i+1j+1}^{[k-1]}$. This combination leads once more to a simple geometric interpretation (Figure 6.1c).

Property 6.4 (*Edges and corners of the Bézier surface*). The Bézier surface interpolates the control points at the corners, *i.e.*,

$$\begin{aligned}\sigma_K(0, 0; (b_{ij})_{i,j=0,\dots,K}) &= b_{00}, \\ \sigma_K(1, 0; (b_{ij})_{i,j=0,\dots,K}) &= b_{K0}, \\ \sigma_K(0, 1; (b_{ij})_{i,j=0,\dots,K}) &= b_{0K}, \\ \sigma_K(1, 1; (b_{ij})_{i,j=0,\dots,K}) &= b_{KK}.\end{aligned}$$

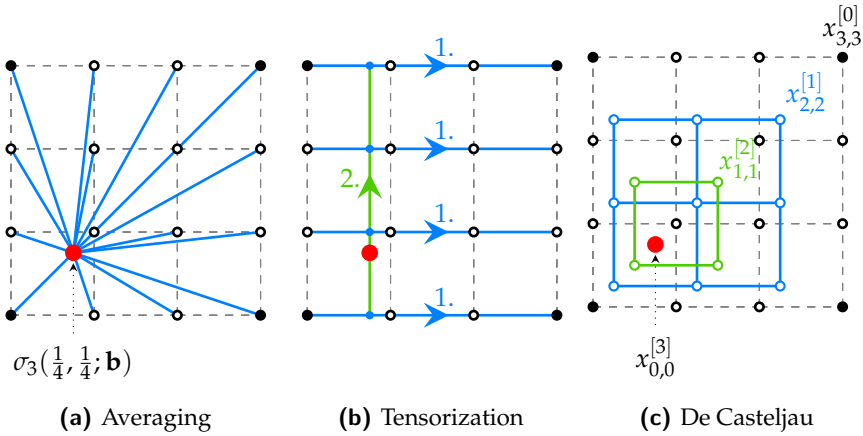


Fig. 6.1 Computation of a cubic Bézier surface via (6.1) as a weighted mean of the control points (a), via (6.2) through a one-parameter family of Bézier curves (b), and via the De Casteljau algorithm (6.3) (c). Interpolation and control points are indicated by filled and open circles, respectively; the Bézier polygon is shown as gray dashed lines.

Furthermore, the edges of the Bézier surface are Bézier curves defined as

$$\begin{aligned} \sigma_K(0, t_2; (b_{ij})_{i,j=0,\dots,K}) &= \beta_K(t_2; b_{00}, \dots, b_{0K}), \\ \sigma_K(1, t_2; (b_{ij})_{i,j=0,\dots,K}) &= \beta_K(t_2; b_{K0}, \dots, b_{KK}), \\ \sigma_K(t_1, 0; (b_{ij})_{i,j=0,\dots,K}) &= \beta_K(t_1; b_{00}, \dots, b_{K0}), \\ \sigma_K(t_1, 1; (b_{ij})_{i,j=0,\dots,K}) &= \beta_K(t_1; b_{0K}, \dots, b_{KK}). \end{aligned}$$

Proof. By construction of the tensorized version of Bézier surfaces. □

Different Bézier surfaces can be glued smoothly quite easily. For this, we introduce the notion of *composite Bézier surface*.

Definition 6.5 (*Composite Bézier surface*). Consider a set of MN Bézier surfaces σ_K^{mn} of order K , $m = 0, \dots, M - 1$, $n = 0, \dots, N - 1$, driven by the control points $(b_{ij}^{mn})_{i,j=0,\dots,K} \in \mathbb{R}^r$. The *composite Bézier surface* \mathbf{S} is defined as

$$\mathbf{S} : [0, M] \times [0, N] \rightarrow \mathbb{R}^r : (t_1, t_2) \mapsto \sigma_K^{mn}(t_1 - m, t_2 - n; (b_{ij}^{mn})_{i,j=0,\dots,K})$$

for $\lfloor t_1 \rfloor = m$, $\lfloor t_2 \rfloor = n$, defined as in Definition 3.6.

Property 6.6 (C^k -continuity [Far02, § 16.1]). Two Bézier surfaces of order K $\sigma_K(t_1, t_2; (b_{ij}^l)_{i,j=0,\dots,K})$ and $\sigma_K(t_1, t_2; (b_{ij}^r)_{i,j=0,\dots,K})$ can be joined C^k -continuously in t_1 -direction via

$$\begin{aligned} \mathbf{S} : [0, 2] \times [0, 1] &\rightarrow \mathbb{R}^r : \\ (t_1, t_2) &\mapsto \begin{cases} \sigma_K(t_1, t_2; (b_{ij}^l)_{i,j=0,\dots,K}) & \text{if } t_1 \in [0, 1], \\ \sigma_K(t_1 - 1, t_2; (b_{ij}^r)_{i,j=0,\dots,K}) & \text{if } t_1 \in (1, 2], \end{cases} \end{aligned} \quad (6.4)$$

if for all $j = 1, \dots, K$ the sequence pairs $(b_{j,0}^l, \dots, b_{j,K}^l)$, $(b_{j,0}^r, \dots, b_{j,K}^r)$ induce a one-dimensional C^k -continuous Bézier spline via (3.13). An analogous condition holds if the two surfaces are to be matched smoothly in the t_2 -direction.

Example 6.7 (Composite cubic surface). According to Property 6.6, the composite cubic Bézier surface

$$\begin{aligned} \mathbf{S} : [0, 2] \times [0, 1] &\rightarrow \mathbb{R}^r : \\ (t_1, t_2) &\mapsto \begin{cases} \sigma_3(t_1, t_2; (b_{ij}^l)_{i,j=0,\dots,3}) & \text{if } t_1 \in [0, 1], \\ \sigma_3(t_1 - 1, t_2; (b_{ij}^r)_{i,j=0,\dots,3}) & \text{if } t_1 \in (1, 2], \end{cases} \end{aligned}$$

will be C^1 continuous in the t_1 -direction if

$$b_{3j}^l = b_{0j}^r = \frac{b_{2j}^l + b_{1j}^r}{2}, \text{ for } j = 0, \dots, 3.$$

Remark 6.8. The C^1 conditions correspond to conditions of C^1 continuity of curves. Furthermore, C^k conditions are here given for two curves of same order, but the condition can be easily generalized to different orders using Property 3.9.

6.2 Bézier surfaces on manifolds

Bézier surfaces can be transferred to a manifold setting in different ways, where each approach generalizes a particular evaluation scheme for Bézier surfaces in Euclidean space. In this section, we introduce three possible definitions of Bézier surfaces on a Riemannian manifold \mathcal{M} . The three of

them lead in general to different results.

To extend Bézier surfaces to metric spaces \mathcal{M} , we mainly use the weighted average 2.53 between points $y_1, \dots, y_n \in \mathcal{M}$ as core concept. This tool can replace naturally not only convex combinations, but also multilinear interpolations.

The definitions of Bézier surfaces on manifolds are generalizations of (6.1)–(6.3). Note that equation (6.1) represents a convex combination and each iteration step in equation (6.3) represents a bilinear interpolation.

Definition 6.9 (*Generalized Bézier surface*). Given control points $b_{ij} \in \mathcal{M}$, $i, j = 0, \dots, K$, we define a corresponding *generalized Bézier surface* of type I, II and III as

$$\sigma_K^I(t_1, t_2; (b_{ij})_{i,j=0,\dots,K}) = \text{av}[(b_{ij})_{i,j=0,\dots,K}, (B_{iK}(t_1)B_{jK}(t_2))_{i,j=0,\dots,K}], \quad (6.5)$$

$$\sigma_K^{II}(t_1, t_2; (b_{ij})_{i,j=0,\dots,K}) = \beta_K(t_1; (\beta_K(t_2; (b_{ij})_{j=0,\dots,K}))_{i=0,\dots,K}), \quad (6.6)$$

$$\sigma_K^{III}(t_1, t_2; (b_{ij})_{i,j=0,\dots,K}) = x_{00}^{[K]}, \quad (6.7)$$

where $\beta_K(\cdot; (b_m)_{m=0,\dots,K})$ denotes a Bézier curve in \mathcal{M} (see Definition 3.12), and where $x_{00}^{[K]}$ is defined recursively via the generalized De Casteljau algorithm.

Definition 6.10 (*De Casteljau algorithm for surfaces*). The De Casteljau algorithm generalized on manifolds for Bézier surfaces is initialized as $x_{ij}^{[0]} = b_{ij}$, for $i, j = 0, \dots, K$, and reads

$$x_{ij}^{[k]} = \text{av} \left[\left(x_{ij}^{[k-1]}, x_{i,j+1}^{[k-1]}, x_{i+1,j}^{[k-1]}, x_{i+1,j+1}^{[k-1]} \right), (w_{00}, w_{01}, w_{10}, w_{11}) \right],$$

for $k = 1, \dots, K$, and $i, j = 0, \dots, K - k$. The weights are given by

$$\begin{aligned} w_{00} &= (1 - t_1)(1 - t_2), \\ w_{01} &= (1 - t_1)t_2, \\ w_{10} &= t_1(1 - t_2), \\ w_{11} &= t_1t_2. \end{aligned}$$

All these three types reduce to the classical Bézier surface if the manifold is the Euclidean space $\mathcal{M} = \mathbb{R}^r$. However, they will generally differ from each other on general manifolds, as illustrated in Figure 6.2.

Each approach has its advantages and disadvantages.

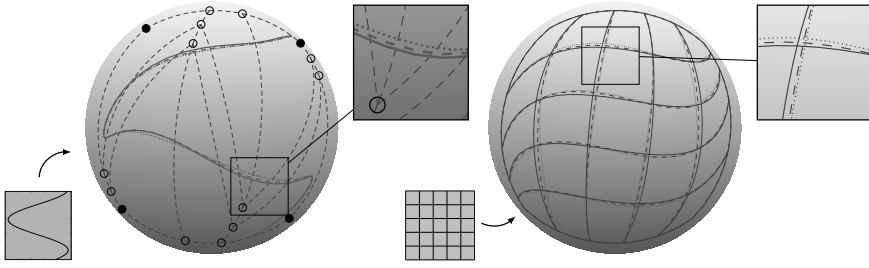


Fig. 6.2 The Bézier surfaces of type *I* (solid), *II* (dashed) and *III* (dotted) differ from each other. The differences are illustrated here on the sphere S^2 by mapping a curve (left) or a grid (right) from the parameterization domain onto the Bézier surface. Note that the Bézier surface boundaries coincide, though. The control and interpolation points of the Bézier surface are displayed in the left picture; the gray dashed lines indicate the control points grid. The curves deviate by more than 10^{-2} which lies far above our computational precision (the weighted averaging according to Definition 2.53 is performed by a gradient descent with accuracy 10^{-7}). Figure 6.3 highlights further differences between Bézier surfaces of type *II*.

The first approach (σ_K^I) entails solving a rather complex optimization problem. However, it does not suffer from the drawbacks of the other two approaches that will be precised momentarily.

In comparison to σ_K^I , the third approach (σ_K^{III}) is based on the De Casteljau algorithm: if it requires multiple steps, all of them consist in comparatively simple computation of weighted geodesic averages with only four points. Unfortunately, the main drawback of this approach is that there does not seem to be a straightforward way to patch multiple Bézier surfaces of type *III* together in a C^1 -continuous manner (this will be discussed in Section 6.3.2).

Finally, the evaluation of σ_K^{II} requires only weighted averages of two points as it is based on the one-dimensional De Casteljau algorithm. When simple analytic formulas exist for the Riemannian exponential and logarithm (e.g., for $\mathcal{M} = S^m$), this method is very advantageous since the averaging can be based on (2.6). On the other hand, unlike the surfaces of type *I* and *III*, the definition of σ_K^{II} is not symmetric: it does not satisfy the relation

$$\sigma_K^{II}(t_1, t_2; (b_{ij})_{i,j=0,\dots,K}) = \sigma_K^{II}(t_2, t_1; (b_{ij})_{i,j=0,\dots,K}^\top).$$

In the definition of σ_K^{II} , instead of computing the Bézier curve with respect

to t_2 in the first place, one could alternatively compute the Bézier curves $\beta_K(t_1; (b_{ij})_{i=0,\dots,K})$ for each j with respect to t_1 , to obtain new control points for a Bézier curve which is then evaluated at t_2 ,

$$\beta_K(t_2; (\beta_K(t_1; (b_{ij})_{i=0,\dots,K}))_{j=0,\dots,K}). \tag{6.8}$$

In general, both approaches will yield different surfaces, as shown in Figure 6.3. There is however no clear reason to prefer to start with t_1 or t_2 . The taste here will generally depend on the application, and it is not the scope of this thesis.

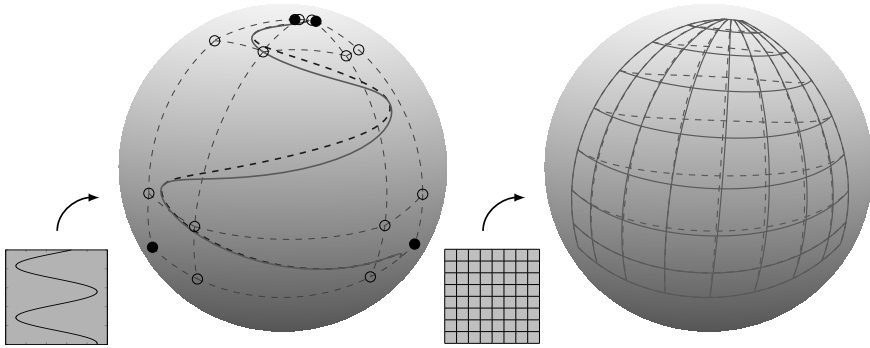


Fig. 6.3 The two possible construction methods (6.6) and (6.8) for Bézier surfaces of type II produce in general different results (their boundaries always coincide, however). Here the difference is shown on the sphere with the same conventions as in Figure 6.2: the solid line corresponds to computing the curve first in the direction t_2 (as in (6.6)), while the dashed line represents the curve computed first along t_1 (as in (6.8)).

The following theorem ensures that Bézier surfaces exist under the condition that the control points are sufficiently close to each other.

Theorem 6.11 (Existence of Bézier surfaces). *Let control points $(b_{ij})_{i,j=0,\dots,K} \in \mathcal{M}$ be given. If $\text{co}(\{b_{ij}\}_{i,j=0,\dots,K})$ is proper, then σ_K^I , σ_K^{II} , and σ_K^{III} in Definition 6.9 exist, are unique, and define smooth surfaces in $\text{co}(\{b_{ij}\}_{i,j=0,\dots,K}) \subset \mathcal{M}$.*

Proof. If the weighted geodesic averages involved in the computation of the Bézier surfaces exist and are unique, then the Bézier surfaces exist and are unique; furthermore, they will lie in the convex hull of their control points by definition. The existence of all averages follows from [AGSW16b, Theorem 9]. Their uniqueness and smoothness is direct, by definition of $\text{co}(\{b_{ij}\}_{i,j=0,\dots,K})$ (see Definition 2.55). □

6.3 Composite Bézier surfaces on manifolds

The final goal of this chapter is to construct piecewise surfaces to interpolate a set of data points. This section is dedicated to properly set the definition of composite Bézier surfaces

$$\mathbf{S} : [0, M] \times [0, N] \rightarrow \mathcal{M} : (t_1, t_2) \mapsto \sigma_K^{ij}(t_1 - i, t_2 - j; (b_{kl}^{ij})_{k,l=0,\dots,K})$$

for $[t_1] = i, [t_2] = j$, control points $b_{kl}^{ij} \in \mathcal{M}$ and σ_K computed by Definition 6.9. Unlike composite curves, constructing those C^1 piecewise surfaces is more complicated, as the conditions to ensure smoothness are not as easy to generalize to manifolds. The section is thus divided in three parts: the first one is the most easy, as it discusses how two Bézier surfaces can be glued together at zeroth order. The second part presents the challenges related to C^1 gluing, and the last part resolves the problems.

6.3.1 C^0 -patching

As in Euclidean space, two generalized Bézier surfaces can be patched together via the generalization of (6.4) to a manifold \mathcal{M} ,

$$\begin{aligned} \mathbf{S}^Y : [0, 2] \times [0, 1] &\rightarrow \mathcal{M} : \\ (t_1, t_2) &\mapsto \begin{cases} \sigma_K^Y(t_1, t_2; (b_{ij}^1)_{i,j=0,\dots,K}) & \text{if } t_1 \in [0, 1], \\ \sigma_K^Y(t_1 - 1, t_2; (b_{ij}^r)_{i,j=0,\dots,K}) & \text{if } t_1 \in (1, 2], \end{cases} \end{aligned} \quad (6.9)$$

for $Y = I, II$, or III (and analogously for patching in the t_2 -direction). The Bézier spline will be C^0 -continuous under the same conditions as in the Euclidean space.

Theorem 6.12 (C^0 -continuity). *For $Y = I, II$, and III , the patched Bézier surface \mathbf{S}^Y in \mathcal{M} is continuous if $b_{K,j}^1 = b_{0,j}^r$ for $j = 0, \dots, K$.*

Proof. The generalized Bézier surfaces (Definition 6.9) are smooth by Theorem 6.11. Hence, it remains to consider the continuity at the interface of the two domains, *i.e.*, at $t_1 = 1$. From the definition of the Bézier surfaces σ^Y , it follows immediately that only the control points $b_{K,j}^1$ and $b_{0,j}^r$ are involved in the corresponding weighted averages. Furthermore, $\sigma_K^Y(1, t_2; (b_{ij}^1)_{i,j=0,\dots,K})$ and $\sigma_K^Y(0, t_2; (b_{ij}^r)_{i,j=0,\dots,K})$ are the Bézier curves in \mathcal{M}

6 | Interpolation with Bézier surfaces

defined by the control points $(b_{K,j}^l)_{j=0,\dots,K}$ and $(b_{0,j}^r)_{j=0,\dots,K}$, respectively. Thus, if those control points coincide, the corresponding curves coincide too. \square

Exp-Log

Remark 6.13. In Riemannian spaces where the logarithm and the exponential maps are accessible and easy to compute, one might be tempted to replace the weighted geodesic averages (in the definitions of σ_K^I and σ_K^{III}) by the similar and simpler expression

$$\text{av}[(y_1, \dots, y_n), (w_1, \dots, w_n)] \approx \exp_y \left(w_1 \log_y(y_1) + \dots + w_n \log_y(y_n) \right),$$

which performs a weighted average in the tangent space at some $y \in \mathcal{M}$. For instance, in the definition of σ_K^{III} we might redefine

$$x_{ij}^{[k]} = \exp_{x_{ij}^{[k-1]}} \left(\sum_{\substack{r \in \{i, i+1\} \\ s \in \{j, j+1\}}} \log_{x_{ij}^{[k-1]}} \left(w_{rs} x_{rs}^{[k-1]} \right) \right).$$

Here, $y = x_{ij}^{[k-1]}$. In this case, we can observe two things:

- (i) the curve $\sigma_K^{III}(0, t_2; (b_{ij}^r)_{i,j=0,\dots,K})$ is a Bézier curve with control points $(b_{0,j}^r)_{j=0,\dots,K}$. The involved weighted geodesic averaging reduces to two-point averages at each step of the recursive process. By (2.6), the two-point averages computed by the exp-log approach or the classical averaging formula are identical.
- (ii) the curve $\sigma_K^{III}(1, t_2; (b_{ij}^l)_{i,j=0,\dots,K})$, however, is *not* the Bézier curve corresponding to control points $(b_{K,j}^r)_{j=0,\dots,K}$. Here, the averaging is based on the tangent space at a different point than the two of which the weighted average is computed.

As a consequence of this second observation, C^0 -continuous patching of Bézier surfaces is no longer possible, even in symmetric spaces such as the sphere. This is illustrated in Figure 6.4.

6.3.2 C^1 -patching challenges

In contrast to C^0 -continuity, which is easily achieved, C^1 -patching represents a challenge. Indeed, the corresponding conditions in the Euclidean

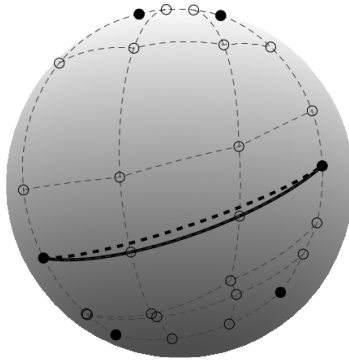


Fig. 6.4 Two cubic Bézier surfaces computed on the sphere with the De Casteljau algorithm modified as in Remark 6.13. At the interface where control points coincide, the surfaces do not match up continuously due to the simplification of weighted geodesic averaging.

space (Property 6.6) cannot be used as we will see. In the following we will always choose the outmost control points of each Bézier surface patch such that C^0 -continuity is ensured.

The Euclidean conditions for C^1 -continuity

$$b_{K,j}^l = \frac{b_{K-1,j}^l + b_{1,j}^r}{2}$$

now generalize to the Riemannian setting as

$$b_{K,j}^l = \text{av}[(b_{K-1,j}^l, b_{1,j}^r), (\frac{1}{2}, \frac{1}{2})] \tag{6.10}$$

for all j . There are multiple problems with this generalization.

Consider four Bézier surfaces that are patched together as a composite surface $\mathbf{S} : [0, 2] \times [0, 2] \rightarrow \mathcal{M}$:

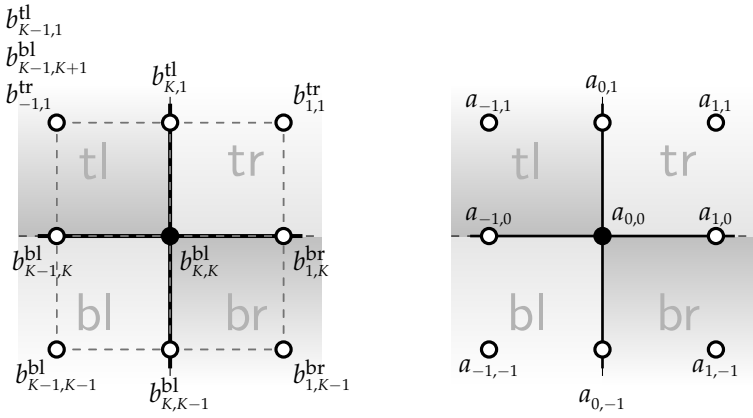
$$(t_1, t_2) \mapsto \begin{cases} \sigma_K^Y(t_1, t_2; (b_{ij}^{bl})_{i,j=0,\dots,K}) & \text{if } (t_1, t_2) \in [0, 1] \times [0, 1], \\ \sigma_K^Y(t_1 - 1, t_2; (b_{ij}^{br})_{i,j=0,\dots,K}) & \text{if } (t_1, t_2) \in [1, 2] \times [0, 1], \\ \sigma_K^Y(t_1, t_2 - 1; (b_{ij}^{tl})_{i,j=0,\dots,K}) & \text{if } (t_1, t_2) \in [0, 1] \times [1, 2], \\ \sigma_K^Y(t_1 - 1, t_2 - 1; (b_{ij}^{tr})_{i,j=0,\dots,K}) & \text{if } (t_1, t_2) \in [1, 2] \times [1, 2], \end{cases} \tag{6.11}$$

The superscripts bl, br, tl, tr stand for *bottom left*, *bottom right*, *top left*, and *top right* respectively.

Near the corner point $b_{K,K}^{bl} = b_{0,K}^{br} = b_{K,0}^{tl} = b_{0,0}^{tr}$, the six C^1 -continuity conditions (6.10) are represented in Figure 6.5a and read

$$\begin{aligned}
 b_{K,K-1}^{bl} &= \text{av}[(b_{K-1,K-1}^{bl}, b_{1,K-1}^{br}), (\frac{1}{2}, \frac{1}{2})], \\
 b_{K-1,K}^{bl} &= \text{av}[(b_{K-1,K-1}^{bl}, b_{K-1,1}^{tl}), (\frac{1}{2}, \frac{1}{2})], \\
 b_{K,K}^{bl} &= \text{av}[(b_{K-1,K}^{bl}, b_{1,K}^{br}), (\frac{1}{2}, \frac{1}{2})], \\
 b_{K,K}^{bl} &= \text{av}[(b_{K,K-1}^{bl}, b_{K,1}^{tl}), (\frac{1}{2}, \frac{1}{2})], \\
 b_{K,1}^{tl} &= \text{av}[(b_{K-1,1}^{tl}, b_{1,1}^{tr}), (\frac{1}{2}, \frac{1}{2})], \\
 b_{1,K}^{br} &= \text{av}[(b_{1,K-1}^{br}, b_{1,1}^{tr}), (\frac{1}{2}, \frac{1}{2})].
 \end{aligned}
 \tag{6.12}$$

These are six equations in nine control points. In the Euclidean space, the equations are linearly dependent such that only five equations are independent. Therefore, one can choose four of the nine control points as independent variables, and the five other follow from the conditions.



(a) C^1 -conditions at the interfaces (b) Renaming of the control points

Fig. 6.5 (a): C^1 -conditions (6.12) at the interface of four Bézier surfaces. Along each dashed line, the middle point should be the average of the end points. (b): Renaming scheme for the control points for Property 6.14.

In the Riemannian setting, however, the linear dependence of the six equations turns into an incompatibility: in general it is not possible to satisfy all six equations (unless several control points collapse to a single point in \mathcal{M}). A special situation occurs in symmetric spaces. Proposition 6.14 shows that, in those spaces, one can always find a set of control points satisfying the six equations (6.12) without collapsing to the same point. How-

Recall of p. 132. \mathcal{M} is symmetric if $\forall x \in \mathcal{M}$ there exists an isometry $I_x : \mathcal{M} \rightarrow \mathcal{M}$ mapping any geodesic $\gamma : [0, 1] \rightarrow \mathcal{M}$ with $\gamma(\frac{1}{2}) = x$ onto $I_x(\gamma) : t \mapsto \gamma(1-t)$.

ever, one can only choose three of the nine control points freely instead of four. For simplicity of the presentation of this Proposition, the control points are renamed as

$$b_{i,K+j}^{\text{br}} = b_{K+i,j}^{\text{tl}} = b_{ij}^{\text{tr}} = b_{K+i,K+j}^{\text{bl}} = a_{ij},$$

for $i, j \in \{-1, 0, 1\}$ (see Figure 6.5b). The control points with indices below 0 or beyond K simply indicate control points in the neighboring Bézier surface; a_{ij} refers to control points in a specific area where the four patches meet, around the corner a_{00} .

Proposition 6.14 (C^1 -conditions on \mathcal{M}). *Let \mathcal{M} be a symmetric space. Consider the nine control points a_{ij} , $i, j \in \{-1, 0, 1\}$.*

differentiability
conditions

- (i) *Let $a_{0,0}, a_{i_1,j_1}, a_{i_2,j_2}$ be given with $(i_1, j_1) \neq -(i_2, j_2)$, $|i_1| = |j_1|$. One can find the other six control points such that (6.12) is satisfied.*
- (ii) *If instead four control points are given initially, those can in general not be complemented with five other control points such that (6.12) is satisfied.*

Proof. To prove (i), let us first complement the six equations (6.12) with two additional “diagonal” equations

$$\begin{aligned} a_{0,0} &= \text{av} \left[(a_{-1,-1}, a_{1,1}), \left(\frac{1}{2}, \frac{1}{2}\right) \right], \\ a_{0,0} &= \text{av} \left[(a_{-1,1}, a_{1,-1}), \left(\frac{1}{2}, \frac{1}{2}\right) \right]. \end{aligned}$$

Fixing $a_{0,0}$, this makes eight equations for eight control points. Each equation expresses one control point as the midpoint on the geodesic between two other control points. Given an isometry $I_{a_{0,0}}$ on the symmetric space, the equations of the form $a_{0,0} = \text{av}[(a_{ij}, a_{-i,-j}), (\frac{1}{2}, \frac{1}{2})]$ establish the relation

$$a_{ij} = I_{a_{0,0}}(a_{-i,-j}) \quad i, j \in \{-1, 0, 1\}.$$

By this identity, the remaining equations

$$a_{-1,0} = \text{av} \left[(a_{-1,-1}, a_{-1,1}), \left(\frac{1}{2}, \frac{1}{2}\right) \right] \quad \text{and} \quad a_{1,0} = \text{av} \left[(a_{1,-1}, a_{1,1}), \left(\frac{1}{2}, \frac{1}{2}\right) \right]$$

as well as

$$a_{0,-1} = \text{av} \left[(a_{-1,-1}, a_{1,-1}), \left(\frac{1}{2}, \frac{1}{2}\right) \right] \quad \text{and} \quad a_{0,1} = \text{av} \left[(a_{-1,1}, a_{1,1}), \left(\frac{1}{2}, \frac{1}{2}\right) \right]$$

are redundant by symmetry. Thus, there are actually only two unknown

control points left, the others being fixed by the above identity. It is therefore straightforward to check that two of them may be chosen (in addition to $a_{0,0}$ to determine all the other.

To prove (ii), a simple counterexample suffices. Consider the unit sphere $\mathcal{M} = S^2$. We set four control points $a_{-1,1} = a_{1,1} = [0 \ 0 \ 1]^T$, $a_{0,0} = [\frac{\sqrt{2}}{2} \ 0 \ \frac{\sqrt{2}}{2}]^T$, and $a_{-1,0} = [\frac{1}{2} \ -\frac{1}{2} \ \frac{\sqrt{2}}{2}]^T$ (Figure 6.6). Choosing four out of six equations in (6.12), it is straightforward to compute the following values

$$a_{0,1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad a_{0,-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad a_{-1,-1} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}, \quad a_{1,-1} = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}.$$

If we now set $a_{1,0} = \text{av}[(a_{1,-1}, a_{1,1}), (\frac{1}{2}, \frac{1}{2})]$, then the constraint $a_{0,0} = \text{av}[(a_{-1,0}, a_{1,0}), (\frac{1}{2}, \frac{1}{2})] := \tilde{a}_{0,0}$ is violated. □

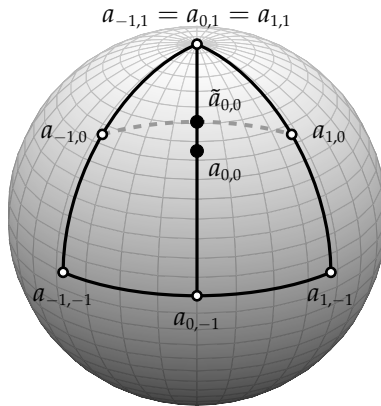


Fig. 6.6 Illustration of the counterexample from Proposition 6.14.

Compatibility of the C^1 equations around $a_{0,0}$ was the first challenge to achieve C^1 continuity of composite Bézier surfaces of type I to III. A second challenge comes with the fact that C^1 continuity is not necessarily achieved even if it was possible to find control points satisfying the conditions (6.12). This observation is illustrated in the top line of Figure 6.8.

Proposition 6.15 (Insufficiency of C^1 -conditions). *The conditions (6.10) are not sufficient to ensure C^1 -continuity of the composite Bézier surface (6.9) of type I, II, or III.*

Proof. We first give a counterexample for types I and III. When $K = 1$, σ_K^I and σ_K^{III} coincide and each Bézier surface is just obtained by a weighted geodesic averaging between four control points. Consider $\mathcal{M} = \mathbb{R}^2$ and six control points $b_{ij} = (i, j)$, with $i \in \{0, 1, 2\}$ and $j \in \{0, 1\}$. Based on those control points, we build the following composite Bézier surface

$$\mathbf{S} : [0, 2] \times [0, 1] \rightarrow \mathcal{M} : \\ (t_1, t_2) \mapsto \begin{cases} \sigma_1(t_1, t_2; b_{0,0}, b_{0,1}, b_{1,0}, b_{1,1}) & \text{if } t_1 \in [0, 1], \\ \sigma_1(t_1 - 1, t_2; b_{1,0}, b_{1,1}, b_{2,0}, b_{2,1}) & \text{if } t_1 \in (1, 2], \end{cases}$$

as illustrated in Figure 6.7a. Furthermore, we modify smoothly the metric

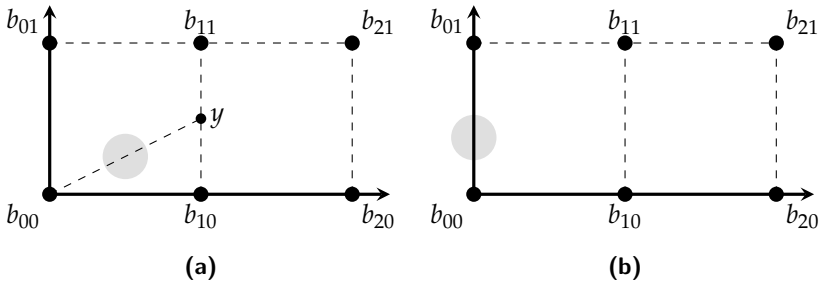


Fig. 6.7 Illustration of the two manifolds from the counterexamples in Proposition 6.15. A gray region indicates a deviation from the Euclidean metric.

of \mathbb{R}^2 between $b_{0,0}$ and $y = (1, \frac{1}{2})$, such that it is slightly decreased compared to the Euclidean metric. Hence, the geodesic from b_{00} to y is still a straight curve in \mathbb{R}^2 but has a shorter length than in the Euclidean space. Consider the position at time $(t_1, t_2) = (1, \frac{1}{2})$, at the interface of the two surfaces. Hence, $b := \mathbf{S}(t_1, t_2)$ can be expressed either as

$$\text{av} [(b_{00}, b_{01}, b_{10}, b_{11}), ((1 - t_1)(1 - t_2), (1 - t_1)t_2, t_1(1 - t_2), t_1t_2)]$$

or as

$$\text{av} [(b_{10}, b_{11}, b_{20}, b_{21}), ((2 - t_1)(1 - t_2), (2 - t_1)t_2, (t_1 - 1)(1 - t_2), (t_1 - 1)t_2)].$$

Obviously, $b = \mathbf{S}(1, \frac{1}{2}) = (1, \frac{1}{2})$. The optimality conditions for both averages are [Kar77, Thm. 1.2]

$$F(t_1, t_2, b) := (1 - t_1)(1 - t_2) \log_b(b_{00}) + (1 - t_1)t_2 \log_b(b_{01}) + t_1(1 - t_2) \log_b(b_{10}) + t_1t_2 \log_b(b_{11}) = 0$$

and

$$(2 - t_1)(1 - t_2) \log_b(b_{10}) + (2 - t_1)t_2 \log_b(b_{11}) + (t_1 - 1)(1 - t_2) \log_b(b_{20}) + (t_1 - 1)t_2 \log_b(b_{21}) = 0,$$

where $\log_b(b_{i,j}) = b_{i,j} - b$ except for $\log_b(b_{0,0}) = \zeta(b_{0,0} - b)$ for some $\zeta \in (0, 1)$ (as the metric is reduced). The latter equation is expressed on the Euclidean space with the canonical inner product, so it can be solved and it yields $b = \mathbf{S}(t_1, t_2) = (2 - t_1)(1 - t_2)b_{1,0} + (2 - t_1)t_2b_{1,1} + (t_1 - 1)(1 - t_2)b_{2,0} + (t_1 - 1)t_2b_{2,1}$. The right derivative $\frac{\partial \mathbf{S}}{\partial t_1}$ at $(t_1, t_2) = (1, \frac{1}{2})$ equals $(1, 0)$. The left derivative, however, is a little bit more intricate. By the implicit function theorem, we have

$$\begin{aligned} \frac{\partial \mathbf{S}}{\partial t_1} &= -(D_b F(1, \frac{1}{2}, b))^{-1} D_{t_1} F(1, \frac{1}{2}, b) = D_{t_1} F(1, \frac{1}{2}, b) \\ &= -\frac{\log_b(b_{00}) + \log_b(b_{01}) - \log_b(b_{10}) - \log_b(b_{11})}{2} \\ &= \zeta \begin{bmatrix} \frac{1}{2} \\ \frac{1}{4} \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{4} \end{bmatrix} \neq \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \end{aligned}$$

Here we have used that around $b = (1, \frac{1}{2})$ the function F is given by $F(1, \frac{1}{2}, b) = \frac{1}{2} \log_b(b_{10}) + \frac{1}{2} \log_b(b_{11}) = \frac{1}{2}(b_{10} - b) + \frac{1}{2}(b_{11} - b)$. Thus, the Bézier surfaces of type *I* and *III* are not glued C^1 -continuously, even though all conditions (6.10) are satisfied.

Consider now Bézier surfaces of type *II*, and the composite surface built in the same way as above. The only difference is that, now, the Euclidean metric is slightly decreased somewhere between b_{00} and $(b_{00} + b_{01})/2$. Hence, the connecting geodesic from b_{00} to b_{01} is given by $t \mapsto (0, f(t))$ with $f(t) > t$ in the interval $(\frac{1}{2}, 1)$ (Figure 6.7b). We have $\beta_1(t_2; b_{i0}, b_{i1}) = (i, t_2)$ for $i \in \{1, 2\}$ and $\beta_1(t_2; b_{00}, b_{01}) = (0, f(t_2))$. Obviously, for $t_1 \geq 1$ we obtain $\mathbf{S}(t_1, t_2) = (t_1, t_2)$, while for $t_1 \in [0, 1]$ and $t_2 \in (\frac{1}{2}, 1)$ we have $\mathbf{S}(t_1, t_2) = (t_1, (1 - t_1)f(t_2) + t_1t_2)$, yielding a discontinuous first derivative at $t_1 = 1$. □

Recall of p. 103, implicit function theorem: let $y = f(x)$ and $F(x, f(x)) = 0$, for f and F smooth. Hence, $\frac{\partial f}{\partial x} = -D_y F(x, y)^{-1} \frac{\partial F}{\partial x}$.

In the following subsection, we propose a modification of the definition of Bézier surfaces of type *I* and *II* in order to achieve C^1 -continuity. Unfortunately, we couldn't find any straightforward remedy for the Bézier surface of type *III*, *i.e.*, we didn't find a manner to transform the De Casteljau algorithm in a way that is (i) consistent with Bézier surfaces in Euclidean space and, (ii), allows to have C^1 -continuity.

6.3.3 C^1 -patching solution

We propose now a way to overcome the different problems presented in the previous section. The goal is to glue together MN Bézier surfaces in a C^1 manner. The control points of each Bézier surface are noted as follows. Let $b_{ij}^{mn} \in \mathcal{M}$ be the ij^{th} control point of the mn^{th} Bézier surface, where $i, j \in \{0, \dots, K\}$ and $(m, n) \in \{0, \dots, M-1\} \times \{0, \dots, N-1\}$. The associated composite Bézier surface is thus

$$\mathbf{S}^Y : [0, M] \times [0, N] \rightarrow \mathcal{M} : (t_1, t_2) \mapsto \sigma_K^Y(t_1 - m, t_2 - n; (b_{ij}^{mn})_{i,j=0,\dots,K}) \quad (6.13)$$

for $\lfloor t_1 \rfloor = m$, $\lfloor t_2 \rfloor = n$, defined as in Definition 3.6, and for $Y = I, II$.

We define indices outside the usual range as

$$b_{-1,j}^{mn} = b_{K-1,j}^{m-1,n}, \quad b_{K+1,j}^{mn} = b_{1,j}^{m+1,n}, \quad b_{j,-1}^{mn} = b_{j,K-1}^{m,n-1}, \quad \text{and} \quad b_{j,K+1}^{mn} = b_{j,1}^{m,n+1}.$$

Let us additionally define fictional points beyond the domain boundary: along t_1 , we define

$$\begin{aligned} b_{-1,j}^{0,n} &= \exp_{b_{0,j}^{0,n}} \left(-\log_{b_{0,j}^{0,n}} \left(b_{1,j}^{0,n} \right) \right), \\ b_{K+1,j}^{M,n} &= \exp_{b_{K,j}^{M,n}} \left(-\log_{b_{K,j}^{M,n}} \left(b_{K-1,j}^{M,n} \right) \right), \end{aligned}$$

for $j = 0, \dots, K$ and $n = 0, \dots, N$. Along t_2 , we define further

$$\begin{aligned} b_{i,-1}^{m,0} &= \exp_{b_{i,0}^{m,0}} \left(-\log_{b_{i,0}^{m,0}} \left(b_{i,1}^{m,0} \right) \right), \\ b_{i,K+1}^{m,N} &= \exp_{b_{i,K}^{m,N}} \left(-\log_{b_{i,K}^{m,N}} \left(b_{i,K-1}^{m,N} \right) \right), \end{aligned}$$

for $i = -1, \dots, K+1$ and $m = 0, \dots, M$.

control point

In fact, in Euclidean space the conditions

$$b_{K,j}^{mn} = \frac{b_{K-1,j}^{mn} + b_{K+1,j}^{mn}}{2}$$

imply that the control points $b_{K,j}^{mn}$ can be ignored altogether; indeed, in (6.1) one may simply replace any $b_{K,j}^{mn}$ by $\frac{b_{K-1,j}^{mn} + b_{K+1,j}^{mn}}{2}$. The analogous holds true for the control points $b_{i,K}^{mn}$, $b_{i,0}^{mn}$, and $b_{0,j}^{mn}$. This trick will restore C^1 -continuity in the Riemannian setting, as detailed below. This strategy has also its interest in the Euclidean setting as it permits to neglect conditions (6.10) and obtain a differentiable composite Bézier surface nevertheless. Following this idea, the generalized Bézier surfaces of type *I* and *II* are defined as follows.

Definition 6.16 (*Generalized C^1 -Bézier surfaces of type I*). The generalized Bézier surface of type *I* is expressed as a weighted average of b_{ij}^{mn} , $i, j \in \mathcal{I}$, with $\mathcal{I} = \{-1, 1, 2, \dots, K-1, K+1\}$. Namely,

$$\sigma_K^I(t_1, t_2; (b_{ij}^{mn})_{i,j=0,\dots,K}) = \text{av} \left[(b_{ij}^{mn})_{i,j \in \mathcal{I}}, (w_i(t_1)w_j(t_2))_{i,j \in \mathcal{I}} \right]. \quad (6.14)$$

The associated weights are given by

$$w_i(t) = \begin{cases} \frac{1}{2}B_{0K}(t) & \text{if } i = -1, \\ B_{1K}(t) + \frac{1}{2}B_{0K}(t) & \text{if } i = 1, \\ B_{iK}(t) & \text{if } i = 2, \dots, K-2, \\ B_{K-1,K}(t) + \frac{1}{2}B_{KK}(t) & \text{if } i = K-1, \\ \frac{1}{2}B_{KK}(t) & \text{if } i = K+1, \\ 0 & \text{if } i \in \{0, K\}. \end{cases}$$

Similarly, in (6.2), the one-dimensional Bézier curves can be computed just based on b_{ij}^{mn} , $i, j \in \mathcal{I}$. We use the same trick to modify the definition of the generalized Bézier surface of type *II*.

Definition 6.17 (*Generalized C^1 -Bézier surfaces of type II*). The generalized Bézier surface of type *II* is based on the control points b_{ij}^{mn} , $i, j \in \mathcal{I} = \{-1, 1, 2, \dots, K-1, K+1\}$, and is expressed as

$$\sigma_K^{II}(t_1, t_2; (b_{ij}^{mn})_{i,j=0,\dots,K}) = \beta_K(t_1; (\beta_K(t_2; (b_{ij}^{mn})_{j=0,\dots,K}))_{i=0,\dots,K}). \quad (6.15)$$

The one-dimensional Bézier curves are either computed as

$$\beta_K(t; (b_j)_{j=0, \dots, K}) = \text{av}[(b_j)_{j \in \mathcal{I}}, (w_j(t))_{j=0, \dots, K}],$$

or via an alternative modification of the De Casteljau algorithm where $x_0^{[K]} := \beta_K(t; (b_k)_{k=0, \dots, K})$:

$$\begin{aligned} x_0^{[0]} &= \text{av}[(b_{-1}, b_1), (\frac{1}{2}, \frac{1}{2})], \\ x_j^{[0]} &= b_j, && \text{for } j = 1, \dots, K - 1, \\ x_K^{[0]} &= \text{av}[(b_{K-1}, b_{K+1}), (\frac{1}{2}, \frac{1}{2})], \\ x_j^{[k]} &= \text{av}[(x_j^{[k-1]}, x_{j+1}^{[k-1]}), (1 - t, t)], && \text{for } k = 1, \dots, K, \\ &&& \text{and } j = 0, \dots, K - k. \end{aligned}$$

Note that every Bézier surface is a smooth function into the multi-geodesically convex hull of the control points, under the same conditions as in Theorem 6.11.

In the Euclidean space, all definitions are equivalent to the original Bézier surface as long as the control points satisfy the conditions of C^1 -continuity. In the manifold case, where the conditions (6.12) can at most approximately be satisfied, the definitions will lead to C^1 -continuity as shown in Figure 6.8 and in Theorem 6.18.

Theorem 6.18. *The composite Bézier surface (6.13) of type I (resp. type II) defined as in Definition 6.16 (resp. Definition 6.17) is C^1 -continuous.*

differentiability conditions

Proof. Each single patch is smooth by Theorem 6.11. Furthermore, C^0 -continuity follows as before. It remains to show that the normal derivatives at all interfaces between two adjacent Bézier surfaces coincide. To do so, consider the interface between the surfaces $(0, 0)$ and $(1, 0)$ at $t_1 = 1$ and $t_2 \in [0, 1]$ (the proof for the other interfaces works analogously). Consider also the set $\mathcal{I} = \{-1, 1, 2, \dots, K - 1, K + 1\}$. In the case of Bézier surfaces of type I, we have

$$\begin{aligned} b &:= \mathbf{S}^I(t_1 = 1, t_2) = \text{av} \left[(b_{ij}^{0,0})_{i,j \in \mathcal{I}}, (w_i(1)w_j(t_2))_{i,j \in \mathcal{I}} \right] \\ &= \text{av} \left[(b_{ij}^{1,0})_{i,j \in \mathcal{I}}, (w_i(0)w_j(t_2))_{i,j \in \mathcal{I}} \right]. \end{aligned}$$

6 | Interpolation with Bézier surfaces

The optimality conditions for both averages are [Kar77, Thm. 1.2]:

$$F_1(t_1, t_2, b) := \sum_{i,j \in \mathcal{I}} w_i(t_1) w_j(t_2) \log_b \left(b_{ij}^{0,0} \right) = 0,$$

$$F_2(t_1, t_2, b) := \sum_{i,j \in \mathcal{I}} w_i(t_1 - 1) w_j(t_2) \log_b \left(b_{ij}^{1,0} \right) = 0.$$

By the implicit function theorem, the left and right derivatives of \mathbf{S}^I with respect to t_1 at $(1, t_2)$ are given by

$$\begin{aligned} \left. \frac{\partial \mathbf{S}^I}{\partial t_1} \right|_{(1^-, t_2)} &= -(D_b F_1(1, t_2, b))^{-1} D_{t_1} F_1(1, t_2, b) \\ &= -(D_b F_1(1, t_2, b))^{-1} \left[\frac{K}{2} \sum_{j \in \mathcal{I}} w_j(t_2) \left(\log_b \left(b_{K+1,j}^{0,0} \right) - \log_b \left(b_{K-1,j}^{0,0} \right) \right) \right], \\ \left. \frac{\partial \mathbf{S}^I}{\partial t_1} \right|_{(1^+, t_2)} &= -(D_b F_2(1, t_2, b))^{-1} D_{t_1} F_2(1, t_2, b) \\ &= -(D_b F_2(1, t_2, b))^{-1} \left[\frac{K}{2} \sum_{j \in \mathcal{I}} w_j(t_2) \left(\log_b \left(b_{1,j}^{1,0} \right) - \log_b \left(b_{-1,j}^{1,0} \right) \right) \right]. \end{aligned}$$

As $F_1(1, t_2, b) = F_2(1, t_2, b)$ for all $b \in \mathcal{M}$ and by the definition of the $b_{ij}^{0,0}$ for $i \notin \{0, \dots, K\}$, we see directly that

$$\left. \frac{\partial \mathbf{S}^I}{\partial t_1} \right|_{(1^-, t_2)} = \left. \frac{\partial \mathbf{S}^I}{\partial t_1} \right|_{(1^+, t_2)}.$$

In the case of *surfaces of type II*, we note that [PN07]

$$\frac{\partial}{\partial t} \beta_K(t = 0; (b_j)_{j=0, \dots, K}) = \log_b(b_1)$$

for $b = \text{av}[(b_{-1}, b_1), (\frac{1}{2}, \frac{1}{2})]$ and

$$\frac{\partial}{\partial t} \beta_K(t = 1; (b_j)_{j=0, \dots, K}) = \log_b(b_{K+1})$$

for $b = \text{av}[(b_{K-1}, b_{K+1}), (\frac{1}{2}, \frac{1}{2})]$. Setting

$$b(t_2) = \text{av} \left[\left(\beta_K(t_2; (b_{K-1,j}^{0,0})_{j=0, \dots, K}), \beta_K(t_2; (b_{K+1,j}^{0,0})_{j=0, \dots, K}) \right), \left(\frac{1}{2}, \frac{1}{2} \right) \right],$$

we have

$$\frac{\partial \mathbf{S}^{II}}{\partial t_1} \Big|_{(1^-, t_2)} = \log_{\mathfrak{S}(t_2)} \left(\beta_K(t_2; (b_{K+1,j}^{0,0})_{j=0, \dots, K}) \right),$$

$$\frac{\partial \mathbf{S}^{II}}{\partial t_1} \Big|_{(1^+, t_2)} = \log_{\mathfrak{S}(t_2)} \left(\beta_K(t_2; (b_{1,j}^{1,0})_{j=0, \dots, K}) \right)$$

Again, the derivatives from either side coincide. □

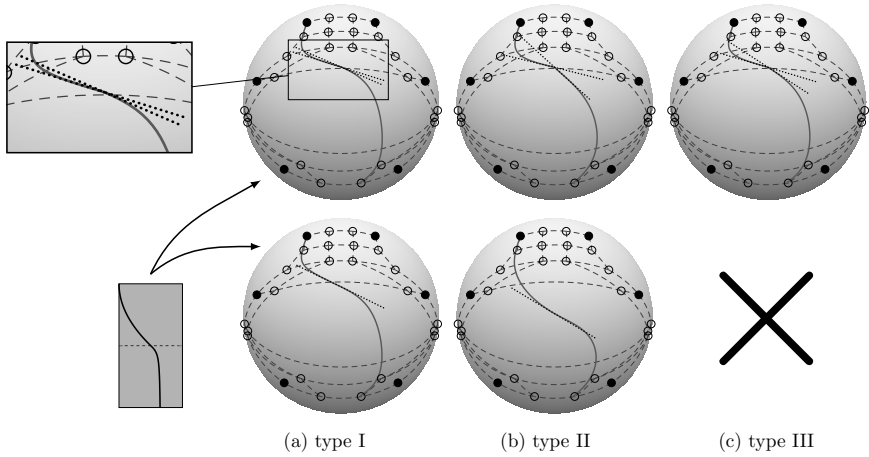


Fig. 6.8 The conditions (6.10) do not suffice to ensure differentiability (top), while the remedy from Theorem 6.18 permits to construct C^1 surfaces of type *I* and type *II* (bottom). Here, the composite Bézier surface is composed of two patches. The visualization is analogous to Figure 6.2, left: a smooth curve γ from the parameter domain is mapped to S^2 via the composite Bézier surface. The control points are represented with empty circles and the corners of each patch are the filled circles. The grayed dashed lines represent the control points grid. The derivatives of $\mathbf{S}(\gamma)$ at the interface are represented with the dotted lines, which are tangents to $\mathbf{S}(\gamma)$ from either side of the interface between the two Bézier patches. As no remedy could be provided for type *III*, no picture is given.

The next step is now to interpolate a set of data points with composite Bézier surfaces. The aim of the next section is to propose a way to choose control points such that the interpolating surface minimizes its mean squared acceleration when the manifold reduces to the Euclidean space.

6.4 Control points generation for surface interpolation

Given data points $d_{mn} \in \mathcal{M}$, $(m, n) \in \{0, \dots, M\} \times \{0, \dots, N\}$, the goal is to interpolate those by a smooth surface

$$\mathbf{S} : [0, M] \times [0, N] \rightarrow \mathcal{M}$$

with $\mathbf{S}(m, n) = d_{mn}$, consisting of C^1 -continuously patched *cubic* Bézier surfaces on each domain $[m, m+1] \times [n, n+1]$ as in (6.13). To this end we need to generate appropriate control points b_{ij}^{mn} for $m, n \in \{0, \dots, M-1\} \times \{0, \dots, N-1\}$ and $i, j = 0, \dots, 3$. The control points must respect the interpolation constraints

$$b_{0,0}^{mn} = d_{m,n}, \quad b_{3,0}^{mn} = d_{m+1,n}, \quad b_{0,3}^{mn} = d_{m,n+1}, \quad b_{3,3}^{mn} = d_{m+1,n+1}. \quad (6.16)$$

for $m = 0, \dots, M-1$, $n = 0, \dots, N-1$. They must also respect the C^0 -patching constraints of Theorem 6.12, *i.e.*,

$$b_{3,j}^{m,n} = b_{0,j}^{m+1,n} \text{ and } b_{j,3}^{m,n} = b_{j,0}^{m,n+1} \text{ for } j = 0, \dots, 3. \quad (6.17)$$

The resulting composite surface must be C^1 -smooth and must thus be generated by the approach of Theorem 6.18.

To make the interpolating surface as nice as possible, we would like to optimize the position of the control points such that the mean squared acceleration of the composite surface \mathbf{S} is minimized. Just like in Chapters 3 and 4, this is a highly nonlinear optimization problem. We thus follow the same simplified route as before: we formulate the problem in a Euclidean space, such that its solution can be expressed as a linear system. Then, we transfer this linear system to the manifold case.

6.4.1 Variational formulation of control point generation in \mathbb{R}^r

In the Euclidean space \mathbb{R}^r , we would like to minimize the objective function

$$\begin{aligned} F[\mathbf{S}] &= \int_{[0,M] \times [0,N]} \left\| \frac{\partial^2 \mathbf{S}}{\partial(t_1, t_2)} \right\|_{\mathbb{F}}^2 d(t_1, t_2) \\ &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \int_{[0,1] \times [0,1]} \left\| \frac{\partial^2 \sigma_3^{mn}}{\partial(t_1, t_2)} \right\|_{\mathbb{F}}^2 d(t_1, t_2), \end{aligned} \quad (6.18)$$

where $\|\cdot\|_F$ is the Frobenius norm, $\frac{\partial^2}{\partial(t_1, t_2)}$ is the Hessian operator for any bivariate function, and σ_3^{mn} is the cubic Bézier surface defined on the patch (m, n) and driven by the control points b_{ij}^{mn} . For notation simplicity, we will just denote the surface by σ or σ^{mn} when no confusion is possible. Note that the constraints (6.16) will automatically ensure the interpolation of the data points.

Each single Bézier surface patch can be expressed with Bernstein polynomials, according to (6.1). Then, the objective function turns into a quadratic function in the control points b_{ij}^{mn}

$$F[\mathbf{S}] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \hat{F}[\sigma^{mn}], \quad (6.19)$$

where the energy \hat{F} of a Bézier surface with control points b_{ij}^{mn} , $i, j \in \{0, \dots, 3\}$, is defined as

$$\hat{F}[\sigma^{mn}] = \sum_{i,j,o,p=0}^3 \alpha_{ijop} b_{ij}^{mn} \cdot b_{op}^{mn}. \quad (6.20)$$

Here, the symbol \cdot indicates the Euclidean dot product. Denoting the Frobenius inner product on 2×2 -matrices by $A : B = \sum_{i,j=1}^2 A_{ij} B_{ij}$, the coefficients α_{ijop} in the above energy are given by

$$\alpha_{ijop} = \int_{[0,1]^2} \left[\frac{\partial^2 B_{i3}(t_1) B_{j3}(t_2)}{\partial(t_1, t_2)} \right] : \left[\frac{\partial^2 B_{o3}(t_1) B_{p3}(t_2)}{\partial(t_1, t_2)} \right] d(t_1, t_2). \quad (6.21)$$

The explicit Hessian of the Bernstein polynomial products are given by

$$\frac{\partial^2 B_{i3}(t_1) B_{j3}(t_2)}{\partial(t_1, t_2)} = \begin{pmatrix} \frac{\partial^2 B_{i3}(t_1) B_{j3}(t_2)}{\partial^2 t_1} & \frac{\partial B_{i3}(t_1)}{\partial t_1} \frac{\partial B_{j3}(t_2)}{\partial t_2} \\ \frac{\partial B_{i3}(t_1)}{\partial t_1} \frac{\partial B_{j3}(t_2)}{\partial t_2} & B_{i3}(t_1) \frac{\partial^2 B_{j3}(t_2)}{\partial^2 t_2} \end{pmatrix}. \quad (6.22)$$

Note that the coefficients α_{ijop} can be readily computed analytically and are independent of the configuration.

To later be able to transfer this formulation to the manifold setting, every control point should be expressed as its difference with the four interpolation points of the patch. These differences will be later translated into Riemannian logarithms. Since the objective function $F[\mathbf{S}]$ only contains derivatives, the contributions $\hat{F}[\sigma]$ of its single Bézier patches are invariant

under a uniform translation of the control points. Hence

$$\hat{F}[\sigma^{mn}] = \frac{1}{4} \sum_{r,s \in \{0,1\}} \sum_{i,j,o,p=0}^3 \alpha_{ijop} v_{ij}^{mn}(r,s) \cdot v_{op}^{mn}(r,s), \quad (6.23)$$

where we introduced the auxiliary variables for the differences (see Figure 6.9)

$$v_{ij}^{mn}(r,s) = b_{ij}^{mn} - d_{m+r,n+s} \quad (6.24)$$

for $i, j = 0, \dots, 3, r, s = 0, 1, m = 0, \dots, M - 1$ and $n = 0, \dots, N - 1$. Note that for symmetry reasons we shifted the control points by each corner of the corresponding patch and then took the average of the four energy values resulting from those four shifts.

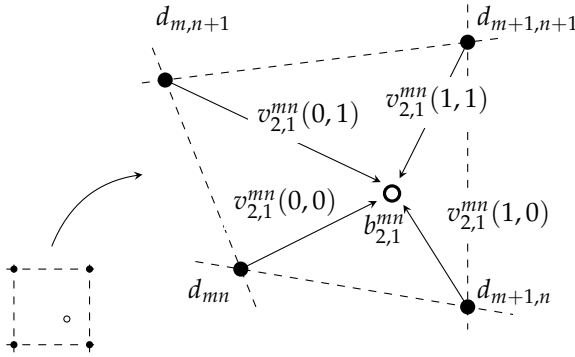


Fig. 6.9 Geometric interpretation of variables $v_{ij}^{mn}(r,s)$.

To summarize, the total energy can be expressed as

$$F[\mathbf{S}] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{r,s \in \{0,1\}} \sum_{i,j=0}^3 (L(V))_{ij,rs}^{mn} \cdot v_{ij}^{mn}(r,s) \quad (6.25)$$

with $V = (v_{ij}^{mn}(r,s))_{\substack{(m,n) \in \{0, \dots, M-1\} \times \{0, \dots, N-1\} \\ i,j \in \{0, \dots, 3\}, r,s \in \{0,1\}}}$ and the linear operator L

$$(L(V))_{ij,rs}^{mn} = \frac{1}{4} \sum_{o,p=0}^3 \alpha_{ijop} v_{op}^{mn}(r,s). \quad (6.26)$$

This energy has to be minimized for the control points b_{ij}^{mn} or equivalently the $v_{ij}^{mn}(r,s)$ under constraints (6.16), (6.17), and (6.12).

6.4.2 System reduction by constraint elimination

To minimize (6.25) as a simpler unconstrained problem, we eliminate the interpolation constraints (6.16), the continuity constraints (6.17), and the differentiability constraints (6.12). To this end, without loss of generality, we observe that one just has to keep three independent control points, and all the other follow from the constraints. More precisely, consider the set $D = \{0, \dots, M\} \times \{0, \dots, N\}$, and the control points

$$b_{kl}^{mn} \text{ with } (k, l) \in Q = \{(1, 0), (0, 1), (1, 1)\} \text{ and } (m, n) \in D.$$

In the Euclidean space, the constraints (6.16), (6.17) and (6.12) uniquely determine all the remaining control points. Note that, again, the notations are extended to allow control points indices outside $\{0, \dots, 3\}$ (see Figure 6.5a), which is convenient as the constraints are all expressed around each data point. That is $b_{kl}^{mn} = b_{k,3+l}^{m,n-1} = b_{3+k,l}^{m-1,n} = b_{3+k,3+l}^{m-1,n-1}$ for $k, l \in \{-1, 0, 1\}$. Observe that the points b_{ij}^{mn} with $(m, n) \notin \{0, \dots, M-1\} \times \{0, \dots, N-1\}$ are just fictive additional control points.

To prepare the transfer to the manifold setting, we replace equivalently the b_{kl}^{mn} by their translated version

$$u_{kl}^{mn} = b_{kl}^{mn} - d_{mn}, \text{ for } (k, l) \in Q, \text{ and } (m, n) \in D, \quad (6.27)$$

and consider them as the optimization variables. Then, constructing the energy variables $v_{ij}^{mn}(r, s)$ in terms of these u_{kl}^{mn} requires two operators.

First, it requires a linear operator S that generates the remaining vectors u_{kl}^{mn} for $(k, l) \in \{-1, 0, 1\}^2 \setminus Q$ around each data point d_{mn} based on the constraints (see Figure 6.10). That is, the operator $S : (u_{kl}^{mn})_{(k,l) \in Q}^{(m,n) \in D} \mapsto (u_{kl}^{mn})_{k,l \in \{-1,0,1\}}^{(m,n) \in D}$ is defined as follows:

$$S(u_{kl}^{mn}) = \begin{cases} u_{-1,1}^{mn} & = 2u_{0,1}^{mn} - u_{1,1}^{mn}, \\ u_{0,0}^{mn} & = 0, \\ u_{-1,0}^{mn} & = -u_{1,0}^{mn}, \\ u_{-1,-1}^{mn} & = -u_{1,1}^{mn}, \\ u_{0,-1}^{mn} & = -u_{0,1}^{mn}, \\ u_{1,-1}^{mn} & = 2u_{1,0}^{mn} - u_{1,1}^{mn}, \\ u_{kl}^{mn} & = u_{kl}^{mn} \text{ for } (k, l) \in Q. \end{cases} \quad (6.28)$$

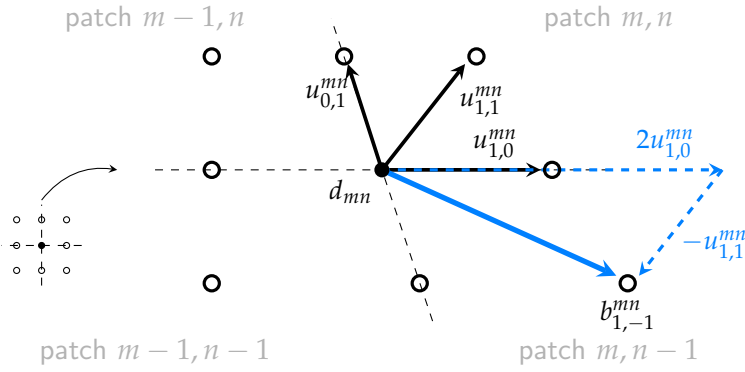


Fig. 6.10 Geometric interpretation of operator S . Based on the three black vectors, the five other are expressed thanks to the constraints (6.16), (6.17) and (6.12) (here, the example is given for $u_{1,-1}^{mn}$).

At this step, one has access to one vector u_{kl}^{mn} for each corresponding control point b_{kl}^{mn} . However, the energy variables $v_{ij}^{mn}(r, s)$ split up those u_{kl}^{mn} among the four corners of a given patch. This is done by the operator \tilde{T} (see Figure 6.11). The operator \tilde{T} maps thus the set of vectors u_{kl}^{mn} onto the set of vectors $v_{ij}^{mn}(r, s)$ by exploiting the relation

$$v_{ij}^{mn}(r, s) = u_{kl}^{\tilde{m}\tilde{n}} + (d_{\tilde{m}\tilde{n}} - d_{m+r, n+s}),$$

where \tilde{m} , \tilde{n} , k , and l satisfy

$$\begin{aligned} (\tilde{m}, \tilde{n}) &= (m + a_i, n + a_j) \\ (k, l) &= (i - 3a_i, j - 3a_j) \end{aligned} \quad \text{for } a_i = \begin{cases} 0 & \text{if } i \in \{0, 1\}, \\ 1 & \text{if } i \in \{2, 3\}. \end{cases}$$

To prepare the transfer to the manifold setting, we introduce the following intermediate notations:

$$w_{ij}^{mn}(r, s) = u_{kl}^{\tilde{m}\tilde{n}}, \tag{6.29}$$

$$z_{ij}^{mn}(r, s) = d_{\tilde{m}\tilde{n}} - d_{m+r, n+s}. \tag{6.30}$$

Those notations might seem cumbersome at first, but it is easy to recognize a Euclidean logarithm map in $z_{ij}^{mn}(r, s)$, which will be very useful later.

Abbreviating all variables with

$$\begin{aligned}
 U &:= (u_{kl}^{mn})_{k,l \in \{-1,0,1\}}^{(m,n) \in D} & V &:= (v_{ij}^{mn}(r,s))_{\substack{i,j=0,\dots,3 \\ r,s \in \{0,1\}}}^{(m,n) \in D} \\
 W &:= (w_{ij}^{mn}(r,s))_{\substack{i,j=0,\dots,3 \\ r,s \in \{0,1\}}}^{(m,n) \in D} & Z &:= (z_{ij}^{mn}(r,s))_{\substack{i,j=0,\dots,3 \\ r,s \in \{0,1\}}}^{(m,n) \in D}
 \end{aligned}$$

and introducing the linear operator

$$T : U \mapsto W \quad (6.31)$$

we finally obtain that \tilde{T} is also linear and is defined as

$$\tilde{T} : U \mapsto V = T(U) + Z. \quad (6.32)$$

Note that the operator T can be interpreted to operate on each patch (m, n) separately as follows: T translates any vector $u_{kl}^{\tilde{m}\tilde{n}}$, which belongs to a control point of the patch, from its base point $d_{\tilde{m}\tilde{n}}$ to all four patch corners $d_{m+r, n+s}$, $(r, s) \in \{0, 1\}^2$, resulting in the four new vectors $w_{ij}^{mn}(r, s)$, $r, s \in \{0, 1\}$. Of course, in Euclidean space this translation is trivial, however, it will turn into a nontrivial parallel transport in the manifold setting.

The map \tilde{T} is illustrated in Figure 6.11.

The total energy (6.25) of Bézier curves in Euclidean space can finally be rewritten as

$$F[\mathbf{S}] = \frac{1}{4} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{r,s \in \{0,1\}} \sum_{i,j=0}^3 (L\tilde{T}S\tilde{U})_{ij,rs}^{mn} \cdot (\tilde{T}S\tilde{U})_{ij,rs}^{mn} \text{ for } \tilde{U} = (u_{kl}^{mn})_{(k,l) \in Q}^{(m,n) \in D}$$

and is minimized by

$$\tilde{U}_{\text{opt}} = -(S^*T^*LTS)^{-1}(S^*T^*LZ), \quad (6.33)$$

where a superscript asterisk denotes the adjoint operator (note that L is self-adjoint). See also Appendix D for a proof.

6.4.3 Transfer to the manifold setting

It remains to transfer the different operators and the final formula (6.33) for control points generation to a Riemannian manifold setting.

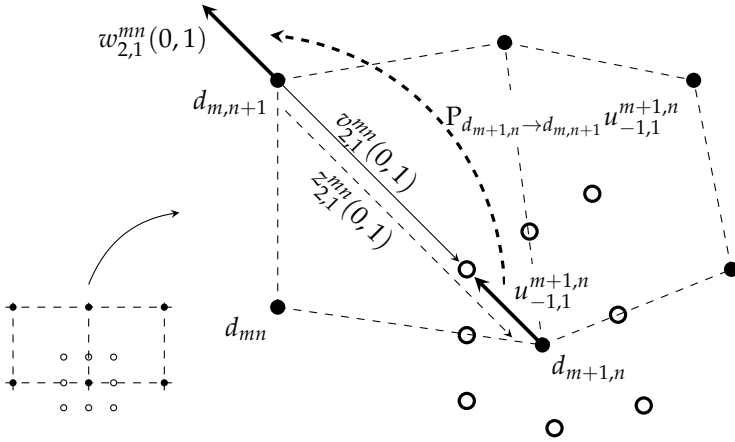


Fig. 6.11 Geometric interpretation of operator \tilde{T} . The variables $v_{ij}^{mn}(r, s)$ (cf. Figure 6.9) are constructed as follow: (i) a vector u_{kl}^{mn} in d_{mn} is transported to another interpolation point d , and (ii) to this vector one adds the difference between d and d_{mn} .

- The u_{kl}^{mn} were defined in (6.27) as the difference between two Euclidean points. Their generalization to the Riemannian setting is given by

$$u_{kl}^{mn} = \log_{d_{mn}}(b_{kl}^{mn}),$$

for $k, l \in Q$, $(m, n) \in D$. This means that we will actually optimize over tangent vectors u_{kl}^{mn} to the manifold and only afterwards convert them into control points b_{kl}^{mn} via the exponential map.

- The operator L defined in (6.26) can now be interpreted as an operator from B into itself, where B is the Cartesian product of tangent spaces

$$B = \prod_{\substack{i,j=0,\dots,3 \\ r,s \in \{0,1\} \\ m=0,\dots,M-1 \\ n=0,\dots,N-1}} T_{d_{m+r,n+s}} \mathcal{M}.$$

- The operator S in (6.28) is now considered as an operator on tangent spaces. Its formulation remains unchanged as for a fixed pair (m, n) , all vectors $u_{kl}^{mn} \in T_{d_{m,n}} \mathcal{M}$.
- Formula (6.30), which defines the components of Z , is generalized to

the manifold setting as

$$z_{ij}^{mn}(r, s) = \log_{d_{m+r, n+s}} d_{\tilde{m}\tilde{n}}. \quad (6.34)$$

- The operator T from (6.31) is redefined as the parallel transport of the variables u_{kl}^{mn} to the corners of the corresponding patch:

$$T : (u_{kl}^{mn})_{k,l \in \{-1,0,1\}}^{(m,n) \in D} \mapsto (w_{ij}^{mn}(r, s))_{\substack{i,j=0,\dots,3 \\ r,s \in \{0,1\}}}^{(m,n) \in D},$$

$$w_{ij}^{mn}(r, s) = P_{d_{\tilde{m}\tilde{n}} \rightarrow d_{m+r, n+s}} u_{kl}^{\tilde{m}\tilde{n}}.$$

The same notation as in (6.29) is used here.

- The operator \tilde{T} from (6.32) is transferred to the manifold setting using the manifold versions of T and Z .
- Last but not least, the adjoint operator S^* is given by

$$S^* : (u_{kl}^{mn})_{k,l \in \{-1,0,1\}}^{(m,n) \in D} \mapsto (u_{kl}^{mn})_{(k,l) \in Q}^{(m,n) \in D} :$$

$$S(u_{kl}^{mn}) := \begin{cases} u_{1,0}^{mn} &= u_{1,0}^{mn} - u_{-1,0}^{mn} + 2u_{1,-1}^{mn}, \\ u_{1,1}^{mn} &= u_{1,1}^{mn} - u_{-1,1}^{mn} - u_{-1,-1}^{mn} - u_{1,-1}^{mn}, \\ u_{0,1}^{mn} &= u_{0,1}^{mn} - u_{0,-1}^{mn} + 2u_{-1,1}^{mn}, \end{cases}$$

and the adjoint operator T^* is given by

$$T^* : (w_{ij}^{mn}(r, s))_{\substack{i,j=0,\dots,3 \\ r,s \in \{0,1\}}}^{(m,n) \in D} \mapsto (u_{kl}^{mn})_{k,l \in \{-1,0,1\}}^{(m,n) \in D},$$

$$u_{kl}^{\hat{m}\hat{n}} := \sum_{\substack{r,s \in \{0,1\} \\ m \in \hat{m} + A_k \\ n \in \hat{n} + A_l}} P_{d_{m+r, n+s} \rightarrow d_{\hat{m}\hat{n}}} w_{k+3(\hat{m}-m), l+3(\hat{n}-n)}^{mn}(r, s),$$

where $A_{-1} = \{-1\}$, $A_0 = \{-1, 0\}$, and $A_1 = \{0\}$.

The algorithm for generating the control points on a Riemannian manifold \mathcal{M} now proceeds as follows.

1. Compute $Z = (z_{ij}^{mn}(r, s))_{ijmnr}$ via (6.34).
2. Compute S^*T^*LZ and solve (6.33) for U_{opt} by a conjugate gradient iteration.

parallel
transport

6 | Interpolation with Bézier surfaces

3. Compute all u_{kl}^{mn} for $k, l \in \{-1, 0, 1\}$ and $(m, n) \in D$ via SU_{opt} .
4. Compute all control points $b_{kl}^{mn} \in \mathcal{M}$ for $k, l \in \{-1, 1\}$ and $(m, n) \in D$ via the exponential map at d_{mn} (see below for details). Note that all other control points are not used in the composite Bézier surface evaluation (6.14) or (6.15); they are thus irrelevant.

In the last step of the algorithm, the computation of the control points has to be performed in a way that ensures $\mathbf{S}(m, n) = d_{mn}$. This requires a different procedure for Bézier surfaces of type *I* or *II*.

For Bézier surfaces of type *I* we simply use

$$b_{kl}^{mn} = \exp_{d_{mn}}(u_{kl}^{mn}), \quad k, l \in \{-1, 1\}, (m, n) \in D.$$

Indeed, this automatically satisfies

$$\mathbf{S}^I(m, n) = \text{av} \left[(b_{-1,-1}^{mn}, b_{-1,1}^{mn}, b_{1,-1}^{mn}, b_{1,1}^{mn}), \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4} \right) \right] = d_{mn}.$$

For Bézier splines of type *II* we set

$$\begin{aligned} b_{k0}^{mn} &= \exp_{d_{mn}}(u_{k0}^{mn}), \\ b_{kl}^{mn} &= \exp_{b_{k0}^{mn}} \left(P_{d_{mn} \rightarrow b_{k0}^{mn}} u_{0l}^{mn} \right), \quad k, l \in \{-1, 1\}, (m, n) \in D. \end{aligned}$$

Interpolation is thus also achieved, as

$$\begin{aligned} b_{-1,0}^{mn} &= \text{av} \left[(b_{-1,-1}^{mn}, b_{-1,1}^{mn}), \left(\frac{1}{2}, \frac{1}{2} \right) \right] \\ b_{1,0}^{mn} &= \text{av} \left[(b_{1,-1}^{mn}, b_{1,1}^{mn}), \left(\frac{1}{2}, \frac{1}{2} \right) \right] \\ \mathbf{S}^{II}(m, n) &= \text{av} \left[(b_{-1,0}^{mn}, b_{1,0}^{mn}), \left(\frac{1}{2}, \frac{1}{2} \right) \right] = d_{mn}. \end{aligned}$$

An example is provided in Figure 6.12. Numerical examples can be found in Section 6.6.

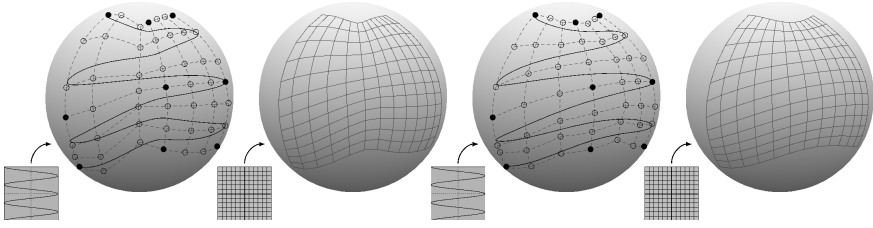


Fig. 6.12 Optimal placement of control points (circles) for given interpolation points (dots) obtained by the algorithm described in Section 6.4 and applied to the sphere S^2 . The left graphs show the configuration before optimization ($F[\mathbf{S}] = 8.185$): the control points b_{kl}^{mn} for $(k, l) \in Q$, $m, n \in \{0, 1\}$ are simply obtained by geodesic averaging between the interpolation points. The right graphs show the configuration after optimization ($F[\mathbf{S}] = 4.932$).

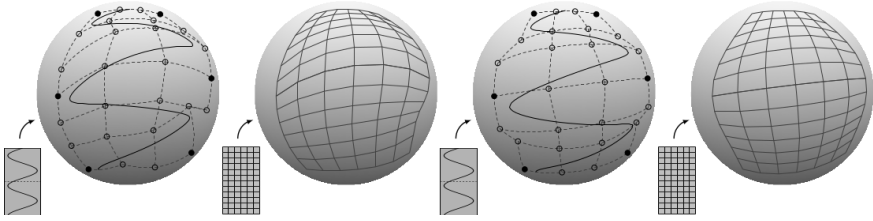


Fig. 6.13 Instead of transporting the vectors to all four corners of a patch (left graphs, $F[\mathbf{S}] = 15.96$), vectors could also be transported to a single corner in each patch (right graphs, where they are transported only to the bottom-left corner of each patch, $F[\mathbf{S}] = 22.16$). However, such a modification will in general yield configurations with a higher energy; furthermore, we can observe a lack of symmetry. As in the previous figure, two configurations are represented for each test: (i) the parameterization of the surface to a curve $\mathbf{S}(\gamma)$, for $\gamma : (t_1, t_2) \mapsto \gamma(t_1, t_2) \in \mathbb{R}^2$, and (ii), a grid of the domain.

6.5 Accelerated generation of control points

The generation of the control points from the previous section was inspired from the technique already applied in Chapters 3 and 4. However, we were forced to include parallel transportation to the algorithm, which is usually a costly operation. Indeed, we see in practice that the control points generation of Section 6.4 is a heavy procedure that must be performed offline.

control point

In this section, we consider an alternative method to accelerate the control points generation, based on the same idea as Bézier surfaces of type II. In a word, we express conditions to compute control points for curves, and extend this to surfaces afterwards. The acceleration comes also from two additional constraints. We restrict ourselves to (i) rely only on the exponential map and logarithm map (just like in Chapter 4) and (ii) reduce the number of data points taken into accounts while computing a given control point. This will be made clear in the following.

Let us first consider a Euclidean space and come back to composite Bézier curves for a time. Given points d_m in \mathbb{R}^r , there exists a unique C^2 -interpolating composite cubic Bézier curve \mathbf{B} whose second derivative direction vanishes at the domain boundary [Far02, §9.3] and which minimizes its mean squared acceleration $\int_0^M \|\mathbf{B}''(t)\|^2 dt$ among all interpolating curves [Far02, §9.5].

The B-spline representation of this optimal curve is $\mathbf{B} = \sum_{m=-1}^{M+1} \alpha_m \mathbf{B}_m$, with coefficients $\alpha_{-1}, \dots, \alpha_{M+1} \in \mathbb{R}^r$ and where $\mathbf{B}_m = \mathbf{B}(\cdot - m)$ given by (see Figure 6.14)

$$\mathbf{B}(t) = \begin{cases} \beta_3(t + 2; 0, 0, 0, \frac{1}{6}) & \text{if } t \in [-2, -1], \\ \beta_3(t + 1; \frac{1}{6}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}) & \text{if } t \in [-1, 0], \\ \beta_3(t - 0; \frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{6}) & \text{if } t \in [0, 1], \\ \beta_3(t - 1; \frac{1}{6}, 0, 0, 0) & \text{if } t \in [1, 2], \\ 0 & \text{else.} \end{cases} \quad (6.35)$$

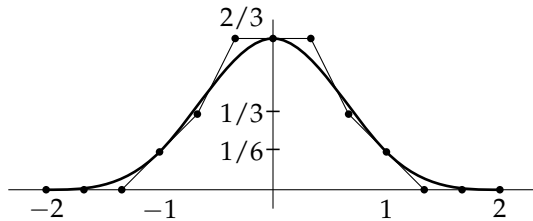


Fig. 6.14 B-spline representation of the optimal curve.

The constraints $\mathbf{B}(m) = d_m$ and $\mathbf{B}''(0) = \mathbf{B}''(M) = 0$ result in the linear

system

$$\frac{1}{6} \underbrace{\begin{bmatrix} 4 & 1 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & & 1 & 1 \\ & & & & 4 \end{bmatrix}}_{=:A^M} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix} = \underbrace{\begin{bmatrix} d_1 - \frac{d_0}{6} \\ d_2 \\ \vdots \\ d_{M-2} \\ d_{M-1} - \frac{d_M}{6} \end{bmatrix}}_{=:P^M(d_0, \dots, d_M)},$$

under the constraints $\alpha_0 = d_0$, $\alpha_M = d_M$, $\alpha_{-1} = 2\alpha_0 - \alpha_1$ and $\alpha_{M+1} = 2\alpha_M - \alpha_{M-1}$.

Inserting (6.35) into $\mathbf{B} = \sum_{m=-1}^{M+1} \alpha_m \mathbf{B}(\cdot - m)$, the control points b_j^m are given by

$$b_0^m = d_m, \quad b_1^m = \frac{2}{3}\alpha_m + \frac{1}{3}\alpha_{m+1}, \quad b_2^m = \frac{1}{3}\alpha_m + \frac{2}{3}\alpha_{m+1}, \quad b_3^m = d_{m+1}.$$

We can now transfer this approach to surfaces. The B-spline representation of a composite surface is

$$\mathbf{S} = \sum_{m=-1}^{M+1} \sum_{n=-1}^{N+1} \alpha_{mn} \mathbf{B}_{mn}$$

with $\mathbf{B}_{mn}(t_1, t_2) = \mathbf{B}_m(t_1)\mathbf{B}_n(t_2)$. Those basis elements are just tensorized versions of the 1D case. A natural way to find the coefficients $\alpha_{mn} \in \mathbb{R}^r$ is thus to first identify the coefficients of the $N + 1$ curves interpolating d_{0n}, \dots, d_{Mn} , $n = 0, \dots, N$, and then interpret those coefficients as interpolation points for the curves along the other dimension (or the other way around, of course).

The problems to solve are now, for all m and n ,

$$\begin{aligned} \tilde{\alpha}_{0n} &= d_{0n}, & \tilde{\alpha}_{Mn} &= d_{Mn}, & A^M(\tilde{\alpha}_{1n}, \dots, \tilde{\alpha}_{M-1,n})^\top &= P^M(d_{0n}, \dots, d_{Mn}), \\ \alpha_{0n} &= \tilde{\alpha}_{m0}, & \alpha_{Mn} &= \tilde{\alpha}_{mN}, & A^N(\alpha_{m1}, \dots, \alpha_{m,N-1})^\top &= P^N(\tilde{\alpha}_{m0}, \dots, \tilde{\alpha}_{mN}). \end{aligned}$$

Equivalently, one can first compute intermediate points in one shot for all (m, n) via

$$\mathbf{P}(d, m, n) := \tilde{d}_{mn} = P_m^M \left(P_n^N(d_{00}, \dots, d_{0N}), \dots, P_n^N(d_{M0}, \dots, d_{MN}) \right);$$

then, denoting by $\bar{A} = A^{-1}$, the α_{mn} are given by

$$\alpha_{mn} = \mathbf{A}(\bar{d}, m, n) = \sum_{i=1}^M \sum_{j=1}^N \bar{A}_{mi}^M \bar{A}_{nj}^N \bar{d}_{ij}.$$

We can note that the entries of \bar{A}^M and \bar{A}^N decay exponentially away from the diagonal. Hence, one could choose a small number $\tau \in \mathbb{N}$ and allow a small error in the optimal calculation. Hence, the optimal coefficients will thus be approximated as

$$\alpha_{mn} = \sum_{i=m-\tau}^{m+\tau} \sum_{j=n-\tau}^{n+\tau} \bar{A}_{mi}^M \bar{A}_{nj}^N \bar{d}_{ij},$$

where the summation is of course restricted to indices for which the entries are defined. Finally, the Bézier control points b_{ij}^{mn} for $i, j \in \{1, 2\}$ are obtained via

$$b_{ij}^{mn} = \frac{3-i}{3} \frac{3-j}{3} \alpha_{mn} + \frac{3-i}{3} \frac{j}{3} \alpha_{m,n+1} + \frac{i}{3} \frac{3-j}{3} \alpha_{m+1,n} + \frac{i}{3} \frac{j}{3} \alpha_{m+1,n+1}.$$

The transfer to the Riemannian setting is then direct. We observe that the equations stay valid under translations such that we can replace all α_{mn} and d_{mn} by respectively $\hat{\alpha}_{mn} = \alpha_{mn} - d_{\text{ref}}$ and $\hat{d}_{mn} = d_{mn} - d_{\text{ref}}$. In summary, we compute $\bar{d}_{mn} = \mathbf{P}(\hat{d}, m, n)$ and then obtain $\hat{\alpha}_{mn} = \mathbf{A}(\bar{d}, m, n)$.

On a Riemannian manifold \mathcal{M} , the Euclidean difference is replaced by the logarithm map at d_{ref} , such that every point is mapped to $T_{d_{\text{ref}}}\mathcal{M}$. In the computation of $\hat{\alpha}_{mn} = \log_{d_{\text{ref}}}(\alpha_{mn})$ one should choose $d_{\text{ref}} = d_{mn}$ as the closest interpolation point. The choice of a small τ now has the advantage that the computation requires only few logarithms $\log_{d_{\text{ref}}}(p_{\hat{m}\hat{n}})$ which are typically expensive to obtain and form the numerical bottleneck of the approach. At the end, $\alpha_{mn} \in \mathcal{M}$ is retrieved as $\alpha_{mn} = \exp_{d_{mn}}(\hat{\alpha}_{mn})$ and the control points for $i, j \in \{1, 2\}$ are computed as a weighted average

$$b_{ij}^{mn} = \text{av} \left[(\alpha_{mn}, \alpha_{m,n+1}, \alpha_{m+1,n}, \alpha_{m+1,n+1}), \left(\frac{3-i}{3} \frac{3-j}{3}, \frac{3-i}{3} \frac{j}{3}, \frac{i}{3} \frac{3-j}{3}, \frac{i}{3} \frac{j}{3} \right) \right].$$

The next section presents numerical examples of surface interpolation based on the control points generation from Sections 6.4 and 6.5.

6.6 Numerical examples

To finish this chapter, we finally present some examples of composite Bézier surfaces computed on different manifolds. For reasons of computational efficiency (especially on manifolds for which no closed formulae of Riemannian operators are available), the examples all represent a Bézier surface of type *II*.

6.6.1 On the the sphere S^2

A first simple computational example on the sphere S^2 has already been shown in Figure 6.12.

A possible application on this manifold is image transfer from the plane onto the sphere. Figure 6.15 shows a rectangular map of the world, which serves as the parameterization domain of a smooth composite Bézier surface on the sphere. The surface parameterization then provides a one-to-one map between points on the rectangle and points on the sphere, which can be used to map the world image onto the sphere.

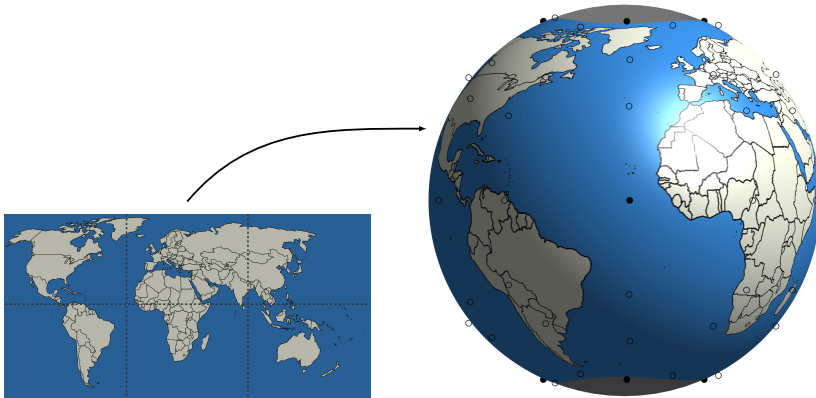


Fig. 6.15 A rectangular map of the world is smoothly mapped onto the sphere via a composite Bézier surface, only fixing a few interpolation points.

The composite curve is optimized via the algorithm from Section 6.4. Table E.1 recalls the explicit formulas of the Riemannian operators used to perform this on S^n .

As a comparison, Figure 6.16 shows a result on S^2 where the control points were computed with the efficient algorithm of Section 6.5.

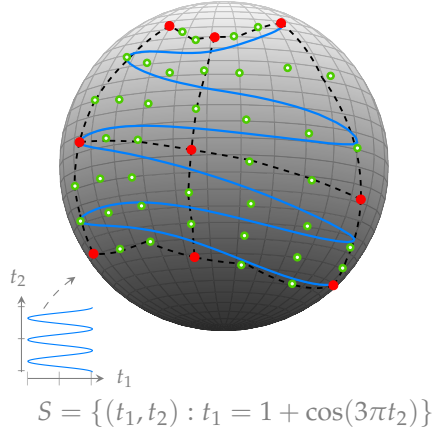


Fig. 6.16 Differentiable composite surface interpolation on S^2 . Interpolation points (dots) are shown in red; the control points were generated with the efficient algorithm of Section 6.5 and shown in green (circles).

6.6.2 On the special orthogonal group $SO(3)$

The special orthogonal group is used here to interpolate orientations. Figure 6.17 displays a composite cubic Bézier surface in $SO(3)$ interpolating a random set of truck-orientation (red). The surface was optimized based on the Section 6.4. The optimized control points are shown in green. Note that the surface is smooth and roughly follows the control points, but it does not go through them, as expected.

A second example is given in Figure 6.18. Here, the first line is composed of a point on $SO(3)$ rotated of $t_1 \times 90$ degrees around the z -axis. In the direction t_2 , the data points of the first line are rotated of $t_2 \times 90$ degrees around the x -axis, which gives a torus effect. The control points have also been optimized, but the method from Section 6.4 was slightly adapted to take the periodic boundary into account. We imposed that $u_{k,l}^{0,n} = u_{k,l}^{M,n}$ and $u_{k,l}^{m,0} = u_{k,l}^{m,N}$ for $k, l \in \{-1, 0, 1\}$ and $(m, n) \in \{0, \dots, M\} \times \{0, \dots, N\}$.

The Riemannian operators on $SO(n)$ are summarized in Table E.2.

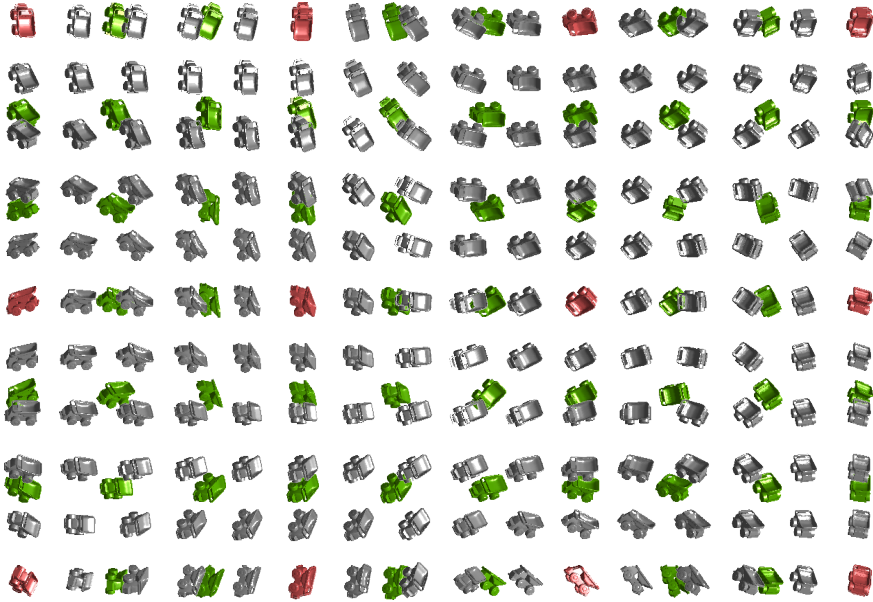


Fig. 6.17 Composite cubic Bézier surface in $SO(3)$ visualized as rotations of a little truck. Interpolation points are in red, optimized control points in green, points on the surface are displayed in gray.

Figure 6.19 displays a composite cubic Bézier surface in $SO(3)$ interpolating a random set of rotations (red).

6.6.3 On the space of open polygonal curves \mathcal{P}

We consider two shape spaces as further examples. The first is the shape space \mathcal{P} of all polygonal curves in the plane with a fixed number n of segments. Here, two shapes are considered equal if they differ only by a rigid motion. A shape $\gamma \in \mathcal{P}$ can therefore be identified with its segment length and angle representation $(\ell_1, \dots, \ell_n, \alpha_1, \dots, \alpha_{n-1}) \in \mathbb{R}^{2n-1}$, where ℓ_j denotes the length of the j^{th} polygon segment and α_j the angle between segment j and $j + 1$. The tangent space $T_\gamma \mathcal{P} = \mathbb{R}^{2n-1}$ can be seen as the space of all length and angle variations.

shape space \mathcal{P}

Instead of a Riemannian metric and with regard to Section 2.6, we directly specify an energy functional W acting on two shapes $\gamma_1, \gamma_2 \in \mathcal{P}$,

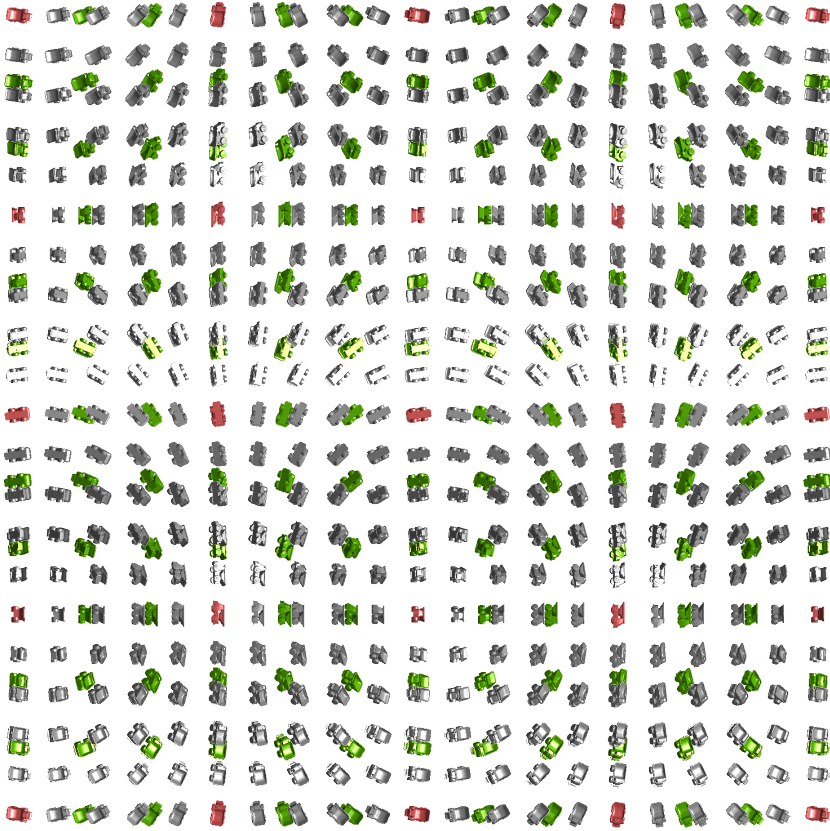


Fig. 6.18 Smooth torus in $SO(3)$ given by a composite cubic Bézier surface. The color-convention is identical as in Figure 6.17.

$\gamma_i = (\ell_1^i, \dots, \ell_n^i, \alpha_1^i, \dots, \alpha_{n-1}^i), i = 1, 2$, by

$$W[\gamma_1, \gamma_2] = \sum_{j=1}^n \frac{(\ell_j^1 - \ell_j^2)^2}{\ell_j^1} + 2 \sum_{j=1}^{n-1} \frac{(\alpha_j^1 - \alpha_j^2)^2}{\ell_j^1 + \ell_{j+1}^1}.$$

The Riemannian metric, for which this W is supposed to approximate the squared Riemannian distance, can be obtained as the second derivative,

$$g_y(v, w) = \frac{d}{dt_1} \frac{d}{dt_2} W(y, y + t_1 v + t_2 w) \Big|_{t_1=t_2=0}.$$

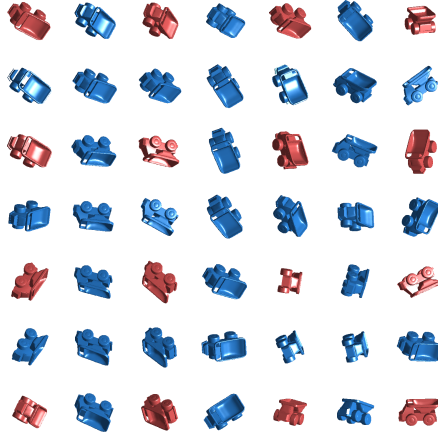


Fig. 6.19 Differentiable composite surface interpolation on $SO(3)$. Interpolation points are shown in red; the reconstructed surface is in blue. The control points (not displayed) were generated with the efficient algorithm of Section 6.5.

It has a physical interpretation of energy dissipation during shape deformation [RW15]. Our calculations are now based on the discrete approximations from Section 2.6.

Figure 6.20 shows a Bézier spline surface in the space of polygonal curves with optimized control points. The interpolation points are segments of silhouettes from the Kimia database [SKK04].

6.6.4 The space of discrete shells S_h

As a second example, we consider the space of shells S_h , as described in [HRWW12]. We give here a quick overview of the manifold before applying the interpolation techniques to it.

shape space S_h

In the continuous case, a shell S_h is given by an oriented C^2 surface S in \mathbb{R}^3 , called the *midplane* of S_h , and is defined as the set

$$S_h = \{p + \lambda\nu(p) \mid p \in S, \lambda \in \left(-\frac{h}{2}, \frac{h}{2}\right) \subset \mathbb{R}\}$$

where $\nu(p)$ denotes the normal at a point $p \in S$. The space of shells S_h comprises all images $\phi(S_h^{\text{ref}})$ of a reference shell S_h^{ref} under any orientation-preserving diffeomorphism ϕ .

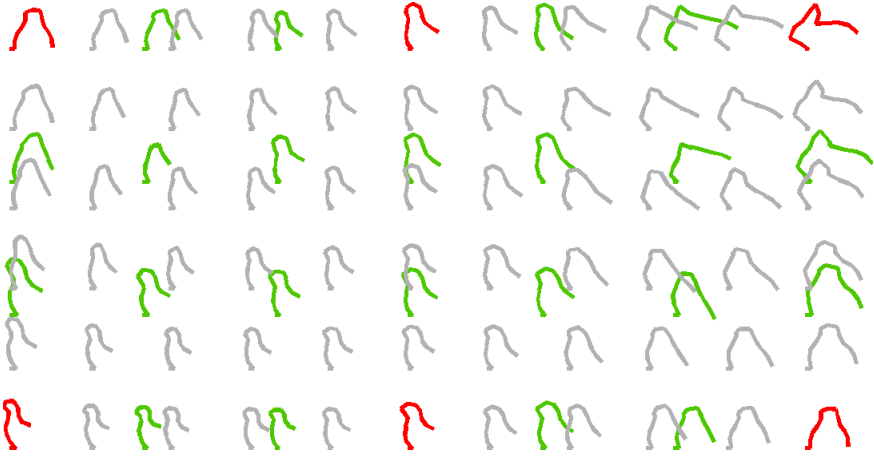


Fig. 6.20 Smooth interpolation on the space of open polygonal curves \mathcal{P} . Interpolation points are shown in red, control points in green, points on the surface in gray (shapes are from the Kimia database [SKK04]).

The tangent space at $S \in \mathcal{S}$ consists of smooth displacement fields $\psi : S \rightarrow \mathbb{R}^3$, and it can be equipped with a Riemannian metric that describes the physical energy dissipation during the deformation of S_h ; for details we refer to [HRWW12].

The discretized analog, also described in [HRWW12], is called *discrete shell* M_h . By “discrete”, we mean a “triangulated surface” in \mathbb{R}^3 , represented by a tuple $(N_h, T_h) \in (\mathbb{R}^3)^m \times (\{1, \dots, m\}^3)^n$, $m, n \in \mathbb{N}$. Here, N_h represents the vertex positions and T_h encodes the triangulation, *i.e.*, each component $(T_h)_l = (i, j, k)$ indicates that $(N_h)_i$, $(N_h)_j$ and $(N_h)_k$ form a triangle. The space of discrete shells can be equipped with a discrete analog of the Riemannian metric on \mathcal{S} . Given a triangle T , we assign to each vertex a local index ranging from 0 to 2. This allows us to define the edge set E_T of T as the set of directed edges connecting the nodes $i - 1$ and i (counted modulo 3). E_h is defined to be the union of the edge sets E_T over all triangles T .

A deformation of a discrete shell M_h can now be viewed as a mapping $\phi : N_h \rightarrow \mathbb{R}^3$. We define the discrete deformation energy $\tilde{W}[\phi]$ of ϕ by

$$\tilde{W}[\phi] = h \sum_{T \in T_h} W_{\text{mem}}(Q_{\text{mem}}^\top[\phi])A_T + h^3 \sum_{E \in E_h} W_{\text{bend}}(Q_{\text{bend}}^E[\phi])A_E$$

for some physical energy densities W_{mem} and W_{bend} (specific examples are given in [HRWW12]). Here, A_T denotes the area of the (undeformed) triangle T . For a given edge $E \in E_h$, the value $A_E = \frac{1}{3}(A_{T_1} + A_{T_2})$ denotes an area fraction of the two adjacent triangles T_1 and T_2 . The operator $Q_{\text{mem}}^\top[\phi]$ describes in-plane strain while $Q_{\text{bend}}^E[\phi]$ describes in-plane bending. We refer to [HRWW12] for details about those expressions.

The discrete geodesic calculus from Section 2.6 can now be employed with the energy

$$W[S_1, S_2] = \tilde{W}[\phi] \text{ for } \phi(S_1) = S_2,$$

which approximates the squared Riemannian distance in the space of discrete shells [HRWW12].

Figure 1.1 already showed a differentiable composite Bézier surface interpolating six given hand shapes (the mesh data was made available by Yeh *et al.* [YLSL11]). Similarly, Figure 6.21 shows a composite Bézier interpolation surface between 3×3 interpolation points (the interpolation points in this figure are meshes made available by Bergou *et al.* [BMWG07]). The control points in Figure 6.21 have been optimized using the algorithm from Section 6.4.

This section is based on the papers [AGSW16b] and [AGSW16a], sometimes cited verbatim. The references of these papers are

[AGSW16b] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striowski, and Benedikt Wirth. Differentiable piecewise-Bézier surfaces on Riemannian manifolds. *SIAM Journal on Imaging Sciences*, 9(4):1788–1828, 2016. doi:10.1137/16M1057978

and

[AGSW16a] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striowski, and Benedikt Wirth. Differentiable piecewise-Bézier interpolation on Riemannian manifolds. In *ESANN2016*, pages 95–100. Springer, 2016

The figures can be reproduced based on the code provided in the toolbox available at this link address:

<https://github.com/pgousenbourg/manint>

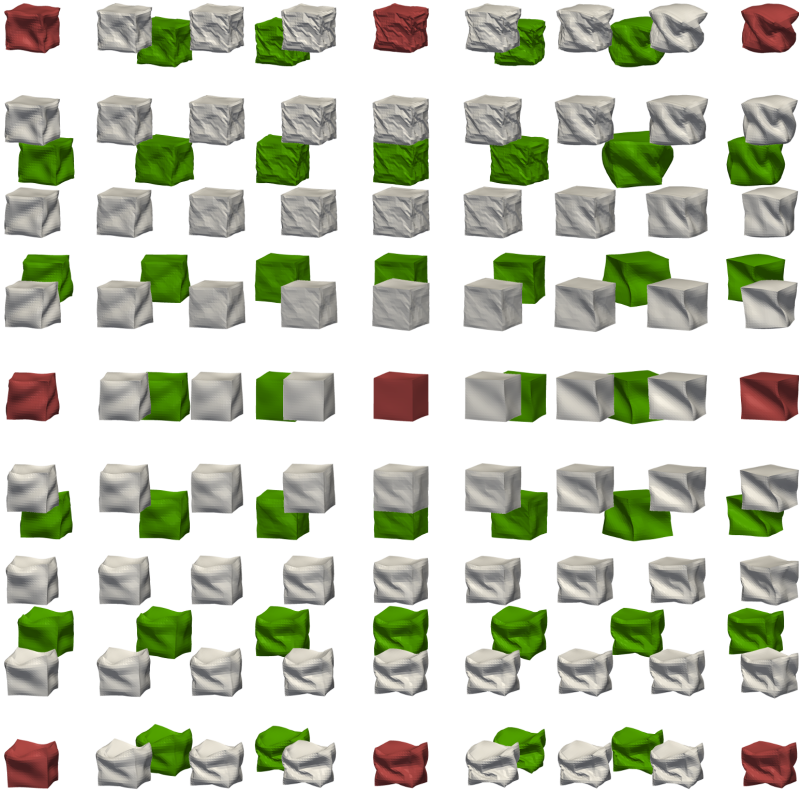


Fig. 6.21 Differentiable piecewise-Bézier interpolation in the space of triangulated shells. Interpolation points are shown in red, control points in green, points on the surface in gray. Dataset courtesy to Bergou *et al.* [BMWG07].

7

Fitting with blended surfaces

AS FINAL CHAPTER, and in order to close the loop, we consider fitting with composite surfaces. In essence, the goal of this chapter is to generalize the blended curves presented in Chapter 4 to the multivariate case. However, although only the bivariate case will be presented here (*i.e.*, surface fitting), the generalization to multivariate is quite straightforward. This is the last contribution of this thesis.

As in the previous chapters, given a set of data points $d_i \in \mathcal{M}$ associated with parameter values $t_i \in \Omega \subset \mathbb{R}^2$, one searches a surface $\mathbf{S}(t)$ that strikes the balance between fitting and regularity. It is now encapsulated within the following minimization problem on a domain $\Omega \subset \mathbb{R}^2$:

$$\min_{\mathbf{S}} \int_{\Omega \subset \mathbb{R}^2} \left\| D^2 \mathbf{S}(t) \right\|_{\mathbf{S}(t)}^2 dt + \lambda \sum_{i=0}^n d^2(\mathbf{S}(t_i), d_i). \quad (7.1)$$

As in Chapter 4, the algorithm to compute such a surface must meet six properties:

- (i) As $\lambda \rightarrow \infty$, the data points are interpolated at the associated parameter values;
- (ii) The surface is continuously differentiable;
- (iii) If \mathcal{M} is a Euclidean space, the surface reduces to the thin plate spline fitting the data points;

- (iv) The method only requires to use the exponential map and the logarithm map;
- (v) The number of tangent vectors to be stored is $\mathcal{O}(n)$, where n is the number of data points;
- (vi) The number of operations to reconstruct $\mathbf{S}(t)$ is $\mathcal{O}(1)$ once the tangent space representations are given.

The first section of this chapter is a reminder of the theory of thin plate splines, *i.e.*, a generalization of the natural (1D) smoothing splines [GS93]. Then the *blended surfaces* are presented and analyzed in Section 7.2. This constitutes the main contribution of this chapter, made in collaboration with Prof. Benedikt Wirth (Universität Münster, Germany). A proper publication is currently in production but the work is still ongoing [AGW20]. Finally, some applications are shown in Sections 7.3.

7.1 Euclidean thin plate splines

Thin plate splines in Euclidean space were introduced and analyzed in the late 1970s [Duc77, Duc78, Mei79, Pow94] as a generalization of the natural smoothing spline. They constitute now a very standard tool for multivariate fitting and interpolation; they appear in various textbook on data fitting, for instance in Green and Silverman [GS93, Chap. 7] to which we refer for a complete introduction.

Thin plate splines take their name from a specific 2D case of a larger problem. Consider a Hilbert space and data points $d_0, \dots, d_n \in \mathbb{R}$ associated with parameter values $t_0, \dots, t_n \in \Omega \subset \mathbb{R}^m$. A smooth interpolation $\mathbf{S} : \Omega \rightarrow \mathbb{R}$ of the pairs $(t_i, d_i) \in \Omega \times \mathbb{R}$ can be found in a variational manner by minimizing a derivative-penalizing cost function $E[\mathbf{S}]$

$$\min_{\mathbf{S}} E[\mathbf{S}] = \min_{\mathbf{S}} \int_{\Omega} \left\| D^k \mathbf{S}(t) \right\|_{\mathbb{F}}^2 dt \text{ such that } \mathbf{S}(t_i) = d_i, \quad (7.2)$$

where $D^k \mathbf{S}$ stands for the k^{th} derivative of \mathbf{S} .

When $\Omega \in \mathbb{R}^2$ and $k = 2$, in particular, a unique optimizer of (7.2) exists [Duc77]. The term *thin plate spline* derives from this setting. In that case, $E[\mathbf{S}]$ can be interpreted as the physical bending energy within a thin flat metal sheet with out-of-plane displacement \mathbf{S} . Noting $t = (x, y)$, the

objective E becomes thus explicitly [GS93]

$$E[\mathbf{S}] = \int_{\Omega} \left[\left(\frac{\partial^2 \mathbf{S}(t)}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 \mathbf{S}(t)}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 \mathbf{S}(t)}{\partial y^2} \right)^2 \right] dx dy \quad (7.3)$$

On a finite domain Ω , the Euler–Lagrange equations of the optimizer read

$$\begin{aligned} \Delta^2 \mathbf{S} &= 0 && \text{in } \Omega \setminus \{t_1, \dots, t_K\}, \\ u \cdot D^2 \mathbf{S} u &= 0, \quad \partial_u \Delta \mathbf{S} + \partial_{u^\perp} (u^\perp \cdot D^2 \mathbf{S} u) &= 0 && \text{on } \partial\Omega, \end{aligned}$$

where Δ^2 is the bilaplacian operator, u denotes the unit outward normal to the boundary $\partial\Omega$ of Ω , u^\perp the tangent normal, ∂_u the normal derivative, and ∂_{u^\perp} the tangential derivative. The boundary conditions make that the problem (7.2) must be solved numerically. However, when $\Omega = \mathbb{R}^2$ is not bounded, there are no boundary conditions and the minimizer can be computed explicitly in the form of a *thin plate spline*.

The bilaplacian operator is given by $\Delta^2 = \sum_i \frac{\partial^4}{\partial x_i^4} + 2 \sum_{i < j} \frac{\partial^4}{\partial x_i^2 \partial x_j^2}$

Definition 7.1 (*Thin plate spline in \mathbb{R}^2*). Consider a dataset of coordinates t_0, \dots, t_n in \mathbb{R}^2 . A *thin plate spline* is a function \mathbf{S} of the form

thin plate spline

$$\mathbf{S} : \mathbb{R}^2 \rightarrow \mathbb{R} : \mathbf{S}(t) := \sum_{i=0}^n a_i \phi(\|t - t_i\|) + \delta \begin{bmatrix} t \\ 1 \end{bmatrix}$$

where $a_i \in \mathbb{R}$ and $\delta \in \mathbb{R}^3$ are coefficients to be chosen (see Definitions 7.2 and 7.3), and where

$$\phi : \mathbb{R}_+ \rightarrow \mathbb{R} : r \mapsto \phi(r) := \begin{cases} \frac{1}{16\pi} r^2 \log r^2 & \text{for } r > 0 \\ 0 & \text{for } r = 0 \end{cases}$$

is a radial basis kernel. If $\sum_{i=0}^n a_i = \sum_{i=0}^n a_i t_i = 0$, then \mathbf{S} is called a *natural thin plate spline*.

It can be shown [GS93, Thm. 7.1] that the only thin plate splines \mathbf{S} with finite energy $E[\mathbf{S}]$ are natural thin plate splines.

Like in the case of Bézier curves (see Chapters 3 and 6), the thin plate spline is defined based on its coefficients a_i and δ . From their values will depend interpolation.

Definition 7.2 (*Interpolating thin plate splines [GS93, §7.5]*). Consider the pairs $(d_i; t_i) = (d_i; x_i, y_i) \in \mathbb{R} \times \mathbb{R}^2$, $i = 0, \dots, n$, with the t_i distinct and

interpolation

non-collinear. The unique thin plate spline \mathbf{S} satisfying $\mathbf{S}(t_i) = d_i$ has coefficients given by

$$\begin{bmatrix} G & T^\top \\ T^\top & 0 \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix} = \begin{bmatrix} D \\ 0 \end{bmatrix},$$

where $a \in \mathbb{R}^{n+1}$ is the vector of coefficients a_i , $D \in \mathbb{R}^{n+1}$ is the vector of data points d_i , $G \in \mathbb{R}^{(n+1) \times (n+1)}$ is a matrix filled with $G_{ij} = \phi(\|t_i - t_j\|)$, and $T \in \mathbb{R}^{3 \times (n+1)}$ is a matrix of values defined as

$$T = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ x_0 & x_1 & \cdots & x_{n-1} & x_n \\ y_0 & y_1 & \cdots & y_{n-1} & y_n \end{bmatrix}.$$

The uniqueness of this definition is proven in [GS93, Thm. 7.2], and the optimality of this surface with respect to (7.2) is proven in [GS93, Thm. 7.3].

fitting

Consider now the fitting case. The data has no more to be interpolated but just approached as $\mathbf{S}(t_i) \approx d_i$. Introducing the fitting parameter $\lambda > 0$, (7.2) becomes

$$\min_{\mathbf{S}} E[\mathbf{S}] + \lambda \sum_{i=1}^n \|\mathbf{S}(t_i) - d_i\|_{\mathbb{F}}^2. \tag{7.4}$$

Note that, when $\lambda \rightarrow \infty$, the problem approaches its interpolation equivalent. When Ω is not bounded, the minimizer is also a natural thin plate spline. The coefficients are given as follows.

Definition 7.3 (*Fitting thin plate splines [GS93, §7.6]*). Consider the pairs $(d_i; t_i) = (d_i; x_i, y_i) \in \mathbb{R} \times \mathbb{R}^2$, $i = 0, \dots, n$, with the t_i distinct and non-collinear. The unique thin plate spline \mathbf{S} minimizing (7.4) has coefficients given by

$$\begin{bmatrix} G + \frac{1}{\lambda}I & T^\top \\ T^\top & 0 \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix} = \begin{bmatrix} D \\ 0 \end{bmatrix},$$

where a , D , G and T are defined as in Definition 7.2.

Remark 7.4. Thin plate splines can be seen as natural generalizations of smoothing curves. In his work, Duchon gives a bound on the error of approximation made when a smooth surface $S : \Omega \rightarrow \mathbb{R}$ is approximated by a thin plate spline \mathbf{S} interpolating the pairs $(S(t_i); t_i)$. We refer the reader to [Duc78, Prop. 3]. The error bound also evaluates the difference between solutions whose support Ω is finite or not.

Some illustrations of fitting and interpolating thin plate splines are given

in Figure 7.1, for different values of λ .

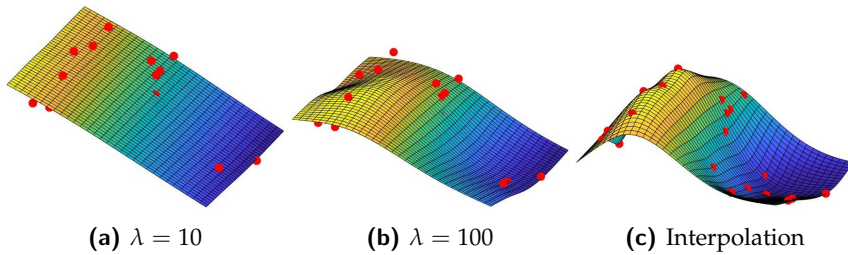


Fig. 7.1 Thin plate splines in \mathbb{R}^3 . The red points are data points.

7.2 Fitting with blended thin plate splines

In Chapter 4, the problem (4.1) was never solved directly on manifolds, but instead replaced by different approximations on local tangent spaces. The same arises here with problem (7.1): solving the thin plate spline problem is typically easy and computationally efficient on a Euclidean space, but much harder and computationally expensive for data in a manifold \mathcal{M} .

Therefore, the natural idea from Chapter 4 is extended to the bivariate case. The manifold \mathcal{M} is approximated by several tangent spaces $T_{p_i}\mathcal{M}$ around linearization points $p_i \in \mathcal{M}$, $i = 0, \dots, N$. On those tangent spaces, a thin plate spline is constructed with Definitions 7.2 or 7.3. Then, the different solutions have to be merged properly such that the resulting surface \mathbf{S} satisfies the six aforementioned properties.

In Section 7.2.1, we present the blending technique of surfaces computed on multiple tangent spaces. The blending is done by Riemannian averaging. In Section 7.2.2, the regularity properties of such blended surfaces are examined. Section 7.2.3 is a discussion about the choice of the blending weights. Finally, Section 7.2.4 summarizes the ideas presented; a proof that the six properties (mentioned in the beginning of the chapter) are validated is also given.

7.2.1 Blended surfaces

Forget for a moment the data points and consider simply a Riemannian manifold \mathcal{M} and a rectangular domain $\Omega \subset \mathbb{R}^2$. The abstract problem is to construct a parameterized surface $\mathbf{S} : \Omega \rightarrow \mathcal{M}$ based only on surfaces computed in tangent spaces to \mathcal{M} . To do so, let Ω be discretized by a grid of $M \times N$ rectangular patches $\Omega_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ with $i = 0, \dots, M$ and $j = 0, \dots, N$ and let each node $t_{ij} = (x_i, y_j)$ be associated with a linearization point $p_{ij} \in \mathcal{M}$ (see Figure 7.2). For convenience, one could set $t_{ij} = (i, j)$, as in the previous Chapters, but let us work in full generality this time.

Locally around $t_{ij} = (x_i, y_j)$, \mathcal{M} will be well approximated by $T_{p_{ij}}\mathcal{M}$; thus, the surface \mathbf{S} will be represented using the tangent space $T_{p_{ij}}\mathcal{M}$. Note that, for this reason and in case $\mathbf{S}(x, y)$ is used to approximate a surface $f(x, y)$, it will be in general a good idea to pick $p_{ij} = f(x_i, y_j)$. The representation of \mathbf{S} in its local tangent space is called a *tangent surface*.

Definition 7.5 (*Tangent surface and local surface*). Let \mathcal{M} be a smooth manifold and $p_{ij} \in \mathcal{M}$ a linearization point. A *tangent surface* associated to p_{ij} is a continuous surface

$$\tilde{S}_{ij} : \Omega \rightarrow T_{p_{ij}}\mathcal{M}.$$

When the tangent surface is mapped back to the manifold \mathcal{M} , the corresponding (manifold-valued) surface is called the *local surface* and is defined as

$$S_{ij} : \Omega \rightarrow \mathcal{M} : t \mapsto S_{ij}(t) = \exp_{p_{ij}}(\tilde{S}_{ij}(t)). \tag{7.5}$$

Each local surface $S_{ij} : \Omega \rightarrow \mathcal{M}$, associated to the linearization point p_{ij} , is a good approximation of the final surface \mathbf{S} only in a neighborhood of (x_i, y_j) if $p_{ij} \approx \mathbf{S}(x_i, y_j)$. Therefore, we can see them as local patches that now need to be connected with each other in order to form the global surface \mathbf{S} . However, whenever one switches from one local representation S_{ij} to another, one would at least expect the resulting surface \mathbf{S} to be continuous (and ultimately smooth, up to a given degree). This is done by a computationally cheap blending mechanism, that will mix several local surfaces in a similar way as in Chapter 4.

In the present setting, we will consider that the “neighborhood of validity of S_{ij} ” is the domain $[x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}]$ (see the gray area in Figure 7.2). Therefore, on the subdomain $\Omega_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$, only four local surfaces are local representations of \mathbf{S} , namely $S_{i,j}$, $S_{i+1,j}$, $S_{i,j+1}$, and $S_{i+1,j+1}$.

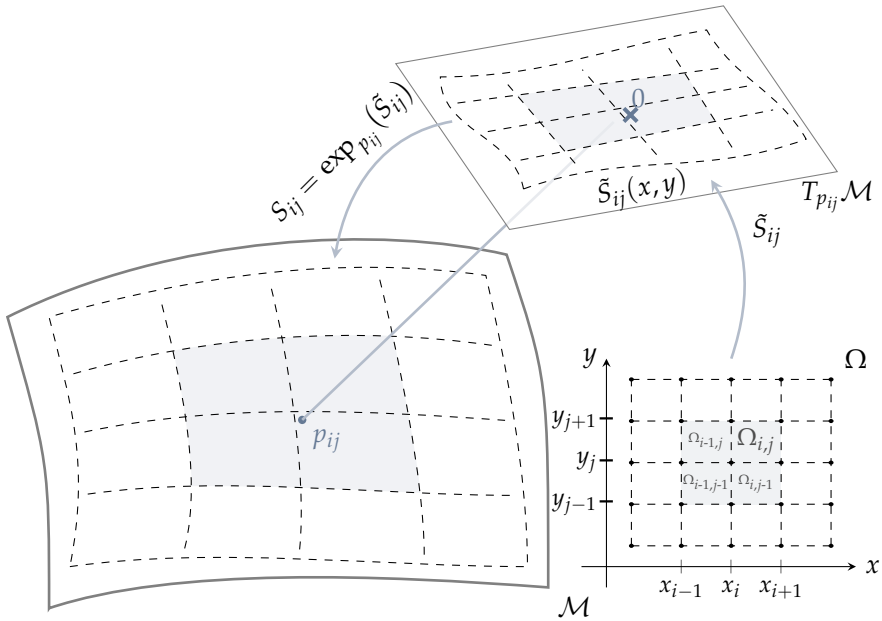


Fig. 7.2 The local surface S_{ij} is obtained by linearization of \mathcal{M} around p_{ij} by its tangent space $T_{p_{ij}}\mathcal{M}$, where a tangent surface \tilde{S}_{ij} is defined. Then \tilde{S}_{ij} is mapped back to \mathcal{M} to yield S_{ij} , which is interpreted as a local representation of the global surface \mathcal{S} on $[x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}]$ (grayed area).

Those four local surfaces will be blended with each other by weighted averaging (see Figure 7.3). The blending weights w_{kl} are chosen such that, the closer to (x_k, y_k) , the more reliable each surface S_{kl} gets, in comparison with the others (for a discussion on the choice of the weights, see Section 7.2.3).

Definition 7.6 (*Blended surface*). Let $\Omega_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ denote the subdomains of Ω . Each corner (x_k, y_l) is associated with a linearization points p_{kl} and with a local surface $S_{kl} : \Omega \rightarrow \mathcal{M} : t \mapsto S_{kl}(t) = \exp_{p_{kl}}(\tilde{S}_{kl}(t))$, for $k \in \{i, i+1\}$, $l \in \{j, j+1\}$. The *blended surface* $\mathbf{S} : \Omega \rightarrow \mathcal{M}$ is defined on each Ω_{ij} as

$$\mathbf{S}(x, y) = \text{av}[S_{ij}(x, y), S_{i+1,j}(x, y), S_{i,j+1}(x, y), S_{i+1,j+1}(x, y); w_{ij}(x, y), w_{i+1,j}(x, y), w_{i,j+1}(x, y), w_{i+1,j+1}(x, y)],$$

where av is the averaging operator from Definition 2.53 and where $w_{kl} : \Omega \rightarrow [0, 1]$ are weight functions with support in $[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]$. Furthermore, $\sum_{i=0}^M \sum_{j=0}^N w_{ij} = 1$. The blended surface is illustrated in Figure 7.3.

Note that, as final goal, the surface \mathbf{S} will be meant to fit a set of data points. Then, naturally, \tilde{S}_{ij} will be the thin plate spline evaluated on $T_{p_{ij}}\mathcal{M}$ and the local descriptor S_{ij} of \mathbf{S} around (x_i, y_j) , will be the tangent thin plate spline \tilde{S}_{ij} mapped back to \mathcal{M} .

Remark 7.7. The averaging operator from Definition 7.6 can be expressed in multiple ways, each of them leading to slightly different solutions (see Section 6.2 for an application to Bézier surfaces, and Figure 6.2 for an illustration). On one hand, the averaging can be the minimizer of the weighted sum of squared distance

$$\text{av}_{\min}[x_1, \dots, x_4; w_1, \dots, w_4] = \underset{x \in \mathcal{M}}{\text{argmin}} \sum_{i=1}^4 w_i d_{\mathcal{M}}^2(x_i, x).$$

This form has typically no closed-form solution and requires to solve a numerically expensive optimization. On the other hand, this averaging can be approached with an iterative dyadic version

$$\text{av}_{\text{dyad}}[x_1, \dots, x_4; w_1, \dots, w_4] = \text{av}[y_1, y_3; w_1 + w_2, w_3 + w_4] \\ \text{for } y_i = \text{av}[x_i, x_{i+1}; w_i, w_{i+1}]$$

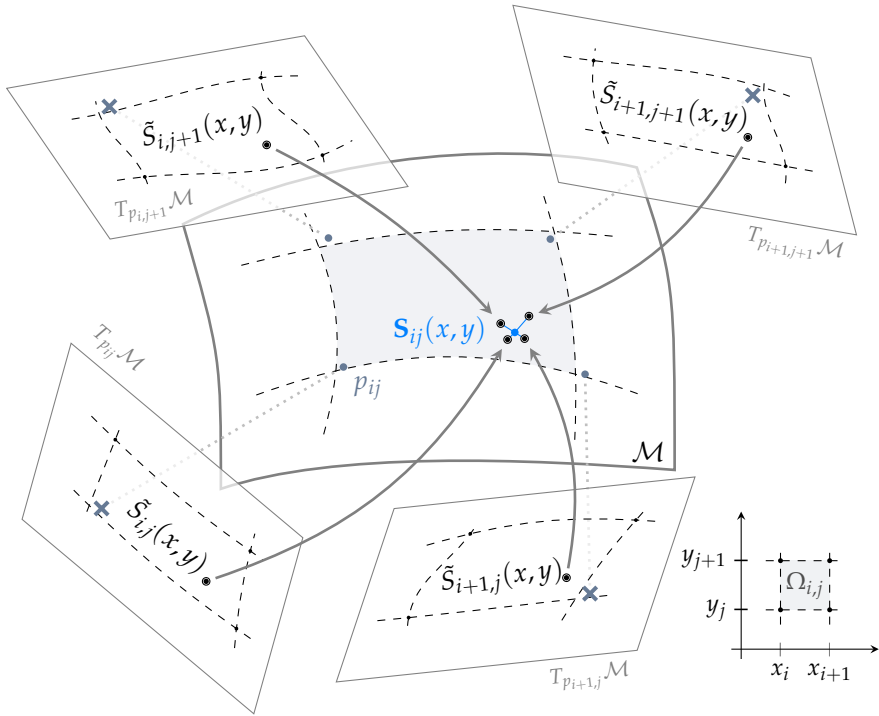


Fig. 7.3 The blended surface S is a weighted average of four local surfaces constructed at linearization points p_{ij} , $p_{i+1,j}$, $p_{i,j+1}$ and $p_{i+1,j+1}$.

This approach has the advantage to require only the evaluation of few Riemannian logarithms and exponentials, as the weighted averaging between two points reduces to a weighted geodesic (see Definition 2.53 and Equation (2.6)). The drawback of this is that a choice has to be made in the order of the variables along which the average is performed (see Figure 6.3 for an illustration of the impact of this choice on Bézier surfaces). In practice, the differences in the result are quite mild while the computational gain is important, so we will choose preferably this last solution. Note that, on symmetric spaces and for averages of four points, both lead to the same result.

Remark 7.8. In Definition 7.6, at most four surfaces S_{kl} are blended together at each point $(x, y) \in \Omega$. Consequently, the blending weights w_{kl} are only used on a support restricted to $[x_{k-1}, x_{k+1}] \times [y_{l-1}, y_{l+1}]$. This restriction could be relaxed such that, in each subdomain Ω_{ij} , one could per-

form a weighted average of more surfaces. For instance, one could blend all local surfaces S_{kl} associated with the corners (x_k, y_l) of the subdomains next to Ω_{ij} . In that case, the support of each weight function w_{kl} would become $[x_{k-2}, x_{k+2}] \times [y_{l-2}, y_{l+2}]$. Ultimately, all local surfaces could be blended together. However, this does not seem to have any advantage (except for one feature discussed in [AGW20]). On the contrary, this strategy comes along with two drawbacks. First, a weighted average involving more points will be computationally more expensive, and second, as the tangent spaces $T_{p_{kl}}\mathcal{M}$ associated to the points $(x_k, y_l) \notin \Omega_{ij}$ are worst representations of \mathcal{M} , surfaces S_{kl} are also worst representations of \mathbf{S} in Ω_{ij} . Therefore, they should not be considered in the blending mechanism.

Remark 7.9. In this chapter, we restrict ourselves to rectangular meshes of Ω . In fact, the blending method is not strictly limited to this framework. Indeed, any polygonal decomposition of Ω is acceptable; in that case, the linearization points are then associated with each vertex of the decomposition. However, such a generalization will come along with an important drawback: the use of dyadic averaging (av_{dyad}) and the choice of the blending weights become more complicated (for instance, such that the blended surface is differentiable). For this reason, we will continue to stick to rectangular grids.

We close the section with a summary of the computational complexity for surface evaluation.

Proposition 7.10 (Complexity of blended surface evaluation). *Evaluating a blended surface $\mathbf{S} : \Omega \rightarrow \mathcal{M}$ at a point $x \in \Omega$ requires at most 7 Riemannian exponentials and 3 Riemannian logarithms.*

Proof. At each point $(x, y) \in \Omega$ at most four local surfaces $S_{kl}(x, y) = \exp_{p_{kl}}(\tilde{S}_{kl}(x, y))$ have to be averaged. The evaluation of each surface involves one Riemannian exponential; the averaging av_{dyad} involves three dyadic averages, where each one requires one Riemannian logarithm and one Riemannian exponential. \square

Remark 7.11. Actually, Proposition 7.10 can be easily generalized to multivariate functions \mathbf{S} . Consider a d -dimensional domain $\Omega \subset \mathbb{R}^d$. Then, for each point $x \in \Omega$, at most 2^d local surfaces must be averaged. This implies first that 2^d Riemannian exponentials are required to evaluate each local surface S_{kl} . The dyadic averaging is now performed along each dimension at a time, such that one needs $\sum_{i=0}^{d-1} 2^i = 2^d - 1$ geodesic averages

(each of them requiring one exp and one log). As a summary, evaluating a d -dimensional blended function \mathbf{S} requires $2^{d+1} - 1$ Riemannian exponentials and $2^d - 1$ Riemannian logarithms. Note that this holds also for blended curves presented in Chapter 4.

7.2.2 Continuity and smoothness of blended surfaces

The continuity and the smoothness of the blended surface is analyzed in this section. The analysis is limited to C^1 -continuity, as in the previous chapters. However, the demonstration can easily be generalized to higher regularity conditions, which is one of the best advantages of this approach.

Theorem 7.12 (C^1 -continuity of blended surfaces). *Let \mathcal{M} be a smooth manifold and consider the blended surface from Definition 7.6. Further, let us abbreviate*

differentiability
conditions

$$D_{ij} = \max_{\substack{k, \bar{k} \in \{i, i+1\} \\ l, \bar{l} \in \{j, j+1\}}} d(p_{kl}, p_{\bar{k}\bar{l}}), \quad \text{and} \quad R_{ij} = \max_{\substack{k \in \{i, i+1\} \\ l \in \{j, j+1\}}} \|\tilde{S}_{kl}\|_{C^0(\Omega_{ij})}.$$

The blended surface \mathbf{S} exists, is unique, and is C^1 if

- (i) the tangent surfaces \tilde{S}_{ij} are C^1 on $[x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}] \cap \Omega$,
- (ii) the weight functions w_{ij} are C^1 on Ω and have a support on $[x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}] \cap \Omega$,
- (iii) $D_{kl} + 2R_{kl} < \frac{r_{kl}}{2}$ for all $k = 0, \dots, M$, $l = 0, \dots, N$, where r_{kl} is the injectivity radius of \mathcal{M} restricted to the ball $B_{D_{kl}+3R_{kl}}(p_{kl}) \subset \mathcal{M}$.

Proof. Let us first prove that \mathbf{S} exists and is unique (i.e., is “well-defined”). Let $(x, y) \in \Omega_{ij}$. By the triangle inequality, the maximum distance between the four points $S_{kl}(x, y)$ inside the average is no larger than $D_{ij} + 2R_{ij}$, which, by condition (iii), is smaller than the injectivity radius r_{ij} . Therefore, the inner dyadic averages of av_{dyad} , given by

$$y_1 = \exp_{S_{ij}(x, y)} \left(\frac{w_{i+1, j}(x, y)}{w_{i, j}(x, y) + w_{i+1, j}(x, y)} \log_{S_{ij}(x, y)}(S_{i+1, j}(x, y)) \right),$$

and

$$y_2 = \exp_{S_{i, j+1}(x, y)} \left(\frac{w_{i+1, j+1}(x, y)}{w_{i, j+1}(x, y) + w_{i+1, j+1}(x, y)} \log_{S_{i, j+1}(x, y)}(S_{i+1, j+1}(x, y)) \right),$$

Without loss of generality, let y_1 and y_2 be closer to $S_{i,j}$ and $S_{i,j+1}$ respectively.

are well-defined. By the triangle inequality, again, they also satisfy

$$\begin{aligned} d(y_1, y_2) &\leq \min_{k, \bar{k} \in \{i, i+1\}} \left(d(y_1, S_{kj}(x, y)) + d(S_{kj}(x, y), S_{\bar{k}, j+1}(x, y)) \right. \\ &\quad \left. + d(S_{\bar{k}, j+1}(x, y), y_2) \right) \\ &\leq 2(D_{ij} + 2R_{ij}), \end{aligned}$$

as $\min_k d(y_1, S_{kj}(x, y))$ and $\min_k d(S_{k, j+1}(x, y), y_2)$ are bounded by $\frac{1}{2}(D_{ij} + 2R_{ij})$. By condition (iii), $y_1, y_2 \in B_{D_{ij}+3R_{ij}}(p_{ij})$. This means that $d(y_1, y_2) \leq 2(D_{ij} + 2R_{ij}) < r_{ij}$. Therefore the outer dyadic average of av_{dyad} is well-defined, and thus $\mathbf{S}(x, y)$ is also well-defined.

C^1 continuity in $\overline{\Omega_{ij}}$ is direct: since the Riemannian logarithm and the exponential are smooth where they are well-defined, then the averaging operator av_{dyad} is also smooth where it is well-defined. By conditions (i) and (ii), the tangent surfaces and the weights used by the averaging operator av_{dyad} are smooth. We have already proven that av_{dyad} is well-defined, so av_{dyad} is smooth, and so is \mathbf{S} .

Finally, it remains to prove C^1 -continuity at the edges of all patches Ω_{ij} . Without loss of generality let us consider some point $(x, y_j) \in \overline{\Omega_{i, j-1}} \cap \overline{\Omega_{ij}}$ (the proof for points $(x_i, y) \in \overline{\Omega_{i-1, j}} \cap \overline{\Omega_{ij}}$ is analog). Denote by \mathbf{S}^- and \mathbf{S}^+ the (differentiable) restriction of \mathbf{S} to $\overline{\Omega_{i, j-1}}$ and $\overline{\Omega_{ij}}$, respectively. The x -derivatives coincide. Indeed, as $w_{i, j-1} = w_{i+1, j-1} = w_{i, j+1} = w_{i+1, j+1} = 0$, \mathbf{S}^- and \mathbf{S}^+ coincide on $[x_i, x_{i+1}] \times \{y_j\}$. We show that the y -derivatives coincide as follows. By condition (ii) and because the weights sum up to one (Definition 7.6), the y -derivative of all weights w_{kl} involved in the computation of \mathbf{S}^- and \mathbf{S}^+ is zero in (x, y_j) , that is,

$$\left. \frac{\partial w_{kl}(x, y)}{\partial y} \right|_{y=y_j} = 0 \text{ for } k \in \{i, i+1\}, l \in \{j-1, j+1\}.$$

Hence, while computing the y -derivative of \mathbf{S}^- (resp. \mathbf{S}^+) in (x, y_j) , all terms involving $\frac{\partial w_{kl}(x, y)}{\partial y}$ vanish. Shortening the notations as $w_{kl} := w_{kl}(x, y_j)$ and $S_{kl} := S_{kl}(x, y)$, one obtains, at (x, y_j) :

$$\begin{aligned} \mathbf{S}^-(x, y) &= \text{av} [y_1, y_2; w_{i, j-1} + w_{i+1, j-1}, w_{i, j} + w_{i+1, j}] \\ \mathbf{S}^+(x, y) &= \text{av} [y_3, y_4; w_{i, j} + w_{i+1, j}, w_{i, j+1} + w_{i+1, j+1}] \end{aligned}$$

where

$$\begin{aligned} y_1 &= \text{av} [S_{i,j-1}, S_{i+1,j-1}; w_{i,j-1}, w_{i+1,j-1}] \\ y_2 &= \text{av} [S_{i,j}, S_{i+1,j}; w_{i,j}, w_{i+1,j}] \\ y_3 &= \text{av} [S_{i,j}, S_{i+1,j}; w_{i,j}, w_{i+1,j}] \\ y_4 &= \text{av} [S_{i,j+1}, S_{i+1,j+1}; w_{i,j+1}, w_{i+1,j+1}]. \end{aligned}$$

As $w_{i,j-1} = w_{i+1,j-1} = w_{i,j+1} = w_{i+1,j+1} = 0$, we observe directly that $\mathbf{S}^-(x, y) = y_2 = y_3 = \mathbf{S}^+(x, y)$. Since \mathbf{S}^- and \mathbf{S}^+ coincide, they have the same y -derivative in (x, y_j) .

So \mathbf{S} is C^1 and well-defined on Ω . \square

Remark 7.13. At (x_i, y_j) all weight functions are equal to zero, except for $w_{ij}(x_i, y_j) = 1$. Furthermore, the C^1 conditions on the weights impose that all weight functions w_{ij} have zero derivative in (x_k, y_l) , for all k, l . This means that, in a neighborhood of (x_i, y_j) , the surface \mathbf{S} corresponds to S_{ij} at first order. It is easy to extend this property to k^{th} order regularity and correspondence. For this, one needs to impose that the weight functions and the tangent surfaces are C^k .

Remark 7.14. An analogous version of Theorem 7.12 also holds if av_{dyad} is replaced by a different smooth averaging operator such as av_{min} . The proof stays the same, but we must adapt condition (iii) as well as arguments to ensure that \mathbf{S} exists and is unique. Indeed, by [AGSW16b, proof of Prop. 12] the weighted average av_{min} is well-defined if the distance between the averaged points is smaller than $\rho_{kl} = \min\{r_{kl}, \frac{\pi}{4\Delta_{kl}}\}/3$, where r_{kl} is the local injectivity radius and $\Delta_{kl} \geq 0$ is an upper bound on the sectional curvatures of \mathcal{M} on $B_{D_{kl}+3R_{kl}}(p_{kl})$. Thus, in condition (iii) and in the proof, r_{kl} must be replaced by ρ_{kl} .

Remark 7.15. If the surface \mathbf{S} cannot be properly represented by local surfaces S_{kl} because condition (iii) is not met, then this problem can be resolved by appropriately shifting the linearization points and/or by sufficiently refining the grid $(x_k, y_l)_{kl}$ on the domain Ω . Indeed, this will reduce the size of the Ω_{kl} as well as the distance between the linearization points; therefore, the quantities D_{kl} and R_{kl} both decrease (and can be driven arbitrarily close to zero). Correspondingly, the local injectivity radii r_{kl} increase so that condition (iii) will be satisfied at some point. This goes beyond the scope of this thesis but some techniques are presented in [AGW20].

average

Note, though, that condition (iii) is a worst case estimate in which all tangent space surfaces \tilde{S}_{ij} are assumed to be fully independent. In data fitting applications, however, all \tilde{S}_{ij} would fit the same data shifted to the corresponding tangent space. Hence, one can expect the local surfaces S_{kl} , associated with a given patch Ω_{ij} , to be quite similar. The quantity $D_{kl} + 2R_{kl}$ from condition (iii) should then be replaced by the (much smaller) distance between the S_{kl} associated with Ω_{ij} .

7.2.3 Choice of the weight functions

Finally, we discuss the choice of weight functions w_{ij} used in Definition 7.6 and in Theorem 7.12. For now, only the support and the regularity of the weight functions are imposed, such that there remains a lot freedom for choosing them properly.

As in Chapter 4, one might try to employ weights that make the blended surface as smooth as possible. Ultimately, the fitting blended surface will be composed of tangent thin plate splines mapped back to \mathcal{M} . Hence, a natural choice of weights would be such that the resulting blended surface $\mathbf{S}(x, y)$ minimizes the thin plate spline energy (7.3) among all possible blended surfaces, when \mathcal{M} is a Euclidean space. However, this would make the weight functions strongly depend on the local surfaces S_{ij} and the manifold \mathcal{M} . As a result, computing them would require expensive optimization problems that we actually want to avoid.

Therefore, like in the case of curves, a compromise could be to compute weights that are optimal at least for the zeroth order approximation of the local surface S_{ij} , i.e., when S_{ij} are constant. However, while those optimal weights exist for curve fitting (see Remark 4.13), they do not exist in the multidimensional case.

Theorem 7.16 (Surface weights [AGW20]). *Let $\mathcal{M} = \mathbb{R}^r$. Consider MN patches $\Omega_{ij} \subset \Omega$, $M, N > 1$, and assume the local surfaces S_{ij} to be constant. Consider weights $w_{ij} : \Omega \rightarrow \mathbb{R}$ with support in $[x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}]$ and summing to one. There is no such weights w_{ij} which, independently of the S_{ij} , minimize the thin plate spline energy of the resulting blended surface $\mathbf{S}(x, y) = \sum_{i=0}^M \sum_{j=0}^N w_{ij}(x, y) S_{ij}$.*

Proof. A detailed proof can be found in [AGW20]. We just sketch it here. In substance, the problem is restricted (without loss of generality) to the case $\mathcal{M} = \mathbb{R}$, and the proof is made by contradiction: one assumes that there exists such w_{ij} and builds the following counter-example. Let $S_{ij} = 1$ for

all $j > j_0$ and $S_{ij} = 0$ otherwise. The corresponding blended surface \mathbf{S}^y necessarily satisfies

$$\begin{cases} \mathbf{S}^y(x, y) = 0 & \text{for } y < y_{j_0} \\ \mathbf{S}^y(x, y) = 1 & \text{for } y > y_{j_0+1}. \end{cases} \quad (7.6)$$

By hypothesis on the weights, \mathbf{S}^y minimizes the thin plate spline energy among all surfaces \mathbf{S} satisfying (7.6). That minimizer is explicitly known (it is constant in x -direction and is equal to the natural cubic spline in the y -direction):

$$\mathbf{S}^y(x, y) = \begin{cases} 0 & \text{if } y < y_{j_0}, \\ 1 & \text{if } y > y_{j_0+1}, \\ g_1(y) = g\left(\frac{y_{j_0+1}-y}{y_{j_0+1}-y_{j_0}}\right) & \text{otherwise,} \end{cases}$$

with $g(t) = 2t^3 - 3t^2 + 1$. Likewise, for $S_{ij} = 1$ if $i > i_0$ and $S_{ij} = 0$ otherwise, the corresponding blended surface is expressed as

$$\mathbf{S}^x(x, y) = \begin{cases} 0 & \text{if } x < x_{i_0}, \\ 1 & \text{if } x > x_{i_0+1}, \\ g_2(x) = g\left(\frac{x_{i_0+1}-x}{x_{i_0+1}-x_{i_0}}\right) & \text{otherwise.} \end{cases}$$

Finally, one considers the case where $S_{ij} = 2$ if $i > i_0$ and $j > j_0$, $S_{ij} = 0$ if $i \leq i_0$ and $j \leq j_0$, and $S_{ij} = 1$ else. In that case the blended surface is given by $\mathbf{S} = \mathbf{S}^y + \mathbf{S}^x$.

By hypothesis in the beginning of the proof, there shouldn't exist any surface \mathbf{S}^* with smaller thin plate spline energy than \mathbf{S} . Therefore, for any smooth test function $\phi : \Omega \rightarrow \mathbb{R}$ with compact support in the interior of $\Omega_{i_0-1, j_0} \cup \Omega_{i_0, j_0}$ one has

$$E[\mathbf{S} + \alpha\phi] \geq E[\mathbf{S}]$$

as long as $\alpha \in \mathbb{R}$ has small enough modulus. However, it can be shown that $\frac{d}{d\alpha} E[\mathbf{S} + \alpha\phi]_{\alpha=0} \neq 0$, which contradicts $E[\mathbf{S} + \alpha\phi] \geq E[\mathbf{S}]$ so that the weights cannot have been optimal (see [AGW20] for a detailed development). \square

Since there is no optimal weights independent from the local surfaces, the weight functions will be chosen as tensorization of the weights used

for curve fitting:

$$w_{ij}(x, y) = \begin{cases} g\left(\frac{x-x_i}{x_{i+1}-x_i}\right) g\left(\frac{y-y_j}{y_{j+1}-y_j}\right) & \text{if } x \in [x_i, x_{i+1}] \times [y_j, y_{j+1}], \\ g\left(\frac{x-x_i}{x_{i+1}-x_i}\right) g\left(\frac{y-y_j}{y_{j-1}-y_j}\right) & \text{if } x \in [x_i, x_{i+1}] \times [y_{j-1}, y_j], \\ g\left(\frac{x-x_i}{x_{i-1}-x_i}\right) g\left(\frac{y-y_j}{y_{j+1}-y_j}\right) & \text{if } x \in [x_{i-1}, x_i] \times [y_j, y_{j+1}], \\ g\left(\frac{x-x_i}{x_{i-1}-x_i}\right) g\left(\frac{y-y_j}{y_{j-1}-y_j}\right) & \text{if } x \in [x_{i-1}, x_i] \times [y_{j-1}, y_j] \end{cases} \quad (7.7)$$

with $g(t) = 2t^3 - 3t^2 + 1$.

Remark 7.17. This choice of weights will at least lead to optimal blending along the coordinate directions; also the tensorized weights match our use of the tensorized average av_{dyad} in Definition 7.6. In fact, in data fitting applications the differences between the different local surfaces S_{ij} will be rather small (at least for neighboring indices) so that the choice of the blending weights is not expected to have too much influence anyway. This expectation is illustrated and confirmed in Section 7.3 (Figure 7.7).

Remark 7.18. Note that equation (7.7) gives a weight associated to each corner of the patch Ω_{ij} . Renaming locally $p_{ij} =: \hat{p}_{10}$, $p_{i+1,j} =: \hat{p}_{11}$, $p_{i,j+1} =: \hat{p}_{01}$ and $p_{i+1,j+1} =: \hat{p}_{11}$, and their respective associated coordinates by $\hat{x}_{mn}, \hat{y}_{mn}$, $m, n \in \{0, 1\}$, then the weight associated to each of them is now

$$\hat{w}_{mn}(x, y) = g\left(\frac{|x - \hat{x}_{mn}|}{\hat{x}_{1n} - \hat{x}_{0n}}\right) g\left(\frac{|y - \hat{y}_{mn}|}{\hat{y}_{m1} - \hat{y}_{m0}}\right), \quad \text{for } m, n \in \{0, 1\}. \quad (7.8)$$

7.2.4 Fitting with blended surfaces

Consider now the data points $d_0, \dots, d_n \in \mathcal{M}$ corresponding to parameter values $t_k = (x_k, y_k)$, $k = 0, \dots, n$, and the regularization parameter $\lambda > 0$. The domain $\Omega \subset \mathbb{R}^2$ is divided in patches Ω_{ij} defined by a regular grid of nodes $(x_i, y_j) \in \Omega$, to which linearization points $p_{ij} \in \mathcal{M}$ are associated, $i = 0, \dots, M, j = 0, \dots, M$. This section gives a proper (algorithmic) definition to the fitting blended surface; then the six properties mentioned in the beginning of the chapter are proved.

Definition 7.19 (*Fitting blended surface on manifolds*). Let \mathcal{M} be a smooth manifold. The blended surface $\mathbf{S} : \Omega \rightarrow \mathcal{M}$ fitting the data points $d_k \in \mathcal{M}$ at $t_k \in \Omega$ is defined as a three steps procedure:

fitting with a
blended
surface

1. Map all the data points d_k to the tangent spaces $T_{p_{ij}}\mathcal{M}$ of the linearization points p_{ij} via the Riemannian logarithm;
2. Compute the tangent thin plate spline $\tilde{S}_{ij}(x, y) \in T_{p_{ij}}\mathcal{M}$ and the associated local surface $S_{ij}(x, y) \in \mathcal{M}$;
3. Blend the surfaces as a $\mathbf{S}(x, y)$ via Definition 7.6, with weights given by (7.8).

Theorem 7.20. *Let \mathcal{M} be a smooth manifold. Let d_k be data points on \mathcal{M} and p_{ij} be linearization points on \mathcal{M} satisfying the conditions of Theorem 7.12. The fitting blended surface $\mathbf{S}(t) : \Omega \rightarrow \mathcal{M}$ of Definition 7.19 satisfies the following properties:*

differentiability conditions

- (i) $\mathbf{S}(x_k, y_k) \rightarrow d_k$ when $\lambda \rightarrow \infty$, for $k = 0, \dots, n$;
- (ii) $\mathbf{S}(x, y)$ is well defined and C^1 ;
- (iii) when \mathcal{M} is a Euclidean space, $\mathbf{S}(x, y)$ is the thin plate spline that minimizes (7.1).

thin plate spline

Proof. Property (iii) is a direct generalization of Lemma 4.17 for thin plate splines. Indeed, as all tangent spaces are Euclidean, then all local surfaces S_{ij} are the same thin plate spline, and so is \mathbf{S} .

Property (ii) is ensured by the hypothesis of Theorem 7.12, as the weight functions used in Definition 7.19 are smooth, well-defined, and satisfy condition (ii).

Finally, we show property (i) as follows. Consider a tangent surface $\tilde{S}_{ij}(x, y) \in T_{p_{ij}}\mathcal{M}$ fitting the lifted data points $\tilde{d}_k \in T_{p_{ij}}\mathcal{M}$. By Definition 7.3, $\tilde{S}_{ij}(x_k, y_k) \rightarrow \tilde{d}_k$ while $\lambda \rightarrow \infty$. Hence, by property (iii) of Theorem 7.12, $S_{ij}(x_k, y_k) \rightarrow d_k$, for k such that $(x_k, y_k) \in [x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}]$, and thus $\mathbf{S}(x_k, y_k) \rightarrow d_k$. □

The Figure 7.4 illustrates numerically the properties (i) and (ii) of Theorem 7.20. In that example, data points (red dots) are spread over a domain $\Omega = [0, 1]^2$ and the blended surface is composed of four patches. The root points are data points as well. They are represented as green circles.

Property (iv) of the introduction is of course validated, as Definition 7.19 makes only use of the Riemannian exponential and logarithm. The two following proposition show that properties (v) and (vi) are met as well.

Proposition 7.21 (Minimal representation of the surface). *The blended surface of Definition 7.19 is uniquely represented by $\mathcal{O}(MNn)$ tangent vectors.*

minimal representation

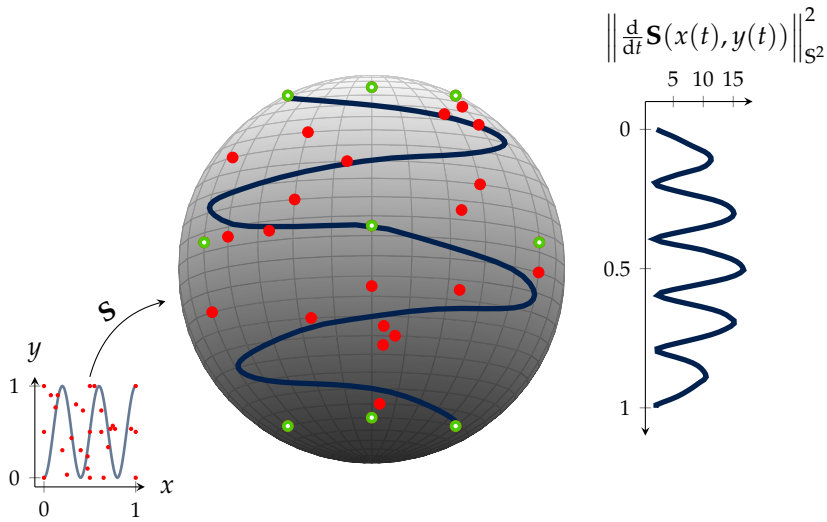


Fig. 7.4 The different patches are glued together C^1 continuously. Data points are represented as red dots and spread over a domain $\Omega = [0, 1] \times [0, 1]$. The four patches are chosen as a regular grid on Ω with $x_i, y_j \in \{0, 0.5, 1\}$. The linearization points of each patch are the data points whose coordinate is the closest to (x_i, y_j) . To illustrate the result, we drew a parameterized curve $\gamma(t) = (x(t), y(t))$ on Ω and then displayed the resulting curve $\mathbf{S}(\gamma(t)) = \mathbf{S}(x(t), y(t))$. The normed velocity of the curve is then displayed as a visual example of the differentiability of the method.

Proof. Each tangent surface is represented by the $n + 3$ tangent coefficients a and δ from Definition 7.3. One has to store MN tangent surfaces (at each linearization points). □

Proposition 7.22 (Exponential and logarithm maps required). *The number of exponentials and logarithms required by the blended surface of Definition 7.19 is*

- MNn logarithms for the construction of the minimal representation of the curve of Proposition 7.21
- 7 exponential maps and 3 logarithm map to reconstruct the surface $\mathbf{S}(x, y)$, for a given parameter value (x, y) , given that minimal representation.

Proof. To compute each tangent surface, one needs first to lift the n data point d_k to the MN tangent spaces $T_{p_{ij}}\mathcal{M}$. This proves (i). The proof of (ii) is direct from Proposition 7.10. □

7.3 Illustrative examples

This last section is dedicated to illustrative examples computed to validate numerically the theory. The examples are done on the sphere S^2 because they are easy to represent.

As first example, Figure 7.5 shows a parameterized path $\mathbf{S}(x(t), y(t))$ followed on a blended surface \mathbf{S} reconstructed based on four local surfaces S_{ij} , $i, j \in \{0, 1\}$. The path $(x(t), y(t)) \in \Omega$ is given by

$$(x(t), y(t)) = (t, 0.5 + 0.5 \cos(5\pi t + \pi)).$$

The surface interpolates ($\lambda \rightarrow \infty$) 20 data pairs $(d_k, t_k) \in \mathcal{M} \times [0, 1]^2$, where the points d_k are represented as red dots. The linearization points are part of the data set and are chosen as (p_{ij}, t_{ij}) , with $t_{ij} = (x_i, y_j) = (i, j)$. Each local surface S_{ij} is represented as well as the resulting blended surface. Visually, one can observe that the surface looks smooth as expected. One can also observe that the local surface S_{ij} is closer and closed to \mathbf{S} as soon as (x, y) gets closer to (x_i, y_j) . This is visible on the displayed sphere, but also on the different error curves (*i.e.*, the distance between the local surface \tilde{S}_{ij} and the blended surface \mathbf{S}) represented next to each linearization point: this is particularly visible for the local surface corresponding to the top-right linearization point $S_{01}(x, y)$. We also observe that, here, the choice of linearization points could be refined: indeed the local surfaces are quite different from each other; the error curves show absolute distance of approximately $1e-2$. This suggests that the patch Ω is too large to represent well \mathbf{S} based on the four tangent spaces at each corner.

By refining the domain in x and y direction, one obtains error curves displayed in Figure 7.6. In this example, the domain $\Omega = [0, 1]^2$ is divided in four sub-domains $\Omega_{ij} = [x_i, y_j] \times [x_{i+1}, y_{j+1}]$, $i, j \in \{0, 1\}$, for $x_i, y_j \in \{0, 0.5, 1\}$. The blended surface $\mathbf{S}(x, y)$ is thus composed of nine blended surfaces and is represented in Figure 7.4 (see Section 7.2.4). As in Figure 7.5, this example only displays a parameterized curve $\mathbf{S}(x(t), y(t)) \in S^2$ on \mathbf{S} . Each plot (denote them by P_{ij}) displayed in Figure 7.6 represent the distance from the local surface $S_{ij}(x(t), y(t))$ to the blended surface $\mathbf{S}(x(t), y(t))$. The spacial disposition of each plot P_{ij} corresponds to the spacial localization of its associated linearization points p_{ij} displayed on the sphere of Figure 7.4. In each plot P_{ij} , the shaded areas represent the values of t for which $(x(t), y(t))$ is in the patch where the S_{ij} is used in the

7 | Fitting with blended surfaces

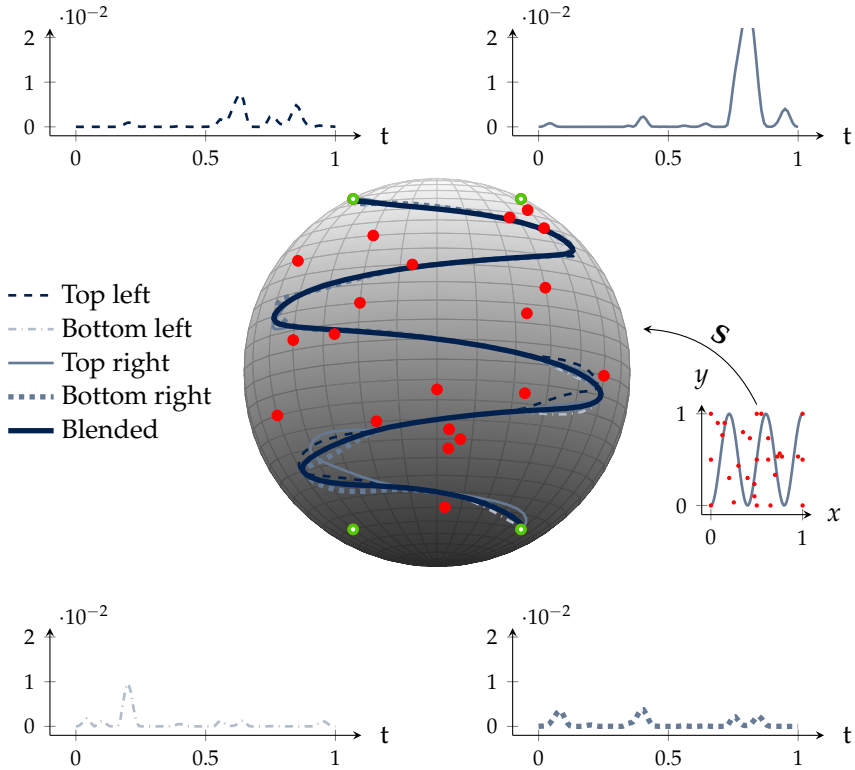


Fig. 7.5 The different local surfaces differ from each other; the blended surface gives more importance to a given local surface as soon as it goes closer to its linearization point.

blending technique. For instance, the local surface $S_{11}(x, y)$, associated to the linearization point p_{ij} at $(x_i, y_i) = (0.5, 0.5)$ is used in the four patches. However, the surface $S_{00}(x, y)$ is only blended on Ω_{00} (top left patch). We can observe that the distance to S is drastically reduced compared to Figure 7.5.

Actually, as mentioned in Remark 7.15, refining the grid is often a solution to meet all the properties of Theorem 7.12, for instance, when the local injectivity radius is small. Solutions for refinement are discussed in [AGW20], and go beyond the scope of this thesis.

Finally, Figure 7.7 illustrates the Remark 7.17. Here, only two patches are represented: $\Omega_{00} = [0, 0.5] \times [0, 1]$ and $\Omega_{10} = [0.5, 1] \times [0, 1]$. The blended surface S is represented as in the previous examples, *i.e.*, by repre-

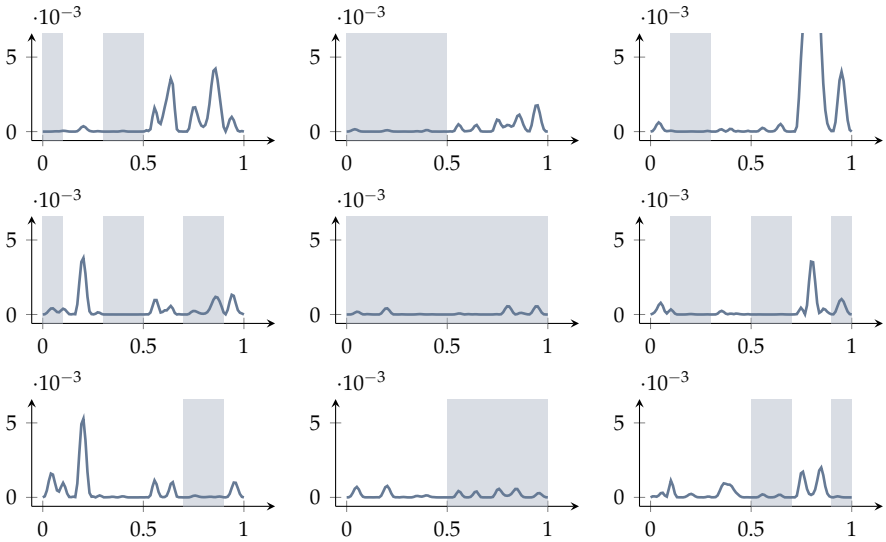


Fig. 7.6 The blended surface follows the local curves in the dedicated patches. The differences between $\mathbf{S}(x(t), y(t))$ and $S_{ij}(x(t), y(t))$ is indeed low on $(x(t), y(t)) \in [x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}]$, represented by the shaded areas.

senting just a the parameterized curve $\mathbf{S}(x(t), y(t))$. In this example, four weight functions are considered: (i) the cubic weights proposed in (7.8) ($g(t) = 2t^3 - 3t^2 + 1$), (ii) a linear weight function ($g(t) = 1 - t$), (iii) a piecewise constant weight function ($g(t) = 1$ for $t < 0.5$ and $g(t) = 0$ for $t \geq 0.5$) and (iv) a cosine weight function ($g(t) = \frac{\cos(\pi t)+1}{2}$).

This example confirms Remark 7.17: indeed, the distance between the resulting blended surfaces \mathbf{S}_k , for k corresponding to one of the four possible cases cited above, is small (of the order of maximum $1e-6$, except for the piecewise constant weights, where distances of $1e-3$ are measured). It also illustrates that discontinuous weights lead to lack of differentiability (for the piecewise constant weights).

weight
function

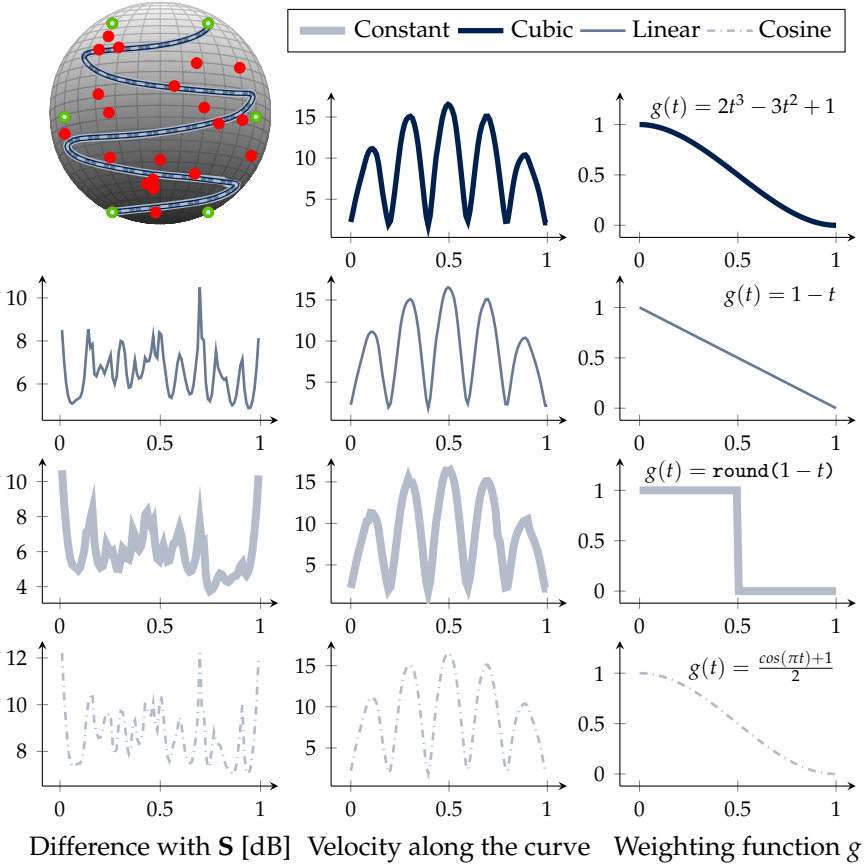


Fig. 7.7 The weight function has very little influence on the quality of the fitting. Even non-differentiable weights do not lead to visually bad solutions (except for the constant solution).

This section is partially based on the project paper [AGW20], sometimes cited verbatim. This paper is a work in progress, so its current reference is:


[AGW20] P.-A. Absil, Pierre-Yves Gouenbourger, and Benedikt Wirth. Smooth surface fitting in Riemannian manifolds using patch-wise linearization. In progress, 2020

The figures can be reproduced based on the code provided in the toolbox available at this link address:

<https://github.com/pgouenbourger/manint>

8

Summary and perspectives

 YOU MADE IT up to here and you read it all? Congratulations! If not, this is what conclusions are made for. The context and the objectives of the thesis will be summarized, as well as the solutions and contributions proposed throughout this document. Most importantly, a global comparison of the different fitting methods is presented in Tables 8.1 and 8.2. And as research has no limit, this final chapter ends with some new research perspectives.

8.1 What was it about?

Data structure has always been the curse of optimization as it comes with equality constraints. Applications in many fields must deal with that, like in computer vision [Fle13], in imaging [BLPS19], or to represent fixed rank matrices as submanifold of all matrices [Van13, Mas19].

Among data structured problems, the fitting problem arises also in many different applications: medical imaging [BRP12], computer graphics [Par10], model reduction [PA16], natural behavior understanding [SKKS14], etc. Those applications involve very various manifolds. Different approaches have been proposed to solve the fitting problem but to our knowledge, most of them were either manifold-oriented or implied time-consuming operations [SASK12, Dyn09, MM17, KDL18].

Fitting data points $d_i \in \mathcal{M}$ associated with parameters $t_i \in \mathbb{R}^d$, $i = 0, \dots, n$, is usually a sub-step of big applications, such that it is preferable to reduce as much as possible the computation time allocated to this task. A simple approach is to get rid of the structure and just fit the data with basic algorithms. This is a good strategy when the manifold is locally flat. When the space has higher curvature, more dedicated (yet fast and manifold-ready) algorithms might be welcome.

This thesis proposes methods that are fast and reliable, easily applicable to any Riemannian manifold for which at least the exponential map and the logarithm map (or good approximations) are available. The strategy is to approach the solution of the optimization problem (1.1) by simultaneously satisfying the following six properties:

- (i) As $\lambda \rightarrow \infty$, the data points are interpolated at the given times;
- (ii) The curve is of class C^1 ;
- (iii) If the manifold \mathcal{M} reduces to a Euclidean space, then the produced curve minimizes (1.1), *i.e.*, it is the natural smoothing spline ($d = 1$) or the fitting thin plate spline ($d = 2$);
- (iv) The only knowledge that the methods require from the manifold is the Riemannian exponential and the Riemannian logarithm;
- (v) The produced curve is represented by $\mathcal{O}(n)$ tangent vectors to the manifold (or simply points on the manifold), where n is the number of data points;
- (vi) Computing $\gamma(t)$ for any given t requires $\mathcal{O}(1)$ exp and log operations once the representation by $\mathcal{O}(n)$ tangent vectors is available.

The outcome consists in (i) different algorithms for fitting and interpolation of structured data $d_i \in \mathcal{M}$ corresponding to positions on of multidimensional domains $\Omega \in \mathbb{R}^d$ (more specifically $d = 1$ and $d = 2$), (ii) an analysis of the pertinence of those algorithms, and (iii) a toolbox with ready-to-run functions based on Manopt [BMAS14]. In addition to that, a recursive expression of the gradient of the objective function (1.1) in the search space of Bézier curves was provided in Chapter 5.

The toolbox is available at this link address:

<https://github.com/pgousenbourg/manint>

8.2 How well were the objectives achieved?

Different methods were presented in this thesis. Four of them are most important, because they permit to tackle one of the four cases presented in the introduction: curve interpolation (Chapter 3, with Bézier curves), curve fitting (Chapter 4, with blended curves), surface interpolation (Chapter 6, with Bézier surfaces), and surface fitting (Chapter 7, with blended surfaces). Each method was described and analyzed in the dedicated chapter in order to highlight its strengths and weaknesses; they were sometimes compared with the (more naive) methods used nowadays in applications like the wind field problem, PMOR or TVUS. In Chapter 5, a first numerical validation was performed on Bézier curves for interpolation and fitting. This permitted to validate that the Bézier approach provided acceptable results for data points not too spread out on the manifold.

Most of the presented methods satisfy the six target properties. A summary is given in Table 8.1. This table also recalls what were the limitations that motivated some other methods. For instance, the geodesic approach (for $d = 1$ or $d = 2$) is always non-differentiable, which motivates all the methods proposed in this thesis; the composite Bézier curves cannot always ensure interpolation when the approach is generalized to fitting, which motivated the conception of the blending technique; the generation of Bézier surfaces was quite complicated and had many chances to face the same interpolatory limitations than its 1D-analog, so blended surfaces were produced.

The following methods are compared:

- Cubic Bézier curves for interpolation, in Definition 3.20;
- Cubic Bézier curves generalized for fitting, in Definition 4.1;
- Blended curves, in Definition 4.12;
- The two Bézier-like approaches, from Section 4.3.2;
- Cubic Bézier surfaces of type I, II and III for interpolation, in Definition 6.16 (type I), Definition 6.17 (type II), and Definition 6.9 (type III);
- Blended surfaces, in Definition 7.19.

As point of comparison, we also list the most current methods usually used in the literature (when the data structure is taken into account, though!):

- A local curve (*i.e.*, a cubic spline computed on one tangent space, and then mapped back to the manifold, see the curves named **LC** in Section 4.6.3);
- A piecewise-geodesic curve;
- A local surface S_{ij} (see Definition 7.5);
- A piecewise geodesic surface.

There are two main advantages to the main four methods of this thesis compared to the classical approaches done in the literature. The first one is that the methods are differentiable. This is not the case of piecewise linear or piecewise geodesic approaches. The second one is that the methods are always local, such that each piece of the resulting curve or surface comes from a tangent space that approximates well the manifold in a certain neighborhood. This is not the case for the local curves (LC), or local surfaces (S_{ij}).

All methods are based on simple computation. They all meet the properties (iv–vi), *i.e.*, to only require the exponential and logarithm maps of the manifold (iv), to only need $\mathcal{O}(n)$ tangent vectors to represent the fitting curve (v), and to only need $\mathcal{O}(1)$ operations to reconstruct it (vi).

The complexity of the methods is nearly equivalent with an obvious advantage to the local versions (LC and S_{ij}) and piecewise geodesic (see Table 8.2, columns “offline” and “online”). All proportions kept, the blended curves and surfaces should be promoted, since the online phase (*i.e.*, the reconstruction of the curves and surfaces) is faster than in the Bézier case. Furthermore, the blending technique is more general since it includes both fitting and interpolation. A (limited) drawback is maybe the higher storage space needed (see Table 8.2, column “storage”). Anyways, the choice of the method depends, at the end of the day, on the application.

	Interp. for $\lambda \rightarrow \infty$	Natural spline/ Differentiability	Only Exps and Logs on \mathbb{R}^m	Storage: $\mathcal{O}(n)$ vectors	Reconstr.: $\mathcal{O}(1)$ operations	
	(i)	(ii)	(iii)	(iv)	(v)	(vi)
Bézier curve (interpolation)	✓	✓	✓	✓	✓	✓
Bézier curve (fitting)	✗/✓	✓/✗	✓	✓	✓	✓
Blended curve	✓	✓	✓	✓	✓	✓
Bézier-like curve	✓	✓	✗	✓	✓	✓
Piecewise geodesic curve	✓	✗	✗	✓	✓	✓
Local curve (LC)	✓	✓	✓	✓	✓	✓
Bézier surface I	✓	✓	✓	✗	✓	✓
Bézier surface II	✓	✓	✓	✓	✓	✓
Bézier surface III	✓	✗	✓	✓/✗	✓	✓
Blended surface	✓	✓	✓	✓	✓	✓
Piecewise geodesic surface	✓	✗	✗	✓/✗	✓	✓
Local surface (S_{ij})	✓	✓	✓	✓	✓	✓

Table 8.1 Blending and local techniques are the only methods that gather all properties. However, the local techniques strongly depend on the chosen reference point. For interpolation, the Bézier surface of type II is also a good choice but Table 8.2 exhibits its limitations at the reconstruction step. Note that, sometimes, it is not possible to give a binary answer: the fitting Bézier curve fills property (ii) and not property (i) when the remedy of Definition 4.7 is applied (and reversely when it is naively generalized). Similarly, Bézier surfaces of type III and geodesic surfaces need only Exps and Logs if the dyadic average is used; if not, then one has to solve the weighted Karcher mean, as in Bézier surfaces of type I.

	Representation (offline)		Reconstruction (online)		Tangent vectors required (storage)		Remarks
	# Logs	# Exps	# Logs	# Exps			
Bézier curve	0	$n^2 - n$	10	6		$2n$	
Blended curve	0	$n^2 - n$	3	1	6n - 4		With Bézier splines
Bézier-like curve I	0	$n^2 - n$	8	4		$2n$	
Bézier-like curve II	0	$n^2 - n$	5	3		$2n$	
Piecewise geodesic curve	0	0	1	1		0	
Local curve (LC)	0	$n^2 - n$	1	0		$2n$	With Bézier splines
Bézier surface I	0	$n^2 - n$	<i>nonlinear opti.</i>			$3n$	With generation of Sect. 6.5.
Bézier surface II	0	$n^2 - n$	46	30		$3n$	the generation of Sect. 6.4 needs $3n$ parallel transports.
Bézier surface III	0	$n^2 - n$	58	42		$3n$	
Blended surface	0	$n^2 - n$	7	3	N(n + 3)		With N thin plate splines
Piecewise geodesic surface	0	0	3	3		0	
Local surface (S_{ij})	0	$n^2 - n$	1	0		n	With one thin plate spline

Table 8.2 The number of exponentials and logarithms required by each method, for n data points, is a good indicator of its complexity. The blending algorithms show quite good reconstruction performances compared to their competitors; in addition they are better suited for data points spread over the domain. The other side of the coin is that more points must be stored in memory and more time is needed to generate the fitting function. However, as this task can be done offline, this drawback is mild. Note that the reconstruction of Bézier surfaces of type I cannot be measured on the same metric, as it requires to evaluate the weighted Karcher mean of sixteen points.

8.3 What to do next?

Different categories of further work may be considered. I would like to present here three of them: some extensions of the desirable properties that guided the whole thesis, some analytic perspectives, and finally some algorithmic considerations.

Among the extensions available for this work, one of them concerns property (iii). A consequence of this property is that, if the manifold \mathcal{M} reduces to a Euclidean space and $\lambda \rightarrow 0$, then the (Euclidean) solution curve γ converges to the least-squares linear regression solution. One could then expect that, on a general manifold \mathcal{M} , the solution curve obtained when $\lambda \rightarrow 0$ would converge to the least-squares geodesic regression of the data points. However, up to now, none of the curve fitting methods presented in this work satisfy this stronger version of the property. To be convinced of this, let us just consider a local curve computed on a given tangent space $T_x\mathcal{M}$, $x \in \mathcal{M}$. The regression curve γ computed on this tangent space will not be mapped to a geodesic on \mathcal{M} unless γ passes through x . As the reference points are always chosen arbitrarily, this case will nearly never happen. How to design an algorithm that satisfies this stronger property along with properties (i)–(vi) remains thus an open problem.

Another extension concerns the smoothness of the curves and surfaces. For now, Property (ii) only requires differentiability. However, one could imagine techniques and applications where k -smoothness could be required. With the blending technique, it does not seem impossible to achieve since the smoothness of the methods only depends on the smoothness of the local curves (resp. surfaces) and on the weight functions. This might be a small supplementary step to do, at low cost.

Then come considerations about the analysis of the fitting curves, and more specifically about the quality of the returned curve. In Chapter 5, the Bézier fitting curve of Definition 4.1 was compared to the solution obtained by optimizing an approximation of (1.1). As a result, one could confirm that the method was better when the points were not too far away from each other. However, this is not an actual bound on how far the solution departs from the optimum of (1.1). This is left to differential geometers. Note that those considerations are already taken into account in a paper in progress [AGW20].

Along with this analysis comes a related question: can we quantify the

influence of the chosen reference point? We observed in Figure 4.10b that the blending method could sometimes behave in an unexpected way due to the choice of the reference points $d_{\text{ref},1}$ and $d_{\text{ref},2}$ (for surfaces, the linearization points p_{ij}). How to optimally *and* efficiently choose these two points remains for now an open question which will be treated in [AGW20].

A third direction for further work concerns algorithmic perspectives and improvements to accelerate the different methods. First, in Chapter 5, the gradient descent algorithm requires the computation of the Jacobi fields. To accelerate the optimization, an interesting future research is to find out whether an approximate evaluation of the Jacobi fields (for instance, by approximately solving an ODE) suffices for convergence of the presented approach.

In all the other algorithms, we see that the reconstruction of the curve is generally fast. On the other hand, this comes with the drawback that the curve must be stored, and that the generation time may become high; furthermore, if a data point is added to the dataset afterwards, the whole curve must be reevaluated. In that last case, the cost of the generation is way too high compared to the added information. Hence, two open questions arise:

- How to accelerate the generation of the curves and surfaces?
- How to efficiently modify the existing representation of the curve by adding one data point to the data set, without re-generating the entire curve?

For instance, the generation of the local curves and surfaces could be made only on most local points. One could imagine that only the closest points will strongly alter the curve locally. The open question is to know how to choose the best “local” data set to generate the curve in an efficient and accurate way. This approach could help to answer to the two aforementioned questions.

Another bottleneck of the methods is the very first operation performed: the lifting of the data points to the tangent spaces with the logarithm map. We see in Table 8.2 that this step is the only cost of the generation. Lifting efficiently the data points would also help to reduce the generation time. But the lifting of the points is also important in the following case. Let us consider the case where the manifold is a circle of radius one, embedded in \mathbb{R}^2 , in the curve fitting case. On that circle, the reference point is $d = \theta$, represented here only with the angle to the center. Consider a set

of data points where $d_0 = d$, $d_1 = \theta + \pi$ and $d_2 = d$. The associated times are $t_i = i$, $i = 0, 1, 2$. The lifting of the data points in the tangent space of d will give $\log_d(d_0) = 0 = \log_d(d_2)$ and $\log_d(d_1) = \pi$. However, for the curve fitting problem, and given the time-dependence of the points, one could expect $\log_d(d_2)$ to be equal to 2π , as a natural way to reach that point *from* d_0 , *through* d_1 would be to continue the route around the circle, and not turn back after reaching d_1 . Different methods are under investigation in [AGW20]. However, if all of them intend in solving this problem and should lead to more accurate solutions, none of them reduces very efficiently the complexity of the generation algorithm. Doing both (reducing the complexity and increasing the accuracy) is still an open problem.

Finally, it should be noted that there is no clear indicator that would globally promote one specific method compared to another. The choice of the method, indeed, depends on the application. In those presented in this thesis, the interest of the Riemannian approach remains questionable, because the quality improvement is relatively small compared to the increased computational cost. For instance, in the PMOR problem (Section 4.6), we could see that the blended solution was the best choice in terms of any reasonable global measure of quality (Figure 4.17a), but that the time needed to obtain one solution doubled compared to any other approach. Therefore, our methods should be stressed on various applications in order to reach a verdict of their exclusive interest and performance. We may find applications where one of the proposed methods would boost the fitting quality with a slight additional computational cost, *e.g.*, negligible compared to the acquisition time of the data points.

Those propositions are left for further research. I do not claim that they will all have a positive outcome, just that there remains an important part of unknown at the end of this work. That is the beauty of research: there is always something new to discover.

Appendices

A Coefficients for interpolation with a composite cubic curve

The problem (3.29) on $\mathcal{M} = \mathbb{R}^m$ is a quadratic function to be optimized with respect to the $n + 1$ optimization variables $X = (b_0^+, b_1^-, b_2^-, \dots, b_n^-)$. The solution of that problem reduces to m independent linear systems $A \cdot X_k = C \cdot D_k$, where X_k is the vector of the k^{th} component of the points of X , and D_k is the vector of the k^{th} component of the data points $(d_i)_{i=0}^n$. We obtain

$$\int_i^{i+1} \|\ddot{\beta}_3(t - i; d_i, b_i^+, b_{i+1}^-, d_{i+1})\|_2^2 dt = 12(-3d_i b_i^+ + 3b_i^+ b_i^+ - 3b_i^+ b_{i+1}^- + 3b_{i+1}^- b_{i+1}^- - 3b_{i+1}^- d_{i+1} + K),$$

with $i = \lfloor t \rfloor$. K gathers the terms that are independent from the optimization variables. Introducing the differentiability constraints (3.18) $b_i^+ = 2p_i - b_i^-$, and β_3^i , the i^{th} segment of the composite cubic Bézier curve, one has

$$\frac{\partial \beta_3^0}{\partial b_0^+} = 6b_0^+ - 3b_1^- - 3d_0 \tag{A.1}$$

$$\frac{\partial \beta_3^0}{\partial b_1^-} = -3b_0^+ + 6b_1^- - 3d_1 \tag{A.2}$$

and for $i = 1, \dots, n-1, n \geq 2$

$$\frac{\partial \beta_3^i}{\partial b_i^-} = 6b_i^- + 3b_{i+1}^- - 9d_i \quad (\text{A.3})$$

$$\frac{\partial \beta_3^i}{\partial b_{i+1}^-} = 3b_i^- + 6b_{i+1}^- - 6d_i - 3d_{i+1}. \quad (\text{A.4})$$

By definition 3.15, (3.29) is minimized when these quantities vanish, which yields the linear system $A \cdot X = C \cdot D$, for $n \geq 2$ where (in matlab indexing)

$$\begin{aligned} A(1, 1 : 2) &= [6 \quad -3] \\ A(2, 1 : 3) &= [-3 \quad 12 \quad 3] \\ A(k, k-1 : k+1) &= [3 \quad 12 \quad 3] \quad k = 3, \dots, n \\ A(n+1, n : n+1) &= [3 \quad 6]. \end{aligned}$$

and

$$\begin{aligned} C(1, 1) &= 3 \\ C(2, 2) &= 12 \\ C(k, k-1 : k) &= [6 \quad 12] \quad k = 3, \dots, n \\ C(n+1, n : n+1) &= [6 \quad 3]. \end{aligned}$$

The third lines in the definition of A and C only hold for $n > 2$. All the other entries are equal to zero.

B Coefficients for interpolation with a hybrid composite curve

The hybrid composite Bézier curve (3.21) on $\mathcal{M} = \mathbb{R}^m$ requires to solve m independent linear systems in the variables $X = (b_1^-, b_2^-, \dots, b_{n-1}^+)$. The systems read $A \cdot X_k = C \cdot D_k$, where X_k is the vector of the k^{th} component of the points of X , and D_k is the vector of the k^{th} component of the data points $(d_i)_{i=0}^n$. The values of the entries of the A and C matrices are, in Matlab indexing

$$\begin{aligned} A(1, 1 : 2) &= [64 \quad 24] \\ A(2, 1 : 3) &= [24 \quad 144 \quad 36] \\ A(k, k - 1 : k + 1) &= [36 \quad 144 \quad 36] \quad k = 3, \dots, n - 2 \\ A(n - 1, n - 2 : n - 1) &= [36 \quad 144]. \end{aligned}$$

and

$$\begin{aligned} C(1, 1 : 2) &= [16 \quad 72] \\ C(2, 2 : 3) &= [60 \quad 144] \\ C(k, k : k + 1) &= [72 \quad 144] \quad k = 3, \dots, n - 2 \\ C(n - 1, n - 1 : n + 1) &= [72 \quad 132 \quad -24]. \end{aligned}$$

The third lines in the definition of A and C only hold for $n > 2$. All the other entries are equal to zero.

C Coefficients for fitting with a composite cubic curve

The problem (4.3) on $\mathcal{M} = \mathbb{R}^m$ is a quadratic function to be optimized with respect to the $2n + 2$ optimization variables $X = (p_0, b_0^+, b_1^-, b_1^+, \dots, b_n^-, p_n)$. As in appendix A, the solution of that problem reduces to m independent linear systems $(A_0 + \lambda A_1) \cdot X_k = \lambda C \cdot D_k$. This system depends on the regularization parameter $\lambda > 0$, on X , and on the points $(d_i)_{i=0}^n$ in D .

For $n \geq 4$, the matrices of coefficients $A_0, A_1 \in \mathbb{R}^{(2n+2) \times (2n+2)}$ and $C \in \mathbb{R}^{(2n+2) \times (n+1)}$ are given by the following sparse matrices.

A_0 is given, for $i = 2, \dots, n - 2$, by

$$\begin{aligned} A_0(1, 1 : 4) &= [24 \quad -36 \quad 6 \quad 6] \\ A_0(2, 1 : 4) &= [-36 \quad 72 \quad -36 \quad 0] \\ A_0(3, 1 : 6) &= [6 \quad -36 \quad 48 \quad -24 \quad 3 \quad 3] \\ A_0(4, 1 : 6) &= [6 \quad 0 \quad -24 \quad 48 \quad -33 \quad 3] \\ A_0(2i + 1, 2i - 1 : 2i + 6) &= [3 \quad -33 \quad 48 \quad -24 \quad 3 \quad 3] \\ A_0(2i + 2, 2i - 1 : 2i + 6) &= [3 \quad 3 \quad -24 \quad 48 \quad -33 \quad 3] \\ A_0(2n - 1, 2n - 3 : 2n + 2) &= [3 \quad -33 \quad 48 \quad -24 \quad 0 \quad 6] \\ A_0(2n, 2n - 3 : 2n + 2) &= [3 \quad 3 \quad -24 \quad 48 \quad -36 \quad 6] \\ A_0(2n + 1, 2n - 1 : 2n + 2) &= [0 \quad -36 \quad 72 \quad -36] \\ A_0(2n + 2, 2n - 1 : 2n + 2) &= [6 \quad 6 \quad -36 \quad 24]. \end{aligned}$$

The coefficients of A_1 , are

$$\begin{aligned} A_1(1, 1) &= \lambda \\ A_1(2, 2) &= 0 \\ A_1(2i - 1, 2i - 1 : 2i) &= \frac{1}{2} [\lambda, \lambda] \\ A_1(2n + 1, 2n + 1) &= 0 \\ A_1(2n + 2, 2n + 2) &= \lambda, \end{aligned}$$

for $i = 2, \dots, n$. Finally, the coefficients of C are given, for $i = 2, \dots, n$, by

$$\begin{aligned} C(1, 1) &= 2\lambda \\ C(2i - 1, i) &= \lambda \\ C(2i, i) &= \lambda \\ C(2n + 2, n + 1) &= 2\lambda. \end{aligned}$$

The other entries are equal to zero.

D Proof of equation (6.33)

The proof is extracted from [AGSW16b]. The total energy (6.25) of Bézier curves in Euclidean space is written

$$F[\mathbf{B}] = \frac{1}{4} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{r,s \in \{0,1\}} \sum_{i,j=0}^3 (L\tilde{T}S\tilde{U})_{ij,rs}^{mn} \cdot (\tilde{T}S\tilde{U})_{ij,rs}^{mn} \text{ for } \tilde{U} = (u_{kl}^{mn})_{(k,l) \in Q}^{(m,n) \in D}$$

and is minimized by

$$\tilde{U}_{\text{opt}} = -(S^*T^*LTS)^{-1}(S^*T^*LZ),$$

where a superscript asterisk denotes the adjoint operator

Indeed, let $(\cdot; \cdot)$ denote the natural inner product on Cartesian products of vector spaces (such that, for instance $F[\mathbf{B}] = (L\tilde{T}S\tilde{U}; \tilde{T}S\tilde{U}) = (L(TS\tilde{U} + Z); TS\tilde{U} + Z)$). Then, the minimizer comes from the first-order optimality condition

$$\begin{aligned} 0 &= (LTS\Phi; TS\tilde{U}_{\text{opt}} + Z) + (LTS\tilde{U}_{\text{opt}} + LZ; TS\Phi) \\ &= 2(LTS\tilde{U}_{\text{opt}} + LZ; TS\Phi) = 2(S^*T^*LTS\tilde{U}_{\text{opt}} + S^*T^*LZ; \Phi) \end{aligned}$$

for all variations Φ of \tilde{U}_{opt} .

E Examples of geometric elements on manifolds

Finally, we present here some examples of explicit formulae of the geometric elements on the sphere (table E.1) and the special orthogonal group (table E.2). They are implemented in the software Manopt [BMAS14] as a proper factory.

Sphere S^{m-1} : the set of normed vectors of size m .	
	$S^{m-1} = \{x \in \mathbb{R}^m : x^\top x = 1\}$ $T_x S^{m-1} = \{v \in \mathbb{R}^m : x^\top v = 0\}$
Inner product	$\langle v_1, v_2 \rangle_x = v_1^\top v_2$
Distance	$d(x, y) = \arccos(x^\top y)$
Exponential	$\exp_x(v) = x \cos(\ v\) + \frac{v}{\ v\ } \sin(\ v\)$
Logarithm	$\log_x(y) = \frac{(I_m - xx^\top)y}{\sqrt{1 - (x^\top y)^2}} \arccos(x^\top y)$
Transport	$P_{x \rightarrow y}(v) = -x \sin(\ \xi\) + \frac{\xi}{\ \xi\ } \cos(\ \xi\) \xi^\top v + \left(I_m - \frac{\xi \xi^\top}{\ \xi\ ^2} \right) v,$ $\xi = \log_x(y)$

Table E.1 Riemannian operators for S^{m-1} extracted from [Ren11].

The special orthogonal group $SO(m)$.	
	$SO(m) = \{X \in \mathbb{R}^{m \times m} : X^\top X = I, \det(X) = 1\}$ $T_X SO(m) = \{H \in \mathbb{R}^{m \times m} : X^\top H + H^\top X = 0\}$
Inner product	$\langle H_1, H_2 \rangle = \text{trace}(H_1^\top H_2)$
Distance	$d(X, Y) = \ \log(X^\top Y)\ _F$
Exponential	$\exp_X(H) = X \exp(X^\top H)$
Logarithm	$\log_X(Y) = X \log(X^\top Y)$
Transport	$P_{X \rightarrow Y}(H) = Y X^\top H$

Table E.2 Riemannian operators for $SO(m)$ extracted from [BA11]. Note the difference between $\exp_X(H)$, the Riemannian exponential, and $\exp(X)$, the matrix exponential.

List of publications

★ Publications in progress

[AGW20] P.-A. Absil, Pierre-Yves Gousenbourger, and Benedikt Wirth. Smooth surface fitting in Riemannian manifolds using patch-wise linearization. In progress, 2020

★ Publications in a journal

[BG18] Ronny Bergmann and Pierre-Yves Gousenbourger. A variational model for data fitting on manifolds by minimizing the acceleration of a Bézier curve. *Frontiers in Applied Mathematics and Statistics*, 4(59):1–16, 2018. doi:10.3389/fams.2018.00059

[GMA18c] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Data fitting on manifolds with composite Bézier-like curves and blended cubic splines. *Journal of Mathematical Imaging and Vision*, 61(5):645–671, 2018. doi:10.1007/s10851-018-0865-2

[AGSW16b] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striowski, and Benedikt Wirth. Differentiable piecewise-Bézier surfaces on Riemannian manifolds. *SIAM Journal on Imaging Sciences*, 9(4):1788–1828, 2016. doi:10.1137/16M1057978

★ **Book chapter**

[SGMS20] Nguyen Thanh Son, Pierre-Yves Gousenbourger, Estelle Massart, and Tatjana Stykel. Balanced truncation for parametric linear systems using interpolation of Gramians: a comparison of algebraic and geometric approaches, 2020. [arXiv:arXiv:2003.04577](https://arxiv.org/abs/2003.04577)

★ **Publications at a conference**

[SGMA19] Nguyen Thanh Son, Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Online balanced truncation for linear time-varying systems using continuously differentiable interpolation on Grassmann manifold. In *6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 165–170. IEEE, 2019

[MGS⁺19] Estelle Massart, Pierre-Yves Gousenbourger, Nguyen Thanh Son, Tatjana Stykel, and P.-A. Absil. Interpolation on the manifold of fixed-rank positive-semidefinite matrices for parametric model order reduction: preliminary results. In *ESANN2019*, pages 281–286. Springer, 2019

[GJA17] Pierre-Yves Gousenbourger, Laurent Jacques, and P.-A. Absil. Fast method to fit a C^1 piecewise-Bézier function to manifold-valued data points: how suboptimal is the curve obtained on the sphere S^2 ? In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, volume 10589 of *Lecture Notes in Computer Sciences*, pages 595–603, Berlin, Heidelberg, 2017. Springer. doi:10.1007/978-3-319-68445-1_69

[GMM⁺17] Pierre-Yves Gousenbourger, Estelle Massart, Antoni Musolas, P.-A. Absil, Laurent Jacques, Julien M Hendrickx, and Youssef Marzouk. Piecewise-Bézier C^1 smoothing on manifolds with application to wind field estimation. In *ESANN2017*, pages 305–310. Springer, 2017

[AGSW16a] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striowski, and Benedikt Wirth. Differentiable piecewise-Bézier interpolation on Riemannian manifolds. In *ESANN2016*, pages 95–100. Springer, 2016

[SGJ15] Chafik Samir, Pierre-Yves Gousenbourger, and Shantanu H. Joshi. Cylindrical surface reconstruction by fitting paths on shape space. In H. Drira, S. Kurttek, and P. Turaga, editor, *Proceedings of the 1st International*

Workshop on DIFFerential Geometry in Computer Vision for Analysis of Shapes, Images and Trajectories (DIFF-CV 2015), pages 11.1–11.10. BMVA Press, 2015. doi:10.5244/C.29.DIFFCV.11

[AGS⁺15] Antoine Arnould, Pierre-Yves Gousenbourger, Chafik Samir, P.-A. Absil, and Michel Canis. Fitting smooth paths on Riemannian manifolds: Endometrial surface reconstruction and preoperative MRI-based navigation. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, volume 9389 of *Lecture Notes in Computer Sciences*, pages 491–498, Berlin, Heidelberg, 2015. Springer. doi:10.1007/978-3-319-25040-3_53

[GSA14] Pierre-Yves Gousenbourger, Chafik Samir, and P.-A. Absil. Piecewise-Bézier C^1 interpolation on Riemannian manifolds with application to 2D shape morphing. In *International Conference on Pattern Recognition (ICPR)*, pages 4086–4091, 2014. doi:10.1109/ICPR.2014.700

★

Abstracts

[GB19] Pierre-Yves Gousenbourger and Ronny Bergmann. Data fitting on manifolds by minimizing the mean square acceleration of a Bézier curve. In *Benelux Meeting*, Lommel, Belgium, March 2019

[GMA18b] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Data fitting on manifolds with blended cubic splines. In *Proceedings of the 35th International Conference on Machine Learning (ICML), workshop Geometry in Machine Learning (GiMLi)*, Stockholm, Sweden, 2018

[GMA18a] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Blended smoothing splines on Riemannian manifolds. In *international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques (iTWIST)*, Marseille, France, 2018

[GM17] Pierre-Yves Gousenbourger and Estelle Massart. Wind field estimation via C^1 Bézier smoothing on manifolds. In *Benelux Meeting*, Spa, Belgium, March 2017

[GAWJ16] Pierre-Yves Gousenbourger, P.-A. Absil, Benedikt Wirth, and Laurent Jacques. Interpolation on manifolds using Bézier functions. In *international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques (iTWIST)*, Aalborg, Denmark, 2016

[GASW16] Pierre-Yves Gousenbourger, P.-A. Absil, Paul Striowski, and Benedikt Wirth. Interpolation on manifolds with differentiable surfaces of Bézier. In *Benelux Meeting*, Soestenberg, The Netherlands, March 2016

Bibliography

- [Abb84] Edwin A. Abbott. *Flatland*. Seeley & Co., Essex Street, 46 – London, 1884.
- [AGS⁺15] Antoine Arnould, Pierre-Yves Gousenbourger, Chafik Samir, P.-A. Absil, and Michel Canis. Fitting smooth paths on Riemannian manifolds: Endometrial surface reconstruction and preoperative MRI-based navigation. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, volume 9389 of *Lecture Notes in Computer Sciences*, pages 491–498, Berlin, Heidelberg, 2015. Springer. doi:10.1007/978-3-319-25040-3_53.
- [AGSW16a] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striowski, and Benedikt Wirth. Differentiable piecewise-Bézier interpolation on Riemannian manifolds. In *ESANN2016*, pages 95–100. Springer, 2016.
- [AGSW16b] P.-A. Absil, Pierre-Yves Gousenbourger, Paul Striowski, and Benedikt Wirth. Differentiable piecewise-Bézier surfaces on Riemannian manifolds. *SIAM Journal on Imaging Sciences*, 9(4):1788–1828, 2016. doi:10.1137/16M1057978.
- [AGW20] P.-A. Absil, Pierre-Yves Gousenbourger, and Benedikt Wirth. Smooth surface fitting in Riemannian manifolds using patch-wise linearization. In progress, 2020.
- [AMS08] Pierre-Antoine Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

- [Ant05] Athanasios C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM, Philadelphia, PA, 2005. doi:10.1137/1.9780898718713.
- [BA11] Nicolas Boumal and P.-A. Absil. A discrete regression method on manifolds and its application to data on $SO(n)$. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 18, pages 2284–2289, 2011. doi:10.3182/20110828-6-IT-1002.00542.
- [BBSW16] Miroslav Bačák, Ronny Bergmann, Gabriele Steidl, and Andreas Weinmann. A second order nonsmooth variational model for restoring manifold-valued images. *SIAM Journal on Computing*, 38(1):567–597, 2016. doi:10.1137/15M101988X.
- [BC70] Frederick Brickell and Roland S Clark. *Differentiable manifolds: an introduction*. Van Nostrand Reinhold, 1970.
- [Ber17] Ronny Bergmann. MVIRT, a toolbox for manifold-valued image registration. In *International Conference on Image Processing (ICIP), Beijing, China, September 17–20*. IEEE, 2017. URL: <https://ronnybergmann.net/mvirt/>, doi:10.1109/ICIP.2017.8296271.
- [BF01] Samuel R. Buss and Jay P. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics*, 20(2):95–126, 2001. doi:10.1145/502122.502124.
- [BFPS18] Ronny Bergmann, Jan Henrik Fitschen, Johannes Persch, and Gabriele Steidl. Priors with coupled first and second order differences for manifold-valued image processing. *Journal of Mathematical Imaging and Vision*, 60(9):1459–1481, 2018. doi:10.1007/s10851-018-0840-y.
- [BG18] Ronny Bergmann and Pierre-Yves Gousenbourger. A variational model for data fitting on manifolds by minimizing the acceleration of a Bézier curve. *Frontiers in Applied Mathematics and Statistics*, 4(59):1–16, 2018. doi:10.3389/fams.2018.00059.
- [BGW15] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric

- dynamical systems. *SIAM Review*, 57(4):483–531, 2015. doi: 10.1137/130932715.
- [BHSW18] Kristian Bredies, Martin Holler, Martin Storath, and Andreas Weinmann. Total Generalized Variation for manifold-valued data. *SIAM Journal on Imaging Sciences*, 11(3):1785–1848, 2018. doi:10.1137/17M1147597.
- [BLPS19] Ronny Bergmann, Friederike Laus, Johannes Persch, and Gabriele Steidl. Recent advances in denoising of manifold-valued images. In *Handbook of Numerical Analysis*, volume 20, pages 553–578. Elsevier, 2019.
- [BLSW14] Ronny Bergmann, Friederike Laus, Gabriele Steidl, and Andreas Weinmann. Second order differences of cyclic data and applications in variational denoising. *SIAM Journal on Imaging Sciences*, 7(4):2916–2953, 2014. doi:10.1137/140969993.
- [BMAS14] Nicolas Boumal, Bamdev Mishra, P.-A. Absil, and Rodolphe Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. URL: <http://www.manopt.org>.
- [BMS10] Silvère Bonnabel, Gilles Meyer, and Rodolphe Sepulchre. Adaptive filtering for estimation of a low-rank positive semidefinite matrix. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems*, 2010.
- [BMV18] Geir Bogfjellmo, Klas Modin, and Olivier Verdier. A numerical algorithm for C^2 -splines on symmetric spaces. *SIAM Journal on Numerical Analysis*, 56(4):2623–2647, 2018. doi: 10.1137/17M1123353.
- [BMWG07] Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. TRACKS: Toward directable thin shells. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):1–10, 2007. doi: 10.1145/1276377.1276439.
- [Boo86] William M Boothby. *An introduction to differentiable manifolds and Riemannian geometry*. Academic press, 1986.

- [Bou13] Nicolas Boumal. Interpolation and regression of rotation matrices. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, pages 345–352, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-40020-9_37.
- [Bou14] Nicolas Boumal. *Optimization and estimation on manifolds*. PhD thesis, Université catholique de Louvain, 2014.
- [Bou20] Nicolas Boumal. An introduction to optimization on smooth manifolds. Available online, May 2020. URL: <http://www.nicolasboumal.net/book>.
- [BRP12] Shalini Jain Bagaria, Darshana D. Rasalkar, and Bhawan K. Paunipagar. Imaging tools for endometriosis: Role of ultrasound, MRI and other imaging modalities in diagnosis and planning intervention. *Endometriosis, Basic Concepts and Current Research Trends*, 24:437–447, 2012.
- [BS09] Silvère Bonnabel and Rodolphe Sepulchre. Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1055–1070, 2009.
- [BXZ08] Chandrajit L. Bajaj, Guo-Liang Xu, and Qin Zhang. Biomolecule surfaces construction via a higher-order level-set method. *Journal of computer science and technology*, 23(6):1026–1036, 2008. doi:10.1007/s11390-008-9184-1.
- [Car46] Élie Cartan. *Leçons sur la géométrie des espaces de Riemann*. Gauthier-Villars, Paris, 1946. Deuxième édition, revue et augmentée.
- [CG15] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015.
- [CKS99] Peter Crouch, G. Kun, and Fatima Silva Leite. De Casteljau algorithm on Lie groups and spheres. *Journal of Dynamical and Control Systems*, 5(3):397–429, 1999. doi:10.1023/A:1021770717822.

- [dC92] Manfredo Perdigão do Carmo. *Riemannian Geometry*. Mathematics (Birkhäuser) theory. Birkhäuser Boston, 1992. doi: 10.1007/978-0-387-29403-2.
- [Des06] Dominique Desjeux. De la Seconde Guerre mondiale à aujourd’hui : les nouveaux questionnements de la consommation. *Que sais-je?*, 1(3754):24–46, 2006.
- [DFH05] Nira Dyn, M.S. Floater, and Kai Hormann. A C^2 four-point subdivision scheme with fourth order accuracy and its extensions. *Mathematical Methods for Curves and Surfaces*, 1(1):145–156, 2005.
- [DLG87] Nira Dyn, David Levin, and John A. Gregory. A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4(1):257–268, 1987. doi:10.1016/0167-8396(87)90001-X.
- [DM16] Chongyang Deng and Weiyin Ma. Efficient evaluation of subdivision schemes with polynomial reproduction property. *Journal of Computational and Applied Mathematics*, 294:403–412, 2016. doi:10.1016/j.cam.2015.09.008.
- [Duc77] Jean Duchon. Splines minimizing rotation-invariant seminorms in Sobolev spaces. In Walter Schempp and Karl Zeller, editors, *Constructive theory of functions of several variables*, volume 571 of *Lecture Notes in Mathematics*, pages 85–100. Springer Berlin Heidelberg, 1977. doi:10.1007/BFb0086566.
- [Duc78] Jean Duchon. Sur l’erreur d’interpolation des fonctions de plusieurs variables par les D^m -splines. *RAIRO Analyse Numérique*, 12(4):325–334, vi, 1978. doi:10.1051/m2an/1978120403251.
- [Dyn09] Nira Dyn. Linear and nonlinear subdivision schemes in geometric modeling. In Felipe Cucker, Allan Pinkus, and Michael J Todd, editors, *Foundation of Computational Mathematics*, volume 363, pages 68–92. Cambridge University Press, 2009. doi:10.1017/CB09781139107068.004.
- [EKPR19] Alexander Effland, Erich Kobler, Thomas Pock, and Martin Rumpf. Time discrete geodesics in deep feature spaces for

- image morphing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 171–182. Springer, 2019.
- [ERS⁺15] Alexander Effland, Martin Rumpf, Stefan Simon, Kirsten Stahn, and Benedikt Wirth. Bézier curves in the space of images. In Jean-François Aujol, Mila Nikolova, and Nicolas Papadakis, editors, *Scale Space and Variational Methods in Computer Vision*, volume 9087, pages 372–384. Springer International Publishing, 2015. doi:10.1007/978-3-319-18461-6_30.
- [Far02] Gerald E. Farin. *Curves and Surfaces for CAGD*. Academic Press, fifth edition, 2002.
- [Fle13] P. Thomas Fletcher. Geodesic regression and the theory of least squares on Riemannian manifolds. *International Journal of Computer Vision*, 105(2):171–185, 2013. doi:10.1007/s11263-012-0591-y.
- [Fré48] René Maurice Fréchet. Les éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l’Institut H. Poincaré*, 10:215–310, 1948.
- [GASW16] Pierre-Yves Gousenbourger, P.-A. Absil, Paul Striowski, and Benedikt Wirth. Interpolation on manifolds with differentiable surfaces of Bézier. In *Benelux Meeting*, Soestenberg, The Netherlands, March 2016.
- [GAWJ16] Pierre-Yves Gousenbourger, P.-A. Absil, Benedikt Wirth, and Laurent Jacques. Interpolation on manifolds using Bézier functions. In *international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques (iTWIST)*, Aalborg, Denmark, 2016.
- [GB19] Pierre-Yves Gousenbourger and Ronny Bergmann. Data fitting on manifolds by minimizing the mean square acceleration of a Bézier curve. In *Benelux Meeting*, Lommel, Belgium, March 2019.
- [GJA17] Pierre-Yves Gousenbourger, Laurent Jacques, and P.-A. Absil. Fast method to fit a C^1 piecewise-Bézier function to manifold-valued data points: how suboptimal is the curve obtained on

- the sphere S^2 ? In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, volume 10589 of *Lecture Notes in Computer Sciences*, pages 595–603, Berlin, Heidelberg, 2017. Springer. doi:10.1007/978-3-319-68445-1_69.
- [GM17] Pierre-Yves Gousenbourger and Estelle Massart. Wind field estimation via C^1 Bézier smoothing on manifolds. In *Benelux Meeting*, Spa, Belgium, March 2017.
- [GMA18a] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Blended smoothing splines on Riemannian manifolds. In *international Traveling Workshop on Interactions between low-complexity data models and Sensing Techniques (iTWIST)*, Marseille, France, 2018.
- [GMA18b] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Data fitting on manifolds with blended cubic splines. In *Proceedings of the 35th International Conference on Machine Learning (ICML), workshop Geometry in Machine Learning (GiMLi)*, Stockholm, Sweden, 2018.
- [GMA18c] Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Data fitting on manifolds with composite Bézier-like curves and blended cubic splines. *Journal of Mathematical Imaging and Vision*, 61(5):645–671, 2018. doi:10.1007/s10851-018-0865-2.
- [GMM⁺17] Pierre-Yves Gousenbourger, Estelle Massart, Antoni Musolas, P.-A. Absil, Laurent Jacques, Julien M Hendrickx, and Youssef Marzouk. Piecewise-Bézier C^1 smoothing on manifolds with application to wind field estimation. In *ESANN2017*, pages 305–310. Springer, 2017.
- [Gro08] Philipp Grohs. Smoothness analysis of subdivision schemes on regular grids by proximity. *SIAM Journal on Numerical Analysis*, 46(4):2169–2182, 2008. doi:10.1137/060669759.
- [GS93] Peter J. Green and Bernard W. Silverman. *Nonparametric Regression and Generalized Linear Models: A roughness penalty approach*. CRC Press, 1993.

- [GSA14] Pierre-Yves Gousenbourger, Chafik Samir, and P.-A. Absil. Piecewise-Bézier C^1 interpolation on Riemannian manifolds with application to 2D shape morphing. In *International Conference on Pattern Recognition (ICPR)*, pages 4086–4091, 2014. doi:10.1109/ICPR.2014.700.
- [HFJ14] Jacob Hinkle, P. Thomas Fletcher, and Sarang Joshi. Intrinsic polynomials for regression on Riemannian manifolds. *Journal of Mathematical Imaging and Vision*, 50(1):32–52, 2014. doi:10.1007/s10851-013-0489-5.
- [HHS⁺15] Mehrtash Harandi, Richard Hartley, Chunhua Shen, Brian Lovell, and Conrad Sanderson. Extrinsic methods for coding and dictionary learning on Grassmann manifolds. *International Journal of Computer Vision*, 114(2):113–136, 2015. doi:10.1007/s11263-015-0833-x.
- [HRWW12] Behrend Heeren, Martin Rumpf, Max Wardetzky, and Benedikt Wirth. Time-discrete geodesics in the space of shells. *Computer Graphics Forum*, 31(5):1755–1764, 2012. doi:10.1111/j.1467-8659.2012.03180.x.
- [HS07] Knut Hüper and Fatima Silva Leite. On the geometry of rolling and interpolation curves on S^n , SO_n , and Grassmann manifolds. *Journal of Dynamical and Control Systems*, 13(4):467–502, Oct 2007. doi:10.1007/s10883-007-9027-3.
- [HW19] Wen Huang and Ke Wei. Riemannian proximal gradient methods, 2019. arXiv:1909.06065.
- [HW20] Martin Holler and Andreas Weinmann. Non-smooth variational regularization for processing manifold-valued data. In *Handbook of Variational Methods for Nonlinear Geometric Data*, pages 51–93. Springer, 2020.
- [IT98] Jin-ichi Itoh and Minoru Tanaka. The dimension of a cut locus on a smooth Riemannian manifold. *Tohoku Mathematical Journal, Second Series*, 50(4):571–575, 1998.
- [JBAS10] Michel Journée, Francis Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

- [JK87] Peter E. Jupp and John T. Kent. Fitting smooth paths to spherical data. *Journal of Applied Statistics*, 36(1):34–46, 1987. doi:12/61765.
- [JKSJ07] Shantanu H. Joshi, Eric Klassen, Anuj Srivastava, and Ian Jermyn. A novel representation for Riemannian analysis of elastic curves in \mathbb{R}^n . In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007. doi:10.1109/CVPR.2007.383185.
- [JSTL07] Anand A. Joshi, David W. Shattuck, Paul M. Thompson, and Richard M. Leahy. Surface-constrained volumetric brain registration using harmonic mappings. *IEEE Transactions on Medical Imaging*, 26(12):1657–1669, 2007. doi:10.1109/TMI.2007.901432.
- [Kar77] Herman Karcher. Riemannian center of mass and mollifier smoothing. *Communications on pure and applied mathematics*, 30(5):509–541, 1977. doi:10.1002/cpa.3160300502.
- [KDB⁺20] Anis Kacem, Mohamed Daoudi, Boulbaba Ben Amor, Stefano Berretti, and Juan Carlos Alvarez-Paiva. A novel geometric framework on Gram matrix trajectories for human behavior understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 42(1):1–14, 2020. doi:10.1109/TPAMI.2018.2872564.
- [KDL18] Kwang-Rae Kim, Ian L. Dryden, and Huiling Le. Smoothing splines on Riemannian manifolds, with applications to 3D shape space, 2018. arXiv:1801.04978.
- [LB19] Changshuo Liu and Nicolas Boumal. Simple algorithms for optimization on Riemannian manifolds with constraints. *Applied Mathematics & Optimization*, pages 1–33, 2019. doi:10.1007/s00245-019-09564-3.
- [Lee97] John M. Lee. *Riemannian Manifolds: An Introduction to Curvature*, volume 176 of *Graduate Texts in Mathematics*. Springer Verlag, New-York, 1997. doi:10.1007/b98852.
- [Lee13] John M Lee. *Introduction to Smooth Manifolds*, volume 128 of *Graduate Texts in Mathematics*. Springer, New York, second edition, 2013.

- [LSKC13] Jan Lellmann, Evgeny Strelakovski, Sabrina Koetter, and Daniel Cremers. Total Variation regularization for functions with values in a manifold. In *International Conference on Computer Vision (ICCV)*, pages 2944–2951. IEEE, 2013. doi: 10.1109/ICCV.2013.366.
- [LSZD17] Lizhen Lin, Brian St. Thomas, Hongtu Zhu, and David B. Dunson. Extrinsic local regression on manifold-valued data. *Journal of the American Statistical Association*, 112(519):1261–1273, 2017. doi:10.1080/01621459.2016.1208615.
- [LW01] Achan Lin and Marshall Walker. CAGD techniques for differentiable manifolds. In *Proceedings of the 2001 International Symposium Algorithms for Approximation IV*, pages 36–43, 2001.
- [LW02] Jing-Rebecca Li and Jacob White. Low rank solution of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):260–280, 2002. doi:10.1137/S0895479801384937.
- [MA18] Estelle Massart and P.-A. Absil. Quotient geometry of the manifold of fixed-rank positive-semidefinite matrices. Technical Report UCL-INMA-2018.06, UCLouvain, November 2018. URL: <http://sites.uclouvain.be/absil/2018.06>.
- [Mas19] Estelle Massart. *Data fitting on positive-semidefinite matrix manifolds*. PhD thesis, UCLouvain, ICTEAM institute, 2019.
- [MBS11] Gilles Meyer, Silvère Bonnabel, and Rodolphe Sepulchre. Regression on fixed-rank positive semidefinite matrices: a Riemannian approach. *Journal of Machine Learning Research*, 12(Feb):593–625, 2011.
- [Mei79] Jean Meiguet. Multivariate interpolation at arbitrary points made simple. *Journal of Applied Mathematics and Physics (ZAMP)*, 30(1):292–304, 1979. doi:10.1007/BF01601941.
- [MGS⁺19] Estelle Massart, Pierre-Yves Gousenbourger, Nguyen Thanh Son, Tatjana Stykel, and P.-A. Absil. Interpolation on the manifold of fixed-rank positive-semidefinite matrices for parametric model order reduction: preliminary results. In *ESANN2019*, pages 281–286. Springer, 2019.

- [MHA19] Estelle Massart, Julien M. Hendrickx, and P-A. Absil. Curvature of the manifold of fixed-rank positive-semidefinite matrices endowed with the Bures–Wasserstein metric. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, volume 11712 of *Lecture Notes in Computer Sciences*, pages 739–748, Berlin, Heidelberg, 2019. Springer. doi: 10.1007/978-3-030-26980-7_77.
- [MHMF19] Rolando Mosquera, Abdallah El Hamidi, Aziz Hamdouni, and Antoine Falaize. Generalization of the Neville-Aitken interpolation algorithm on Grassmann manifolds: Applications to Reduced Order Model, 2019. arXiv:1907.02831.
- [MKJS19] Bamdev Mishra, Hiroyuki Kasai, Pratik Jawanpuria, and Atul Saroop. A Riemannian gossip approach to subspace learning on Grassmann manifold. *Machine Learning*, 108(10):1783–1803, 2019.
- [MM17] Luís Machado and M. Teresa T. Monteiro. A numerical optimization approach to generate smoothing spherical splines. *Journal of Geometry and Physics*, 111:71–81, 2017. doi:10.1016/j.geomphys.2016.10.007.
- [MMRB72] Donella H Meadows, Dennis L Meadows, Jorgen Randers, and William W Behrens. The limits to growth. *New York*, 102:27, 1972.
- [MMS11] Bamdev Mishra, Gilles Meyer, and Rodolphe Sepulchre. Low-rank optimization for distance matrix completion. In *50th Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 4455–4460. IEEE, 2011. doi:10.1109/CDC.2011.6160810.
- [MS06] Luís Machado and Fatima Silva Leite. Fitting smooth paths on Riemannian manifolds. *International Journal of Applied Mathematics & Statistics*, 4(J06):25–53, 2006.
- [MSLK10] Luís Machado, Fatima Silva Leite Leite, and Krzysztof Krakowski. Higher-order smoothing splines versus least squares problems on Riemannian manifolds. *Journal of Dynamical Control and Systems*, 16(1):121–148, 2010. doi:10.1007/s10883-010-9080-1.

- [MSM18] Luís Machado, Fatima Silva Leite, and M. Teresa T. Monteiro. Path planning trajectories in fluid environments. In *13th APCA International Conference on Control and Soft Computing (CONTROLO)*, pages 219–223, 2018. doi:10.1109/CONTROLO.2018.8514278.
- [Muc12] Domenico Mucci. Maps into projective spaces: liquid crystal and conformal energies. *Discrete and Continuous Dynamical Systems*, 17(2):597–635, 2012.
- [NYP13] Esfandiar Nava-Yazdani and Konrad Polthier. De Casteljau’s algorithm on manifolds. *Computer Aided Geometric Design*, 30(7):722–732, 2013. doi:10.1016/j.cagd.2013.06.002.
- [NYY11] Esfandiar Nava-Yazdani and Thomas P. Y. Yu. On Donoho’s Log-Exp subdivision scheme: Choice of retraction and time-symmetry. *Multiscale Modeling & Simulation*, 9(4):1801–1828, 2011. doi:10.1137/100804838.
- [O’N66] Barrett O’Neill. *Elementary differential geometry*. Academic Press INC, London, 1966.
- [O’N83] Barrett O’Neill. *Semi-Riemannian geometry with applications to relativity*. Academic press, 1983.
- [PA16] Lorenz Pyta and Dirk Abel. Interpolatory Galerkin models for the Navier-Stokes-equations. *IFAC-PapersOnLine*, 49(8):204–209, 2016. doi:10.1016/j.ifacol.2016.07.442.
- [Par10] Jonghoon Park. Interpolation and tracking of rigid body orientations. In *ICCAS*, pages 668–673, 2010.
- [Per18] Johannes Persch. *Optimization Methods in Manifold-Valued Image Processing*. PhD thesis, Technische Universität Kaiserslautern, 2018.
- [PFA06] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006. doi:10.1007/s11263-005-3222-z.
- [PN07] Tomasz Popiel and Lyle Noakes. Bézier curves and C^2 interpolation in Riemannian manifolds. *Journal of Approximation*

- Theory*, 148(2):111–127, 2007. doi:10.1016/j.jat.2007.03.002.
- [Pow94] Michael J. D. Powell. The uniform convergence of thin plate spline interpolation in two dimensions. *Numerische Mathematik*, 68(1):107–128, 1994. doi:10.1007/s002110050051.
- [PR08] Jörg Peters and Ulrich Reif. *Subdivision surfaces*, volume 3 of *Geometry and Computing*. Springer-Verlag, Berlin, 2008. doi:10.1007/978-3-540-76406-9.
- [Ren11] Quentin Rentmeesters. A gradient method for geodesic data fitting on some symmetric Riemannian manifolds. In *50th Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 7141–7146. IEEE, 2011. doi:10.1109/CDC.2011.6161280.
- [Ren13] Quentin Rentmeesters. *Algorithms for data fitting on some common homogeneous spaces*. PhD thesis, Université catholique de Louvain, 2013.
- [Rie54] Bernhard Riemann. Über die Hypothesen, welche der Geometrie zu Grunde liegen. *Physikalische Blätter*, 10(7):296–306, 1854. doi:10.1002/phbl.19540100702.
- [RW15] Martin Rumpf and Benedikt Wirth. Variational time discretization of geodesic calculus. *IMA Journal of Numerical Analysis*, 35(3):1011–1046, 2015. doi:10.1093/imanum/dru027.
- [San10] Oliver Sander. Geodesic finite elements for Cosserat rods. *International Journal for Numerical Methods in Engineering*, 82(13):1645–1670, 2010. doi:10.1002/nme.2814.
- [San16] Oliver Sander. Geodesic finite elements of higher order. *IMA Journal of Numerical Analysis*, 36(1):238–266, 2016. doi:10.1093/imanum/drv016.
- [SASK12] Chafik Samir, P.-A. Absil, Anuj Srivastava, and Eric Klassen. A gradient-descent method for curve fitting on Riemannian manifolds. *Foundations of Computational Mathematics*, 12(1):49–73, 2012. doi:10.1007/s10208-011-9091-7.

- [SAV⁺19] David Sabbagh, Pierre Ablin, Gaël Varoquaux, Alexandre Gramfort, and Denis A Engemann. Manifold-regression to predict from MEG/EEG brain signals without source modeling. In *Advances in Neural Information Processing Systems*, pages 7321–7332, 2019.
- [SC11] Evgeny Strekalovskiy and Daniel Cremers. Total Variation for cyclic structures: Convex relaxation and efficient minimization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1905–1911. IEEE, 2011. doi:10.1109/CVPR.2011.5995573.
- [SC13] Evgeny Strekalovskiy and Daniel Cremers. Total Cyclic Variation and generalizations. *Journal of Mathematical Imaging and Vision*, 47(3):258–277, 2013. doi:10.1007/s10851-012-0396-1.
- [SDK⁺12] Jingyong Su, Ian L. Dryden, Eric Klassen, H. Le, and Anuj Srivastava. Fitting smoothing splines to time-indexed, noisy points on nonlinear manifolds. *Image and Vision Computing*, 30(6–7):428–442, 2012. doi:10.1016/j.imavis.2011.09.006.
- [SGJ15] Chafik Samir, Pierre-Yves Gousenbourger, and Shantanu H. Joshi. Cylindrical surface reconstruction by fitting paths on shape space. In H. Drira, S. Kurtek, and P. Turaga, editor, *Proceedings of the 1st International Workshop on DIFFerential Geometry in Computer Vision for Analysis of Shapes, Images and Trajectories (DIFF-CV 2015)*, pages 11.1–11.10. BMVA Press, 2015. doi:10.5244/C.29.DIFFCV.11.
- [SGMA19] Nguyen Thanh Son, Pierre-Yves Gousenbourger, Estelle Massart, and P.-A. Absil. Online balanced truncation for linear time-varying systems using continuously differentiable interpolation on Grassmann manifold. In *6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 165–170. IEEE, 2019.
- [SGMS20] Nguyen Thanh Son, Pierre-Yves Gousenbourger, Estelle Massart, and Tatjana Stykel. Balanced truncation for parametric linear systems using interpolation of Gramians: a comparison of algebraic and geometric approaches, 2020. arXiv:arXiv:2003.04577.

- [Shi08] Tatiana Shingel. Interpolation in special orthogonal groups. *IMA journal of numerical analysis*, 29(3):731–745, 2008.
- [SHPS08] Florian Steinke, Matthias Hein, Jan Peters, and Bernhard Schölkopf. Manifold-valued thin-plate splines with applications in computer graphics. *Computer Graphics Forum*, 27(2):437–448, 2008. doi:10.1111/j.1467-8659.2008.01141.x.
- [SHS10] Florian Steinke, Matthias Hein, and Bernhard Schölkopf. Nonparametric regression between general Riemannian manifolds. *SIAM Journal on Imaging Sciences*, 3(3):527–563, 2010. doi:10.1137/080744189.
- [SK16] Anuj Srivastava and Eric P. Klassen. *Functional and shape data analysis*, volume 475. Springer, 2016.
- [SKJJ11] A. Srivastava, E. Klassen, S. H. Joshi, and I. H. Jermyn. Shape analysis of elastic curves in Euclidean spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:1415–1428, 2011.
- [SKK04] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571, 2004. doi:10.1109/TPAMI.2004.1273924.
- [SKKS14] Jingyong Su, Sebastian Kurtek, Eric Klassen, and Anuj Srivastava. Statistical analysis of trajectories on Riemannian manifolds: Bird migration, hurricane tracking and video surveillance. *The Annals of Applied Statistics*, 8(1):530–552, 2014. doi:10.1214/13-AOAS701.
- [SM03] Endre Süli and David Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.
- [SNB16] Oliver Sander, Patrizio Neff, and Mircea Bîrsan. Numerical treatment of a geometrically nonlinear planar Cosserat shell model. *Computational Mechanics*, 57(5):817–841, 2016. doi:10.1007/s00466-016-1263-5.

- [SQW17] Ju Sun, Qing Qu, and John Wright. A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 8 2017. doi:10.1007/s10208-017-9365-9.
- [TKW16] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *The Journal of Machine Learning Research*, 17(1):4755–4759, 2016.
- [Van13] Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013. doi:10.1137/110845768.
- [VAV09] Bart Vandereycken, P.-A. Absil, and Stefan Vandewalle. Embedded geometry of the set of symmetric positive semidefinite matrices of fixed rank. In *15th Workshop on Statistical Signal Processing (SSP)*, pages 389–392. IEEE, 2009.
- [WD05] Johannes Wallner and Nira Dyn. Convergence and analysis of subdivision schemes on manifolds by proximity. *Computer Aided Geometric Design*, 22(7):593–622, 2005. doi:10.1016/j.cagd.2005.06.003.
- [WDS14] Andreas Weinmann, Laurent Demaret, and Martin Storath. Total Variation regularization for manifold-valued data. *SIAM Journal on Imaging Sciences*, 7(4):2226–2257, 2014. doi:10.1137/130951075.
- [Wei10] Andreas Weinmann. Nonlinear subdivision schemes on irregular meshes. *Constructive Approximation*, 31(3):395–415, 2010. doi:10.1007/s00365-009-9063-1.
- [Wei12] Andreas Weinmann. Subdivision schemes with general dilation in the geometric and nonlinear setting. *Journal of Approximation Theory*, 164(1):105–137, 2012. doi:10.1016/j.jat.2011.09.005.
- [Wik] MOR Wiki — Model Order Reduction Wiki. <http://morwiki.mpi-magdeburg.mpg.de/morwiki/>.
- [WLS⁺20] Michael Watterson, Sikang Liu, Ke Sun, Trey Smith, and Vijay Kumar. Trajectory optimization on manifolds with applica-

- tions to quadrotor systems. *The International Journal of Robotics Research*, page 0278364919891775, 2020.
- [WNYG07] Johannes Wallner, Esfandiar Nava-Yazdani, and Philip Grohs. Smoothness properties of Lie group subdivision schemes. *Multiscale Modeling & Simulation*, 6(2):493–505, 2007. doi:10.1137/060668353.
- [WP06] Johannes Wallner and Helmut Pottmann. Intrinsic subdivision with smooth limits for graphics and animation. *ACM Transactions on Graphics*, 25(2):356–374, 2006. doi:10.1145/1138450.1138459.
- [YLSL11] I-Cheng Yeh, Chao-Hung Lin, Olga Sorkine, and Tong-Yee Lee. Template-based 3D model fitting using dual-domain relaxation. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1178–1190, 2011. doi:10.1109/TVCG.2010.124.
- [YSZL16] Xianghao Yu, Juei-Chin Shen, Jun Zhang, and Khaled Ben Letaief. Alternating minimization algorithms for hybrid precoding in millimeter wave MIMO systems. *IEEE Journal of Selected Topics in Signal Processing*, 10(3):485–500, 2016. doi:10.1109/JSTSP.2016.2523903.
- [Zim19] Ralf Zimmermann. Hermite interpolation and data processing errors on Riemannian matrix manifolds, 2019. arXiv:1908.05875.
- [ZN19] Erchuan Zhang and Lyle Noakes. Optimal interpolants on Grassmann manifolds. *Mathematics of Control, Signals, and Systems*, 31(3):363–383, Sep 2019. doi:10.1007/s00498-019-0241-9.
- [ZOF01] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *Workshop on Variational and Level Set Methods in Computer Vision (VLSM)*, pages 194–201. IEEE, 2001. doi:10.1109/VLSM.2001.938900.
- [ZYZY15] Xiaowei Zhou, Can Yang, Hongyu Zhao, and Weichuan Yu. Low-rank modeling and its applications in image analysis. *ACM Computing Surveys (CSUR)*, 47(2):36, 2015.

List of symbols

Fitting curves and surfaces

- C^k _____ k -continuous differentiability (p. 23)
- $\Gamma_{\mathbf{B}}$ _____ set of optimization variables (p. 62)
- λ _____ regularization parameter (p. 77)
- Ω_{ij} _____ rectangular domain for composite surfaces (p. 204)
- ψ_i _____ blossom function (p. 100)
- $A(\mathbf{b})$ _____ discretized MSA (p. 129)
- $\beta_K(t; b_0, \dots, b_K)$ _____ Bézier curve of degree K (p. 48)
- $\mathbf{B}(t)$ _____ composite curve (Bézier, blended or Bézier-like) (p. 50)
- $\tilde{\beta}_K(t; \tilde{b}_0, \dots, \tilde{b}_K)$ _____ tangent Bézier curve of degree K in $T_x\mathcal{M}$ (p. 91)
- \tilde{b}_i, \hat{b}_i _____ control vector in $T_x\mathcal{M}$ (p. 91)
- b_i, b_{ij} _____ control point (p. 48)
- $B_{jK}(t)$ _____ j^{th} Bernstein polynomial of degree K (p. 48)
- d_{ref} _____ reference point (p. 63)
- d_i, d_{ij} _____ data point (p. 2)
- $f_i(\cdot, x, y)$ _____ endpoint function (blossoms) (p. 99)
- $g_i^{[k]}(t)$ _____ i^{th} Bézier curve of degree k (De Casteljau) (p. 134)

★ | List of symbols

- p_{ij} _____ linearization point (p. 204)
- $\sigma_K(t_1, t_2; (b_{ij})_{i,j=0,\dots,K})$ _____ Bézier surface of degree K (p. 158)
- σ_K^I _____ Bézier surface of type I (averaging) (p. 162)
- σ_K^{II} _____ Bézier surface of type II (tensorization) (p. 162)
- σ_K^{III} _____ Bézier surface of type III (De Casteljau) (p. 162)
- $\mathbf{S}(t)$ _____ composite surface (Bézier, blended or thin plate spline) (p. 157)
- S_{ij} _____ local surface associated to p_{ij} (p. 204)
- \tilde{S}_{ij} _____ tangent surface at $T_{p_{ij}}\mathcal{M}$ (p. 204)
- t_i, t_{ij} _____ parameter associated with a data point (p. 2)
- $w(t)$ _____ weight function (p. 91)
- $x_i^{[k]}$ _____ i^{th} point of the k^{th} step of the De Casteljau algorithm (p. 50)

Elements of differential geometry

- \sim _____ equivalence (p. 24)
- (\mathcal{M}, g) _____ Riemannian manifold (p. 31)
- $(\mathcal{U}, \varphi), (\mathcal{V}, \psi)$ _____ chart (p. 20)
- $(M, \mathcal{A}^+), (N, \mathcal{B}^+)$ _____ manifold (general definition) (p. 22)
- $[\cdot, \cdot]$ _____ Lie bracket (p. 33)
- \mathcal{A}, \mathcal{B} _____ atlas (p. 22)
- $\text{av}[(x_1, \dots, x_n), (w_1, \dots, w_n)]$ _____ weighted geodesic average (p. 39)
- $\text{co}(U)$ _____ multigeodesically convex hull (p. 40)
- $d_2[x, y, z]$ _____ second order absolute finite differences [BBSW16] (p. 129)
- $d_{\mathcal{M}}(\cdot, \cdot)$ _____ Riemannian distance (p. 31)
- $\text{EXP}_x^n(v)$ _____ discrete logarithm (p. 44)
- $\exp_x(\xi)$ _____ Riemannian exponential map (p. 37)

$\frac{D^2}{dt^2}$	Levi-Civita second covariant derivative (p. 36)
γ	curve (p. 27)
$\Gamma_{i,j}^k$	Christoffel symbols (p. 35)
$\Gamma_{g,\xi}$	geodesic variation of $g(\cdot; x, y)$ w.r.t $\xi \in T_x\mathcal{M}$ (p. 131)
$\langle \cdot, \cdot \rangle_x$	inner product on $T_x\mathcal{M}$ (p. 31)
$\frac{1}{n}\text{LOG}$	discrete logarithm (p. 43)
$\log_x(y)$	Riemannian logarithmic map (p. 37)
$\eta_i^{[k]}$	derivative of $g_i^{[k]}$ w.r.t x in direction $\eta \in T_x\mathcal{M}$ (p. 134)
∇	affine connection (p. 33)
$\nabla_\eta \xi$	covariant derivative of ξ w.r.t η (p. 34)
$\nabla_{\mathcal{M}} f(x)$	Riemannian gradient (p. 32)
$P_{x \rightarrow y}(\xi_x)$	parallel transport of ξ_x along $g(t; x, y)$ (p. 41)
$\mathcal{F}(\mathcal{M})$	set of smooth functions (p. 27)
$\mathcal{X}(\mathcal{M})$	set of all vector fields (p. 30)
ξ_x	tangent vector at $x \in \mathcal{M}$ (p. 27)
$D_x f[\xi]$	directional derivative of f w.r.t x in direction ξ (p. 130)
$g(\xi, \eta)$	Riemannian metric, $\xi, \eta \in T_x\mathcal{M}$ (p. 31)
$g(t; x, y)$	geodesic (p. 36)
$J_{E,\nu}^*$	adjoint Jacobi field (p. 143)
$J_{g,\xi}$	Jacobi field along $g(\cdot; x, y)$ (p. 132)
R_x	retraction (p. 30)
r_x	injectivity radius (p. 38)
$T\mathcal{M}$	tangent bundle (p. 29)
$T_x\mathcal{M}$	tangent space (p. 27)

Spaces and manifolds

- $\text{Gr}(n, p)$ _____ Grassmann manifold (p. 25)
- \mathbb{L}^2 _____ set of square integrable functions (p. 69)
- \mathcal{M}, \mathcal{N} _____ (Riemannian) manifold (p. 31)
- $\mathbb{R}_*^{n \times p}$ _____ set of full rank $n \times p$ matrices (p. 115)
- \mathcal{P} _____ shape space of open polygonal curves (p. 193)
- \mathcal{S}_h _____ shape space of 3D-shells (p. 195)
- \mathbb{N} _____ natural numbers (p. 2)
- \mathcal{O}_n _____ orthogonal group (p. 25)
- $\mathcal{S}_+(n, p)$ _____ set of positive semidefinite matrices of size n , rank p (p. 115)
- \mathbb{R} _____ real numbers (p. 2)
- \mathbb{R}^m _____ Euclidean space (p. 22)
- \mathcal{S} _____ shape space of closed curves (p. 70)
- $\text{SO}(n)$ _____ special orthogonal group (p. 25)
- \mathbb{S}^{n-1} _____ sphere (p. 29)
- $\text{St}(n, p)$ _____ Stiefel manifold (p. 23)

Acronyms

- BL-I _____ Bézier-Like curve (type I) (p. 107)
- BL-II _____ Bézier-Like curve (type II) (p. 107)
- CFD _____ Computational Fluid Dynamics (p. 114)
- LC _____ Local Curve (p. 107)
- MOR _____ Model Order Reduction (p. 120)
- MSA _____ Mean squared acceleration (p. 128)
- PMOR _____ Parametric Model Order Reduction (p. 120)

Index

- $\text{Gr}(n, p)$, 25
- \mathcal{O}_d , 25
- λ , 77
- $\mathcal{S}_+(n, p)$, 25, 84, 107, 112, 115, 121
- $\text{SO}(n)$, 25, 29, 110, 153, 192

- acceleration, 36, 74
- affine connection, 33
 - Levi-Civita, 34
 - Riemannian connection, 34
- atlas, 22
 - equivalence, 22
 - maximal atlas, 22

- B-spline, 188
- Bernstein polynomial, 48, 158
- blossom, 100

- CFD, 114
- chain rule, 130
- chart, 20
 - compatible, 21
 - transition map, 21
- Christoffel symbols, 35
- control point, 48, 58, 63, 80, 173, 178, 187, 190
- control vector, 91
- convex hull, 40
- covariant derivative, 34, 128

- curve, 27
 - Bézier curve, 48, 54, 134, 159
 - Bézier-like curve, 101, 104
 - blended curve, 91, 93
 - composite Bézier curve, 50, 64, 81, 88, 138
 - hybrid Bézier curve, 56, 71
 - local curve, 91
 - tangent curve, 91
- cut locus, 38, 87

- data point, 2
- De Casteljau, 50, 54, 100, 134, 159, 162
- differentiability conditions, 52, 55, 60, 66, 81, 96, 103, 106, 169, 175, 209, 215
- directional derivative, 130, 134
- distance, 31, 39
 - discrete distance, 42

- endpoint function, 99
- equivalence, 24
- Euclidean space, 22, 48, 58, 62, 79, 146
- Exp-Log complexity, 66, 89, 98, 104, 106, 208, 216
- exponential map, 18, 37, 70, 116
 - discrete exponential, 44

- fiber, 24
- finite differences, 42, 129
- fitting, 2, 81, 88, 93, 202, 214
- geodesic, 17, 36, 70, 115, 134
 - average, 39, 206, 211
 - average (dyadic), 206
 - coupled, 133
 - discrete n -geodesic, 43
 - discrete average, 43
 - Exp-Log, 18, 38, 39, 54, 166
- geodesic variation, 131
- gradient, 32, 130, 139
- Grassmann manifold, 25
- infinite loop, 266,
- injectivity radius, 38, 85
- inner product, 31
- interpolation, 60, 64, 66, 81, 85, 87, 89, 96, 191, 192, 202
- Jacobi field, 132, 134
 - adjoint, 143
- Karcher mean, 39
- Levi-Civita, 34, 36, 128
- Lie bracket, 33
- linearization point, 204
- logarithmic map, 18, 37, 71, 116, 184
 - discrete logarithm, 43
- manifold, 16, 22
 - embedded submanifold, 23
 - product manifold, 24
 - quotient manifold, 24
 - Riemannian manifold, 17, 31
- metric, 17, 31, 70, 115
- minimal representation, 66, 89, 98, 103, 106, 215
- MOR, 120
- MSA, 129, 139
- natural cubic spline, 64, 81, 96, 109
- norm, 31
- orthogonal group, 25
- parallel transport, 19, 40, 41, 185
 - Schild's ladder, 44
- PMOR, 120
- proper set, 40
- rank, 84
- reference point, 63, 80, 91, 204
- regularizer, 77
- retraction, 18, 30, 37
- shape space, 70, 193, 195
- smooth map, 23
- special orthogonal group, 25
- sphere, 23, 29, 56, 85, 107, 110, 147, 191, 217
- Stiefel manifold, 23, 25
- surface, 157
 - Bézier surface, 158, 159, 162, 174
 - blended surface, 206, 214
 - composite Bézier surface, 160
 - local surface, 204
 - tangent surface, 204
 - thin plate spline, 201
- symmetric space, 132, 168
- tangent bundle, 29
- tangent space, 17, 27
 - root, 27
- tangent vector, 27
- thin plate spline, 201, 215
- transvaginal ultrasound, 67
- vector field, 30

vector transportation, 40

velocity, 74

weight function, 91, 174, 212, 213,
219

wind field, 84, 107, 112

