Geometric Optimisation on Manifolds with Applications to Deep Learning



Mario Lezcano Casado
Keble College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2021

Acknowledgements

I would like to begin by thanking my supervisors, Prof. Raphael Hauser and Prof. Vidit Nanda for their time, effort, and guidance throughout this thesis. Thank you for giving me the flexibility to delve into whatever mad topic I found interesting at the time. Discussions with Jaime Mendizabal and David Martínez-Rubio have been truly excellent. Their suggestions, clarifications, and probing questions pushed me to think in new ways and greatly improved the thesis. My collaborators Adam Golinski and Tom Rainforth helped me to explore the applications that this research has in other fields. Prof. Jose Manuel Gamboa Mutuberría has been a constant source of support and encouragement all these years. To all of them a warm thank you.

A heartfelt thank you to my parents and my brother for their unconditional support, in particular during the final writing of this thesis.

On the less academic side, I would like to thank everybody in and around La Katacumbia. Borondiño, Juan, Adri, Álvaro, Jorge, Recio, Miguel, David vinos, Sam, Felix, Steve, Adam, Sergio, Daniel, María, Laura and everyone at 11 Iffley road... You have made these years in the small city of Oxford incredibly enjoyable. From the parties by the river, to nights at Plush and Cellar, barbecues, swimming pool, squash, gigs at the Tap Social and even a very special summer solstice in Stonehenge followed by a weekend in Bristol that was not for the faint-hearted. In all fairness, a long list of things to have done in the small town of Oxford.

I must single out Garima, Adri, and David, with whom I spent the best of lockdowns. Starting as an odd night in London that looked straight from a B film, it was followed by some of the best food I've had in my life—the arroz con leche, risotto, and banana bread come to mind—Smash-up, Hollow Knight, Cuphead, the month that we worked out... For these things and many more, thank you for making this surreal year special.

I would also like to thank Amigas y Conocidas, Roberto, Lorenzo, Cámara, Charles, Visierdo, Inés and those floating around it such as my beloved Juan. It is always fantastic to be back and be able to forget whatever I'm working on and just have a great night out, whether in Burgos or in Berlin.

There are a few more that have made the last few years enjoyable—Irene and Elyem from la Caixa, Mendi, Paco, David, Aguirre, Aitor and Adrià from Comeis, Guillem and Sara, Debbie, Lina, Elena, Noémie, Manolo...—with whom I have shared some truly amazing moments.

To all of you: Thank you.

Contents

1 Introduction **1**

1.1 Why this Thesis? . . . 1

1.1.1 A practical reason . . . 1

1.1.2 A theoretical reason . . . 2

1.2 Contributions and Outline . . . 4

2 Machine Learning and Neural Networks **6**

2.1 Offline Statistical Learning . . . 6

2.1.1 Framework and terminology . . . 7

2.1.2 A note on statistical learning theory . . . 9

2.2 First Order Optimisation . . . 11

2.2.1 The complexity model . . . 11

2.2.2 Convex optimisation . . . 14

2.2.3 Non-convex optimisation . . . 16

2.3 Neural Network Architectures . . . 18

2.3.1 Neural networks . . . 18

2.3.2 Feedforward networks . . . 18

2.3.3 Recurrent neural networks (RNNs) . . . 19

2.3.4 Why affine layers? . . . 21

2.3.4.1 Graphics Processing Units (GPUs) . . . 21

2.3.4.2 Auto-differentiation: Forward and backward pass . . . 22

2.3.5 An implementation of a feedforward network . . . 22

3 Differential and Riemannian Geometry **26**

3.1 Fibred Manifolds . . . 26

3.2 Invariant Metrics and Principal Bundles . . . 31

3.3 Homogeneous Spaces . . . 36

3.4 Naturally Reductive Homogeneous Spaces . . . 41

3.5 Matrix Manifolds . . . 46

3.5.1 The Stiefel manifold . . . 47

3.5.2 The Grassmannian . . . 50

3.5.3 Symmetric positive definite matrices . . . 52

3.5.4 Fixed-rank matrices . . . 53

4 Fibred Manifolds in First Order Optimisation 54

4.1 Optimisation on Manifolds . . . 54
4.2 Fibred Manifolds and First Order Structure . . . 56
4.3 Riemannian Submersions . . . 63
4.4 Static Trivialisations . . . 67
4.4.1 Diffeomorphisms . . . 67
4.4.2 The Riemannian exponential . . . 67
4.4.2.1 Hadamard manifolds . . . 68
4.4.2.2 The general case . . . 68
4.4.2.3 The adjoint of the Riemannian exponential . . . 70
4.5 Dynamic Trivialisations . . . 73

5 First and Second Order Bounds on the Riemannian Exponential 77

5.1 Introduction . . . 77
5.1.1 The problem and overview of assumptions . . . 77
5.1.2 Previous work . . . 79
5.2 Differential Geometry: Conventions and Notation . . . 81
5.3 First Order Bounds for the Exponential Map . . . 84
5.3.1 Jacobi fields . . . 85
5.3.2 Parallel vector fields . . . 88
5.3.3 Rauch's theorem . . . 88
5.3.4 The law of cosines on manifolds . . . 94
5.4 Second Order Bounds for the Exponential Map . . . 96
5.4.1 The differential equation . . . 96
5.4.2 Manifolds of bounded geometry . . . 98
5.4.3 Curvature bounds . . . 100
5.4.4 A comparison lemma . . . 101
5.4.5 A second order version of Rauch's theorem . . . 103
5.5 Concrete Second Order Bounds . . . 106
5.5.1 Constant curvature . . . 106
5.5.2 Locally symmetric spaces . . . 107
5.6 Convergence Rates for Dynamic Trivialisations . . . 110
5.7 When Is a Retraction an Exponential Map? . . . 113

6 GeoTorch: A Library for Optimisation on Manifolds at Scale 114

6.1 Orthogonal Constraints within Neural Networks . . . 115
6.1.1 Vanishing and exploding gradient problems . . . 115
6.1.2 Optimisation with orthogonal constraints . . . 117

6.2 GeoTorch . . . 120
6.2.1 Libraries for optimisation on manifolds . . . 120
6.2.2 The design of GeoTorch . . . 121
6.2.2.1 Reusing pre-existing classes . . . 123
6.2.2.2 Caching tensors . . . 124
6.2.2.3 Injecting properties into classes . . . 126
6.2.2.4 Initialising parametrisations . . . 127
6.2.2.5 Class system . . . 128
6.3 Exponential Recurrent Neural Networks (EXPRNN / DTRIV) . . . 131
6.3.1 Comparison with previous approaches . . . 131
6.3.1.1 Riemannian Gradient Descent (RGD) . . . 132
6.3.1.2 Parametrisations . . . 132
6.3.2 The exponential map on a GPU . . . 134
6.3.3 Non-linearities . . . 136
6.3.4 Initialisation . . . 137
6.4 Experiments . . . 138
6.4.1 Copying memory task . . . 138
6.4.2 Pixel-by-pixel MNIST . . . 139
6.4.3 TIMIT speech dataset . . . 140

Bibliography **142**

Chapter 1

Introduction

1.1 Why this Thesis?

We start this thesis by justifying the theoretical and practical approach that this thesis takes. We outline the general problems that we study on throughout it. After this, we summarise the results in the thesis and its structure.

1.1.1 A practical reason

Optimisation has always been a field with vast applications in the real world, bringing together the fields of analysis and constructive mathematics—and subsequently computer science—in order to be able to approximate in a principled way the minima of functions traditionally defined on some subset of the Euclidean space.

A natural extension of this setting is that of optimisation on Riemannian manifolds. Most of the interesting sets of matrices used in the context of optimisation turn out to have a manifold structure. Optimisation on manifolds is both theoretically and practically challenging due to the inherent complexity of the objects involved. Even then, optimisation on matrix manifolds has proven to be rather useful in many subfields of machine learning and neural networks (NN). Examples of interesting matrix manifolds in the context of gradient-based optimisation are the set of positive definite matrices in Bayesian statistics as kernels (Rasmussen and Williams 2005), orthogonal matrices within recurrent neural networks (Arjovsky, Shah, and Bengio 2016; Helfrich, Willmott, and Ye 2018; Lezcano-Casado and Martínez-Rubio 2019), NNs with structured linear layers via matrix factorisations such as the QR or the SVD decomposition (J. Zhang, Lei, and Dhillon 2018; Kingma and Dhariwal 2018), or invertible matrices in normalising flows (Berg et al. 2018) and VAEs (Tomczak and Welling 2016).

In parallel, during the last 10 years, neural networks have revolutionised many fields such as speech recognition (Hinton et al. 2012; Saon et al. 2017), image recognition (Krizhevsky, Sutskever, and Hinton 2012; Y. Huang et al. 2019), and natural language processing (Collobert et al. 2011; Ott et al. 2018). Initially, neural networks were a concatenation of linear layers, but with time, these architectures have been getting more and more complex with the introduction of architectures such as attention mechanisms (Bahdanau, Cho, and Bengio 2015; Vaswani et al. 2017), neural Turing machines (Graves, Wayne, and Danihelka

2014), variational auto encoders (VAE) (Kingma and Welling 2014) and generative adversarial networks (GAN) (Goodfellow, Pouget-Abadie, et al. 2014).

This trend of building larger models on more advanced hardware, such as GPUs and TPUs has gone hand in hand with the development of large linear algebra libraries such as PyTorch, Tensorflow, or JAX. These libraries are similar to Numpy or Scipy, but also come with full GPU and TPU support, an autodifferentiation engine, support for automatic mixed-precision (AMP), and many other capabilities to help develop modern high-dimensional deep learning models that scale to datasets with millions of datapoints.

At the time of writing, there is no library to put these two trends together in a practical way. The current libraries that implement optimisation on manifolds techniques are designed for researchers, not for practitioners. For this reason, the thesis aims to fulfil the following two objectives:

GeoTorch: A library for optimisation on manifolds at scale We design and implement a Python library to help the non-expert using all these powerful tools in a way that is efficient, extensible, and simple to incorporate into the workflow of the data scientist, practitioner, and applied researcher. The algorithms implemented in this library have been designed with usability and GPU efficiency in mind, and they can be added to any PyTorch model with just one extra line of code.

Orthogonal RNNs We showcase the effectiveness of these tools on an application of optimisation on manifolds in the setting of time series analysis. In this setting, orthogonal and unitary optimisation is used to constraint and regularise recurrent models and avoid *vanishing* and *exploding gradient* problems. The algorithms designed for GeoTorch allow us to achieve state of the art results in the standard tests for this family of models.

1.1.2 A theoretical reason

The current approach to optimisation on manifolds rarely makes use of the abstract approach, favouring the embedded setting. This seems like a natural thing to do, as the manifolds of interest in Riemannian optimisation are, by definitions, manifolds embedded in $\mathbb{R}^{n\times k}$—*i.e.*, matrix manifolds. When one has a matrix manifold, it is natural to use the structure of the manifold as a manifold embedded in a linear space and identify points on the manifold, vectors, actions by a (matrix) Lie group, isometries and similar objects as matrices or linear functions. This approach often puts aside the entire abstract toolbox from modern differential geometry, like fibred manifolds, group actions, principal connections, or Riemannian homogeneous spaces, in favour of matrix manipulations.

Using matrices and algebraic manipulations rather than sections of the different associated bundles and group actions often hides the actual symmetries that make these manifolds tractable. It is these symmetries that allow us to find closed-form solutions to some differential equations on the manifold. Cartan made a similar point when talking about the abstract index formalism:

> The great opportunities that the *absolute differential calculus* of Ricci and Levi-Civita has given us must not prevent us from avoiding purely formal calculations, where the profusion[1] of indices masks an often very simple geometric reality. It is this reality that I set out to highlight here. (Original in French[2]) (Cartan 1946)

Identifying geometric objects on the manifold with matrices simplifies the background necessary to check whether a proof is true, but it often hides the true reasons why the proof is true. Drawing similarities with type systems in computer science, we can see that while it is true that dropping the types of a correct program does not make it incorrect, it certainly makes it more difficult to reason about. The same happens if we simply write points on the manifold or sections on the tangent bundle of the manifold as matrices: It hides the geometric meaning from these operations.

In this sense, this more abstract approach, whenever applicable, can be used to explain the nature of some numerical computations. In particular, in the field of constrained optimisation, when the set of constraints is a Riemannian manifold with enough symmetries, we are in the best of scenarios as we can use all the tool set from differential, Riemannian, and comparison geometry to explain the behaviour of our algorithms. In this sense, this helps us in getting closer to the goal already set by Hamming in his book on numerical methods

> The purpose of computing is insight, not numbers. (Hamming 1962)

With these points in mind, in this thesis we will use the language of differential geometry for studying the field of optimisation on manifolds. In particular, we will explore the following two research directions:

Use abstract differential geometry to study optimisation problems over matrices We use the more abstract language of global differential geometry and group actions to prove convergence bounds on certain optimisation problems on manifolds, postponing the instantiation of the results on concrete manifolds as much as possible.

In doing so, we follow the approach started by Smith in his PhD thesis (Smith 1993), which was written aimed for a numerical analysis audience in the influential paper (Edelman, Arias, and Smith 1998).

Use comparison geometry to study optimisation problems on Riemannian manifolds We use tools from comparison geometry to give bounds on quantities that are of interest in optimisation problems. In particular, we build on the work of (Kaul 1976) to give explicit bounds on the norm of the second derivative of the Riemannian exponential.

[1] It is of historical interest to note that this expression was popularised in English by Spivak, who jokingly translated it as "debauch of indices" in the first volume of his treaty on differential geometry. The word "débauches" in French has the same meaning as "debauch" in English in almost any situation, being this one of the few exceptions in which it simply means "profusion" with no extra connotation.

[2] Les services éminents qu'a rendus et que rendra encore le Calcul différentiel absolu de Ricci et Levi-Civita ne doivent pas nous empêcher d'éviter les calculs trop exclusivement formels, où les débauches d'indices masquent une réalité géométrique souvent très simple. C'est cette réalité que j'ai cherché à mettre partout en évidence.

1.2 Contributions and Outline

This thesis puts together three fields that are not usually found together. These are global differential geometry, optimisation on manifolds, and deep learning. Researchers in each of these fields approach problems lying at the intersection with completely different tools, notation, and goals in mind. It is for this reason that we include at the beginning of the thesis two introductory chapters, one on classical optimisation, machine learning, and their ties to deep learning (Chapter 2), and another one on differential geometry from a global perspective (Chapter 3). All the results in these chapters are well-known to researchers from the fields of optimisation, machine learning and/or differential and Riemannian geometry, so it is only the exposition and comments throughout it that are original.

In the last three chapters, we present the research carried by the author during these last years. All the results in these chapters are original except when specified otherwise. When a section is well-known to geometers, we carefully announce so either at the beginning of each section or before each result so that it is perfectly clear what is classic and what is new.

Chapter 4. Its main theme is optimisation on manifolds, and its secondary theme is differential and Riemannian geometry. In this chapter, we study how the **theory of pullbacks along submersions** is a fruitful one (Section 4.3). This theory helps to simplify problems in optimisation from spaces with a difficult geometry to simpler ones. We also show how to use the theory laid out in Chapter 3 to **prove second order bounds on the pullback of a function by a Riemannian submersion** (Theorem 4.3.1 and Proposition 4.3.2). We then go and look at the special case when we pullback an optimisation problem to a Euclidean space via a retraction. We call this the **static trivialisation framework**. We make special emphasis on the case when we use the Riemannian exponential map to pullback the problem to a tangent space (Section 4.4.2), as this will be the main tool studied and used in Chapter 5 and implemented in Chapter 6. At the end, we introduce the **dynamic trivialisation framework** (Algorithm 1), which sees the Riemannian exponential as a map from TM to M and uses the vector bundle structure of TM to solve the shortcomings of static trivialisations. These ideas were was presented at ICML 2019 and NeurIPS 2019 (Lezcano-Casado and Martínez-Rubio 2019; Lezcano-Casado 2019).

Chapter 5. Its main theme is Riemannian and comparison geometry, and its secondary theme is optimisation on manifolds. In this chapter, we give bounds on $\|\mathrm{Hess}(f \circ \exp_p)\|$ in terms of first and second order bounds on f and the curvature of the Riemannian manifold (M, g). We start the chapter by doing a literature review of previous approaches to the subject, where we show how these bounds can be readily used in a number of papers that use second order bounds on retractions. We then offer a self-contained proof of Rauch's theorem, as we heavily rely on it throughout the second part of the chapter, and it is not very well-known in the field of optimisation. We also look at its implications in the area of convex optimisation on manifolds at the end of the section. After this, we use these results and a few more to develop novel **second order bounds on the Riemannian exponential** (Theorems 5.4.13 and 5.4.14). From the first and second order bounds, we give **conditional convergence rates for SGD on the function** $f \circ \exp_p$ (Theorem 5.6.2), and the equivalent result for the dynamic trivialisation algorithm (Theorem 5.6.4). The second order bounds and their application to get convergence rates for

static and dynamic trivialisations are the main theoretical results of the thesis. We finish this chapter with a short **characterisation result for retractions that are Riemannian exponential maps** for some connection in terms of sprays on the second tangent bundle TTM (Theorem 5.7.1).

Chapter 6. Its main themes are practical optimisation on manifolds and deep learning. In this chapter, we explain how to implement the ideas of static and dynamic trivialisations that were laid out in Chapter 4. The exposition in this chapter is done by developing the example of optimisation with orthogonal constraints. We then present a literature review of the current state of the art of practical optimisation on manifolds. After this, we go on to describe the implementation details of library **GeoTorch** that we have developed to perform scalable optimisation on manifolds in large models (Section 6.2.2). We finish this chapter and the thesis showing a number of experiments to showcase the practical efficiency of dynamic trivialisations in the context of orthogonal RNNs(Sections 6.3 and 6.4).

GeoTorch is available under MIT license at

`https://github.com/Lezcano/geotorch`

The ideas in this chapter have been submitted to core PyTorch and have been accepted to be part of PyTorch 1.9.0. The work in Section 6.3.2 led to the addition of the matrix exponential into core PyTorch by Nikita Vedeneev. This implementation yielded a $\times 12$ speed-up over the previous fastest GPU implementation.

Chapter 2

Machine Learning and Neural Networks

In this chapter, we aim to contextualise the field of optimisation on manifolds and its applications to neural networks within the broader field of machine learning. As we will see, the field of theoretical machine learning draws from fields as disparate as probability in Banach spaces, optimisation in Hilbert spaces, computer science, and engineering. This makes it difficult to find consistent terminology for this field, where some concepts have been rediscovered several times during the last 50 years. As such, this chapter has an introductory as well as a unifying character, giving a short introduction to all these ideas, their development throughout history and the kind of results that one encounters in each relevant field, while also introducing the nomenclature used in the fields of machine learning and deep learning, which will be used throughout the rest of the thesis. This is done in the hope that readers coming from other fields are not left with a text full of foreign terminology without a proper dictionary.

All the theoretical ideas in this chapter and many more can be found in the books (Vapnik 1995; Nesterov 2004).

Outline of the chapter. We will start by giving a short motivation for the ideas that underlay the current theoretical approach to offline machine learning, with the usual split between generalisation and optimisation. Then, we will provide a historical review of some classical results in the field of statistical learning theory and optimisation in $\mathbb{R}^n$, to serve as an introduction and contextualisation to the convergence results that we will give in the thesis. After that, we will give specific examples of a popular family of models called neural networks, and we will show how to implement them in the **PyTorch** library. Throughout the course of this introduction to the practical side of machine learning, we will make sure to always draw connections to the theory outlined before. We hope this connection between theory and practice helps to put in context the ideas that will be introduced in the thesis, where practical remarks will often be interleaved with more theoretical ones to give algorithms that are solid on both ends.

2.1 Offline Statistical Learning

Before introducing what neural networks are, we shall first present the standard framework for offline statistical learning in its most basic setting.

2.1.1 Framework and terminology

Features and labels. Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets. We say that $\mathcal{X}$ is the **feature space** and $\mathcal{Y}$ is the **label space**. For example, if we wanted to model whether the income of the parents affects whether their children end up studying at Oxford, we would have $\mathcal{X} \cong \mathbb{R}_{\geq 0}$ as the set of all possible incomes (for example in pounds), and $\mathcal{Y} \cong \{0, 1\}$ which models whether certain children of theirs studied at Oxford or not.[1][2] In general, we could have $\mathcal{X}$ and $\mathcal{Y}$ to be any pair of sets, discrete or continuous, without any structure.

Distribution and training set. We assume that there exists a random variable (X, Y) on $\mathcal{X} \times \mathcal{Y}$ that we want to study. We do not have direct access to (X, Y), but we can sample from it. One convenient way of thinking of the random variable (X, Y) is as thinking that there exists certain random variable X on $\mathcal{X}$ which induces a conditional one $Y \mid X$ on $\mathcal{Y}$.

In the context of offline statistical learning, we assume that we have access to m i.i.d. samples $(X_1, Y_1), \dots, (X_m, Y_m) \in \mathcal{X} \times \mathcal{Y}$ from (X, Y). We call these samples the **training set**, and we assume m to be fixed a priori. This is where offline learning differs from online learning. In online learning, we assume that we have access to a sampling oracle that gives us i.i.d. samples of (X, Y) on demand, so our algorithm may adapt as we get more and more samples.[3]

In the example of the acceptance at Oxford, the distribution (X, Y) would be the actual unknown distribution, having certain distribution on the income $\mathcal{X}$ and certain probability of being accepted at Oxford given certain fixed income $Y \mid X$. The training set could be m samples that we collect via a survey.

Model and loss function. In order to model $Y \mid X$, we consider a family of functions $f_\theta \colon \mathcal{X} \to \mathcal{Y}$ indexed by some parameter $\theta \in \Theta$. Usually Θ will be a Euclidean space $\mathbb{R}^n$ for some $n > 0$, but in more general settings, like those that we will consider in this thesis, Θ can be a manifold or a more general topological space. The family of functions $\{f_\theta \mid \theta \in \Theta\}$ is called the **model**. The last thing that we will need, is a **loss function** $\ell \colon \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$, which serves as a proxy for a distance on $\mathcal{Y}$, although it need not be symmetric—and often it is not.

Following the previous example of admissions to Oxford, we would have to model the binary output $\mathcal{Y} = \{0, 1\}$. To do so, one usually considers the relaxation $\mathcal{Y} \cong \{0, 1\} \subseteq [0, 1]$. An example of a model in this case could be what is called **polynomial linear regression of degree** n for a fixed $n > 0$ together with a mapping $\sigma \colon \mathbb{R} \to [0, 1]$. This model can be mathematically described setting $\Theta \cong \mathbb{R}^{n+1}$ and

$$\begin{aligned} f_\theta \colon \mathcal{X} \times \mathbb{R}^{n+1} &\to [0, 1] \\ (x, \theta) &\mapsto f_\theta(x) = \sigma\Big(\sum_{k=0}^{n} \theta_k x^k\Big) \end{aligned}$$

[1] The use of $\cong$ instead of $=$ here has a bit of a pedantic nature. Note that the set of all weights $\mathcal{X}$ *is not* intrinsically equal to the abstract set $\mathbb{R}_{\geq 0}$ but it is merely identified mathematically with it. Hence, $\mathcal{X}$ and $\mathbb{R}_{\geq 0}$ are isomorphic, and not equal.

[2] The model and methodology presented here is an over-simplification which would not be of much use in the real world. In a real model, one would compare many other variables, together with a much more complex analysis than the one that we will present here.

[3] Often these m samples would be split into **training set**, **test set** and **validation set**. Although this practice is fundamental in practice, we will not talk about it here to simplify this introduction.

and we could choose $\text{sigm}(x) \coloneqq \frac{1}{1+\exp(-x)}$ as the mapping. This is called the **sigmoid function**.[4]

In classification models, it is customary to think of the map f_θ with image contained in $[0,1]$ as a random variable on $[0,1]$ modelling $Y|X$. As such, the usual loss function is the **cross-entropy** between the distribution induced by f_θ and the empirical distribution

$$\ell(x,y) = \begin{cases} \log(x) & \text{if } y = 1 \\ \log(1-x) & \text{if } y = 0 \end{cases}$$

Note that, in this case, y is assumed to be in $\{0,1\}$, while x is assumed to be in $(0,1)$, which is the image of f_θ.

Remark 2.1.1 (Modelling a problem). The choice of the model and the loss function heavily depends on the problem at hand, and currently, it is closer to an art than a science. For this reason, we will not explore these further theoretically, as the scope of doing so is limited. We will come back to them in the experiment sections, when we implement certain models to benchmark the algorithms that we will propose against those in the literature.

Expected and empirical risk. With these objects in hand, we can define the **expected risk** or **population risk** as

$$r(\theta) \coloneqq \mathbb{E}[\ell(f_\theta(X), Y)].$$

We would then like to approximate a solution to the **expected risk problem**

$$\arg\min_{\theta\in\Theta} \mathbb{E}[\ell(f_\theta(X), Y)]. \tag{2.1}$$

Of course, we cannot directly compute $r(\theta)$ as we do not have direct access to the distribution of (X,Y), but what we can do is to estimate it via the **empirical risk**

$$R_m(\theta) \coloneqq \frac{1}{m}\sum_{i=1}^{m} \ell(f_\theta(X_i), Y_i).$$

We can compute the empirical risk, as opposed to the population risk. It also has the desirable property of being an unbiased consistent estimator of $r(\theta)$. Using this estimator, we can define the **empirical risk minimisation objective** (ERM)

$$\min_{\theta\in\Theta} \frac{1}{m}\sum_{i=1}^{m} \ell(f_\theta(X_i), Y_i). \tag{2.2}$$

Denote an **expected optimal value** for the expected risk problem and an **empirical optimal value** for the empirical risk problem by

$$\theta^* \in \arg\min_{\theta\in\Theta} r(\theta) \qquad \psi^* \in \arg\min_{\theta\in\Theta} R_m(\theta).$$

In general, we will not be able to compute ψ^* exactly. We will use some optimisation method that gives a sequence of estimates ψ_t given by some optimisation algorithm. Putting all these quantities together, we have the so-called **statistics-optimisation trade-off**.

[4]The image of the sigmoid function is $(0,1)$ rather than the full $[0,1]$, which seems in contradiction with the fact that we are interested in modelling a random variable with values on the ends of $[0,1]$. This is done due to convenience, as $(0,1)$ is diffeomorphic to $\mathbb{R}$ (through the sigmoid function, for example), while $[0,1]$ is not even homeomorphic to $\mathbb{R}$.

Proposition 2.1.2 (Statistics-optimisation trade-off). *We have the following factorisation:*

$$r(\psi_t) - r(\theta^*) \leq \underbrace{\sup_{\theta\in\Theta}(r(\theta) - R_m(\theta)) + \sup_{\theta\in\Theta}(R_m(\theta) - r(\theta))}_{\textit{Statistical term}} + \underbrace{R_m(\psi_t) - R_m(\psi^*)}_{\textit{Optimisation term}}. \tag{2.3}$$

Proof. Interpolating we have

$$r(\psi_t) - r(\theta^*) = (r(\psi_t) - R_m(\psi_t)) + (R_m(\psi_t) - R_m(\psi^*)) + (R_m(\psi^*) - R_m(\theta^*)) + (R_m(\theta^*) - r(\theta^*)).$$

Noting that the term in the third parenthesis is non-positive and taking suprema we get the result. □

The area of **statistical learning theory** studies how to give bounds for the statistical term for different classes of models,[5] and tries to give bounds on the number of samples m needed to make the statistical term smaller than certain given quantity ε. Given that the samples (X_i, Y_i) are stochastic, the results in this field are given as results **with high probability**, that is, one proves that the statistical term is less than ε with probability at least $1 - \delta$, where $\varepsilon, \delta > 0$ are some parameters given a priori. The abstract idea behind these type of results is that of **concentration of measure**, quantifying for example the rate of convergence of certain central limit theorem. These usually come from fine high-dimensional versions of Markov's and Chebyshev's inequalities and other tail inequalities.

On the other hand, the area of **stochastic optimisation** studies how to design algorithms to approximate solutions to the problem in (2.2). Given an algorithm, one proves the convergence of ψ_t as a function of the number of steps t to a minimiser ψ^*. One often assumes some notion of convexity or some degree of regularity of the derivatives of the function being minimised—$R_m(\theta)$ in this case—to be able to prove this kind of results.

2.1.2 A note on statistical learning theory

The statistical term is studied in the **statistical learning theory**, often also referred to as **generalisation theory**, given that it studies how well an empirical optimum—for example $R_m(\psi^*)$ or any other estimator—approximates $r(\theta^*)$. In this approach, the quantity $R_m(\theta)$ is treated as random variable that depends on $(X_1, Y_1), \ldots, (X_m, Y_m)$.

Dimensionless bounds. In practice, machine learning models are larger and larger. In particular, the parameter space Θ can have a dimension in the order of the billions. As such, the bounds that this theory hopes to get are **dimensionless**, that is, independent of the dimension of Θ.[6] For this reason, this theory is based in the field of **high dimensional probability**. When one is able to get dimensionless bounds, it tends to be the case that the proofs and results can be adapted to work in Hilbert and Banach spaces, and as such, it should be no surprise that this theory is itself based in the theory of **probability in Banach spaces**. The reference book in this field is Ledoux and Talagrand's (Ledoux and Talagrand 1991).

[5] More generally, statistical learning theory tries to bound $r(\psi)$ for an arbitrary point $\psi \in \Theta$. The point does not need to come from ERM, and the bounds may be different to the ones presented here. That being said, this is by far the most used approach.

[6] Note that this need not hold on the optimisation term. When computing bounds for the optimisation term later on, we will often put them in terms of the Lipschitz constant. This Lipschitz constant often depends on the dimension of the problem.

Modelling the generalisation term. The approach to get bounds on the statistical term in (2.3) comes from assuming a metric space (Θ, d) on the parameter space and, for a fixed $m > 0$, considering the stochastic process $X_\theta = r(\theta) - R_m(\theta)$ that depends on the random variables $(X, Y), (X_1, Y_1), \dots, (X_m, Y_m)$. The problem can be then translated to bounding above and below the quantity

$$\mathbb{E} \sup_{\theta \in \Theta} X_\theta.$$

Generalising the problem by forgetting the particular structure of X_θ, one looks to give upper and lower bounds on the expectation of a general stochastic process X_θ on a metric space (Θ, d).

The first result in this direction was the **chaining argument**. It was first developed by Kolmogorov in 1934 but never published (Chentsov 1956), and it was then generalised by Dudley (Dudley 1967), who proved **Dudley's entropy integral** under the assumption that X_θ is subgaussian, giving the upper bound

$$\mathbb{E} \sup_{\theta \in \Theta} X_\theta \leq 12 \int_0^\infty \sqrt{\log N(\Theta, d, \varepsilon)}\, \mathrm{d}\varepsilon$$

where $N(\Theta, d, \varepsilon)$ is the **ε-covering number** of the metric space (Θ, d), that is, the minimum number of balls of radius ε needed to cover the whole Θ.

Lower bounds on this quantity were given in the theory of Gaussian processes by Slepian, Fernike, and Sudakov. These results assume X_θ to be a Gaussian process and consider the **natural metric** induced by it in Θ, which is the L^2 metric

$$d(\theta_1, \theta_2) = \|X_{\theta_1} - X_{\theta_2}\|_2 = \mathbb{E}[|X_{\theta_1} - X_{\theta_2}|^2]^{1/2}.$$

Using this distance, a Gaussian process is, by definition, also subgaussian under this metric. These natural metrics allow giving not only lower bounds, but actually matching upper and lower bounds. This was done by Talagrand in the celebrated **majorising measure theorem** (Talagrand 1987; Talagrand 1992; Talagrand 1994). This result is quite technical, as it involves some fairly complex constructions on metric spaces. Intuitively, these structures can be seen as a refinement of the global quantity $N(\Theta, d, \varepsilon)$ to be rather a finer local one.

Going back to statistical learning theory, one of the most basic models of learning to consider is the set

$$\Theta = \{(-\infty, x] \mid x \in \mathbb{R}\}$$

and f_θ to be the indicator function on the interval $\theta \in \Theta$. A natural metric for these functions is the L^∞ metric, as it is a metric that makes the empirical measure $\mu_m = \sum_{i=1}^m \delta_{X_i}$ subgaussian. On the other hand, it is direct to see that the covering numbers $N(\Theta, \|-\|_\infty, \varepsilon) = \infty$ for any ε. As such, all the previous bounds become trivial for the case of infinite sets of indicator functions. However, it is possible to give finite bounds for these functions via a process called **symmetrisation**, which can be summarised in the following chain of inequalities

$$\mathbb{E}\left[\sup_{\theta \in \Theta} \sum_{i=1}^m (f_\theta(X_i) - \mathbb{E}\, f_\theta)\right] \leq 2\, \mathbb{E}\left[\sup_{\theta \in \Theta} \sum_{i=1}^m \varepsilon_k f(X_i)\right] \leq \sqrt{2\pi}\, \mathbb{E}\left[\sup_{\theta \in \Theta} \sum_{i=1}^m g_k f(X_i)\right]$$

where ε_k are i.i.d. symmetric Bernoulli and g_k are i.i.d. $\mathcal{N}(0, 1)$. The quantities in the first and second bound are called the **Rademacher complexity** of the set of functions Θ and the **Gaussian complexity**

of Θ respectively. These inequalities tell us that bounding any of these two complexities is enough to give bounds on the empirical process, and as such, to give a priori bounds on the generalisation of the class of our class of models. Most of the theory in statistical machine learning centres around giving bounds for these two complexity quantities under different assumptions on the family f_θ.

For a rather in-depth introduction to these and other topics from a probability theory perspective, we strongly recommend the book by Ramon van Handel (Handel 2014). For a book with more applications towards the machine learning see (Vershynin 2018). There is also the classical book by Vapnik which lays the foundations of the field of statistical learning (Vapnik 1995).

To finish this note on the statistical term, we shall give an example of the magnitude of this term for a simple family of classifiers.

Proposition 2.1.3. *Let (X, Y) be a random variable on $\mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^n \times \mathbb{R}$. Let $B_n(r) \subseteq \mathbb{R}^n$ be the ball centred at zero with respect to the ℓ^2 norm. Consider two bounded sets $\Theta \subseteq B_n(r_\Theta)$, $\mathcal{X} \subseteq B_n(r_\mathcal{X})$ for two fixed $r_\Theta, r_\mathcal{X} > 0$. Consider the family of linear predictors $f_\theta(x) = \langle \theta, x \rangle$ parametrised by Θ. For a loss function L-Lipschitz on the first variable, we can bound the statistical term as*

$$\mathbb{E} \sup_{\theta \in \Theta} (R_m(\theta) - r(\theta)) = \mathbb{E}\left[\sup_{\theta \in \Theta} \sum_{i=1}^{m} \ell(\langle \theta, X_i \rangle, Y_i) - \mathbb{E}\, \ell(\langle \theta, X \rangle, Y) \right] \leq 2 \frac{r_\mathcal{X} r_\Theta L}{\sqrt{m}}.$$

Proof. See for example (Shalev-Shwartz and Ben-David 2014, Lemma 26.9 and Lemma 26.10). □

If one further assumes subgaussianity of the distribution of (X, Y), it is then possible to convert these bounds in expectation to bounds **with high probability**, that is, one gets tail bounds on the distribution of $R_m(\theta)$.

Remark 2.1.4 (Qualitative properties of the bound). The first thing to note is that, as we mentioned before, this bound is independent of the dimension n of the parameter space Θ and the input space $\mathcal{X}$.

Secondly, this bound tells us that, in the most basic case, the statistical term in (2.3) scales as $\mathcal{O}(\frac{1}{\sqrt{m}})$. It is also possible to show that this bound is asymptotically tight for this family of classifiers. This bound suggests that it is enough to optimise up to a precision on the order of $\mathcal{O}(\frac{1}{\sqrt{m}})$.

The asymptotics $\mathcal{O}(\frac{1}{\sqrt{m}})$ do not appear just in the case of linear classifiers, but also in many other architectures. In essence, they stem from the convergence rates given by the central limit theorem. These rates are asymptotically tight unless extra assumptions are considered. In typical scenarios, under more restrictive hypotheses, the rate of convergence can be improved to $\mathcal{O}(\frac{1}{m})$, which is usually referred to as **fast rate**.

The take home from these ideas is that it should be just necessary to bound the optimisation term at a rate $\mathcal{O}(\frac{1}{\sqrt{m}})$, as this bound is already imposed by the statistical term in the statistics-optimisation factorisation.

2.2 First Order Optimisation

2.2.1 The complexity model

In the rest of the thesis, we will just look at the optimisation term introduced in the previous section. More generally, we will be interested in studying algorithms to approximate a solution of problems of the

form

$$\min_{x \in \mathsf{X}} f(x)$$

where X is a subset of $\mathbb{R}^n$.[7] Note that the set X is what we called Θ in the context of modelling and machine learning.

As it is customary when developing a complexity theory for a class of algorithms, to be able even talk about what an algorithm is without getting lost in computational semantic considerations, it is necessary to introduce a computational model. A **computational model** in the theory of optimisation abstracts the idea of *what is a function*, as a function in computer science can be defined in many possible ways and a definition of a function may be as concrete as to be language-dependant.

In optimisation, the computational model abstracts a function f in terms of its value and derivatives at its points.

Definition 2.2.1. A **k-th order oracle** $\mathscr{O}$ for a class $C^k(\mathsf{X})$ function $f\colon \mathsf{X} \to \mathbb{R}$ is a mapping that, given a point $x \in \mathsf{X}$, returns the k-th order information of f:

$$\mathscr{O}(x) = (f(x), \nabla f(x), \operatorname{Hess} f(x), \dots, f^{k)}(x)).$$

An algorithm in this model may query the oracle for the value of the function and its derivatives at one or more points. Then, the algorithm will process this information together with the information from previous calls to output a point $x_t \in \mathsf{X}$. This new point will be used to generate more queries to the oracle and so on. We note that, in general, there is no reason for which an algorithm should perform just one call to the oracle in every step. In fact, some very important algorithms require of two calls or even a logarithmic number of calls to the oracle in every step. On the other hand, in this thesis we will mostly consider algorithms that perform one call to the oracle in every step.

In this thesis, we will centre our attention on **first order methods**. These are methods that just have access to the value of the function and its gradient. This is the most common class of methods used in the context of neural networks since if a model has n parameters—$\dim \mathsf{X} = n$—the Hessian would have n^2 entries, which is prohibitively large even for small models. It is worth mentioning that, when implementing second order optimisation, one almost never computes the full Hessian. Instead, algorithms are specified in terms of *Hessian-vector products* of the form $H^{-1}(v)$ where v is a vector and H is the Hessian of f—second order methods—or an approximation of it from first order information—quasi-Newton methods. Even then, computing Hessian-vector products tends to be quite costly when compared with first order methods both in time and memory.

Due to memory limitations given the large models used in deep learning, we will just look at first-order algorithms that do not depend on previous calls to the oracle. An algorithm $\mathcal{A}$ of this form is just a function

$$x_{t+1} = \mathcal{A}(x_t, \nabla f(x_t)).$$

[7] The contributions in this thesis will be in the more abstract setting where X is a differentiable manifold not necessarily embedded in $\mathbb{R}^n$. We will restrict the presentation in this introductory section to subsets of $\mathbb{R}^n$ for the sake of simplicity. It will be in Chapter 4 where we will give an introduction to the subject of optimisation on manifolds together with a literature review of the field in Section 5.1.2, once we have given a presentation of the geometric tools and notation in Chapter 3. Even though we will not mention the manifold setting in this section, we will write most definitions in a coordinate-independent fashion, so that they readily generalise to the manifold setting without much effort.

These algorithms are inherently **local**, as they just process the information at the current point to make their next guess.

Remark 2.2.2 (Optimisation algorithms as discretisations of ODEs). A more formal definition of an algorithm being local comes by interpreting such an algorithm as a discretisation of a differential equation. For example, the gradient step is the forward-Euler discretisation of the differential equation

$$\dot{x} = \nabla f(x).$$

This is an autonomous differential equation which gives rise to a flow called the **gradient flow**. Since this is an invariant equation, this view can be generalised to Riemannian manifolds provided a suitable discretisation scheme. This more general definition of local algorithms allows for a constant number of calls to the oracle to be used in each step, rather than just one. The perspective of looking at the algorithm $\mathcal{A}$ as an integrator has yielded a unified way of looking at large families of optimisation algorithms under a common lens (Wibisono, Wilson, and Jordan 2016; Wilson 2018; França, Jordan, and Vidal 2020).

Remark 2.2.3 (A note on stochastic optimisation). As we mentioned in the previous section, in machine learning, the objective function often depends on an unknown random variable ξ so that the objective function takes the form[8]

$$f(x) = \mathbb{E}[f(x, \xi)].$$

In the offline learning setting, the random variable ξ has finite support and represents the dataset so that $f(x) = R_m(x)$.

The field that studies functions with this structure is called **stochastic optimisation**, and it was introduced in the seminal paper by Robbins and Monroe (Robbins and Monro 1951). Stochastic optimisation has become particularly popular in the last decade due to its applications in machine learning.

The simplest stochastic algorithm approximates the gradient of $\mathbb{E}[f(x, \xi)]$ using B i.i.d. samples $\xi_i \sim \xi$ to form a stochastic approximation to the direction of steepest descent of f

$$x_{t+1} = x_t - \frac{\eta}{B} \sum_{i=1}^{B} \nabla f(x_t, \xi_i). \tag{2.4}$$

This is called **stochastic gradient descent** (SGD). The samples ξ_i are called the **minibatch**—or sometimes simply the **batch**—and the number B is called the **minibatch size**. Many other algorithms have been developed in recent years for the stochastic optimisation in the online and offline setting, yielding algorithms widely used in practice such as **ADAGRAD** (Duchi, Hazan, and Y. Singer 2011), and its heuristic variations **RMSPROP** (Hinton et al. 2012) and **ADAM** (Kingma and Ba 2015). In parallel, the field of stochastic optimisation has grown in its own directions, for example by using this framework to set online decision problems (Bubeck and Cesa-Bianchi 2012).

In the rest of the thesis, we will develop the theory in the non-stochastic setting. On the other hand, the results that we will present are quite general, so it should not be too difficult to generalise the ideas to the stochastic setting. That being said, we will see in the experiments section that the theory developed here works surprisingly well in practice in the stochastic setting. We leave the theoretical development of these ideas open for future research.

[8] The random variable ξ represents in the offline setting the product random variable of the dataset $(X_1, Y_1), \ldots, (X_m, Y_m)$.

Having introduced the abstract computational model, we will spend the rest of the section introducing and giving examples of the two main classes of functions considered in the optimisation literature.

2.2.2 Convex optimisation

Convex optimisation studies the setting in which the function f is convex and the set of constraints X is a convex subset of $\mathbb{R}^n$. This is the most classical scenario as there are many interesting problems in practice that are either convex or can be relaxed into a convex problem, while still reflecting most of the initial structure of the problem.

The convexity hypothesis is a quite restrictive one, but at the same time, it allows applying local algorithms to solve global problems over the whole space X. This comes from the fact that for a convex function on a convex space, every local minimiser is a global minimiser. Furthermore, if the function is strictly convex, there is a unique global minimiser. Convexity (resp. strict convexity) is a local property when the function is of class $C^2(\mathsf{X})$—it is equivalent to $\operatorname{Hess} f(x) \succeq 0$ (resp. $\operatorname{Hess} f(x) \succ 0$) at every point $x \in \mathsf{X}$—so it makes tackling the global problem of optimisation by local methods feasible.

It would be rather naïve to try to give a comprehensive introduction to convex optimisation in a couple of pages given the vast literature on the subject. For example, when the function is linear or quadratic, one enters the fields of linear programming, conic programming and semidefinite programming among others, which study algorithms such as interior point methods and simplex methods, each of which decomposes in many other subfamilies. To avoid falling down that rabbit hole, we will just mention methods that can be applied in practice to general non-linear functions, even if we are just able to prove convergence for a subfamily of convex functions.

Furthermore, to simplify the exposition, we will assume that f is at least of class $C^2(\mathsf{X})$. Virtually all the theory in convex optimisation can be reformulated without this hypothesis via the use of subgradients. On the other hand, the $C^2(\mathsf{X})$ regularity assumption often clarifies the exposition, and proofs in the $C^2(\mathsf{X})$ setting can often be generalised to the convex non-differentiable setting via standard algebraic arguments.

To exemplify the kind of results that one gets in this field, let us consider the simplest of the algorithms: Gradient descent. **Gradient descent** (GD)—sometimes called **steepest descent**—accounts for following the direction of steepest descent for a time $\eta > 0$. Starting at a point $x_0 \in \mathsf{X}$, the update rule is given by

$$x_{t+1} = x_t - \eta \nabla f(x_t).$$

The quantity η is usually called the **step-size**, although in machine learning is often called the **learning-rate**. Now, this begs the question: What is a reasonable step-size?

In the same way that one does when giving bounds on the discretisation term of an ODE, to answer this question one has to put some assumptions on the function. The simplest assumption is that of smooth convexity.

Definition 2.2.4. A function $f \in C^2(\mathsf{X})$ is said to be of **α-bounded hessian** if

$$-\alpha \mathrm{I}_n \preceq \operatorname{Hess} f(x) \preceq \alpha \mathrm{I}_n \qquad \forall x \in \mathsf{X}.$$

If the function f is also convex (*i.e.*, $0 \preceq \operatorname{Hess} f(x)$), f is said to be **α-smooth**.

In other words, a C^2 function is of α-bounded Hessian if the eigenvalues of its Hessian lie in the interval $[-\alpha, \alpha]$, and it is α-smooth if they lie in $[0, \alpha]$.

Expanding the function as a Taylor series along the straight line that connects two points $x, y \in \mathsf{X}$ and bounding Lagrange's remainder we get the following bound around $x \in \mathsf{X}$ for a function of α-bounded Hessian

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2}\|x - y\|^2. \tag{2.5}$$

Denote $\gamma(t) = x + t(y - x)$ for $t \in [0, 1]$ the line connecting x and y. This bound can be interpreted geometrically as having an upper bound on $f|_\gamma$ by a quadratic function tangent to f at x with leading term $\frac{\alpha}{2}$. Note that this bound is everywhere sharp for the distance function to a fixed point $f(x) = \frac{\alpha}{2}d(x, x_0)^2$ as this is a quadratic function with Hessian equal to $\alpha \cdot \mathrm{Id}$. For this function, in the unconstrained case $\mathsf{X} = \mathbb{R}^n$, it is possible to get to the minimum x_0 from any point in one gradient descent step letting $\eta = \frac{1}{\alpha}$. Algebraically, we can see all these ideas by setting $y = x_{t+1} = x_t - \frac{1}{\alpha}\nabla f(x_t)$ in (2.5) to get

$$f(x_t) - f(x_{t+1}) \geq \frac{1}{2\alpha}\|\nabla f(x_t)\|^2. \tag{2.6}$$

Geometrically, this means that we improve the value of the function at least by $\frac{1}{2\alpha}\|\nabla f(x_t)\|^2$ after every step. On the other hand, by convexity and Cauchy-Schwartz, one may bound below the norm of the gradient by

$$\frac{f(x_t) - f(x^*)}{\|x_t - x^*\|} \leq \|\nabla f(x_t)\|.$$

Plugging this into (2.6), putting the resulting equation in terms of $\delta_t = f(x_t) - f(x^*)$ and showing that $\|x_t - x^*\|$ is decreasing in t, one gets the inequality

$$\delta_{t+1} \leq \delta_t - \frac{1}{2\alpha\|x_0 - x^*\|^2}\delta_t^2.$$

and bounding the general term of this recurrence, the following full convergence result follows.

Theorem 2.2.5. *Assume that $\mathsf{X} = \mathbb{R}^n$ and f is α-smooth and has a minimiser x^*. Then, gradient descent with step $\eta = \frac{1}{\alpha}$ starting at a point x_0 converges as*

$$f(x_t) - f(x^*) \leq \frac{2\alpha\|x_0 - x^*\|^2}{t + 4}.$$

This result says that the optimisation term in the setting of convex α-smooth functions decays as $\mathcal{O}(\frac{1}{t})$. A much more impressive result is the one developed by Nesterov in 1983. Nesterov proposed an algorithm such that, under the same α-smoothness hypothesis, converged at a rate of $\mathcal{O}(\frac{1}{t^2})$. This was a major result, as a few years before it had been proven that this rate was optimal for the set of convex α-smooth functions. After that result by Nesterov, many other algorithms that converge at this fast rate have been developed. In general, an algorithm that converges faster than vanilla gradient descent for a class of functions is called an **accelerated algorithm**.

This type of results gives us a **worst-case analysis** of the optimisation oracle, as one may reword this as "gradient descent converges to a point with a value at most ε away of the minimum in at most $\mathcal{O}(\frac{1}{\varepsilon})$ steps".

Of course one can easily generalise this algorithm to one where $\mathsf{X} \subseteq \mathbb{R}^n$ is a proper convex subset, provided that we have an efficient way to project from $\mathbb{R}^n$ onto X. This is called **projected gradient descent**. In this case, a similar convergence analysis gives the same rates as in the unconstrained case.

There are many other classes that are studied in the setting of convex optimisation, such as **α-strongly convex** functions ($\operatorname{Hess} f \succeq \alpha \mathrm{I}_n$ for $\alpha > 0$), Lipschitz functions and combinations of these. Another line of research is that of studying convergence under weaker regularity assumptions than differentiability. In this case, one of the most commonly known algorithm given its versatility is Nemirovsky and Yudin's **mirror descent** (Nemirovsky and Yudin 1983), which may be applied in the more general setting of Banach spaces. In general, one can find a vast zoo of algorithms with convergence guarantees in convex optimisation. To name a few, by replacing the penalty function in mirror descent by a non-linear function one gets **proximal methods**. If one can do linear optimisation over the domain, one may use **Frank-Wolfe**. In **coordinate gradient descent** and **block-coordinate gradient descent** one optimises one or more of the coordinates of the function at a time, leaving the others fixed during that iteration. In **alternated direction method of multipliers (ADMM)** for constrained optimisation one tries to approximate a solution to equation given by the Lagrange multipliers iteratively updating the multipliers... For a theoretical introduction to all these methods and others see (Nesterov 2004; Boyd and Vandenberghe 2004).

2.2.3 Non-convex optimisation

In the setting of non-convex optimisation, one drops completely the assumption of f being convex. On the one hand, this gives a much wider range of applicability to the results developed by this theory. On the other, the results that this theory is able to prove are much weaker. In particular, in this theory, as the global properties given by convexity are dropped, one is likely to encounter critical points and local minima of f, to the point that, in general, not even finding but checking that a given point is a local minimum for a non-convex function is NP-hard (Murty and Kabadi 1987). For this reason, in this setting, one often has to content oneself with approximating critical points of f, that is, points at which $\nabla f(x) = 0$.

Of course, as it was the case with convex functions, if one wants to get reasonable bounds, it is necessary to restrict the attention to a subset of functions. Again, the class of functions of α-bounded Hessian is suitable for this task. For these functions, we get the following standard result. We prove it here for completeness, as it will be the result that we will generalise to manifold in Chapter 5 (*cf.*, Theorem 5.6.4).

Theorem 2.2.6. *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a function of α-bounded Hessian bounded below by $f^* \in \mathbb{R}$. Gradient descent on f, with a step-size of $\frac{1}{\alpha}$ starting at a point x_0, converges as*

$$\min_{0\leq k\leq t} \|\nabla f(x_k)\| \leq \sqrt{\frac{2\alpha(f(x_0) - f^*)}{(t+1)}}.$$

Proof. By (2.6), we have that for a function of α-bounded Hessian

$$\frac{1}{2\alpha}\|\nabla f(x_k)\|^2 \leq f(x_k) - f(x_{k+1}).$$

Summing over all the indices and telescoping

$$\frac{1}{2\alpha}\sum_{k=0}^{t}\|f(x_k)\|^2 \leq f(x_0) - f(x_{t+1}) \leq f(x_0) - f^*$$

which yields the desired bound. □

In other words, if the function is of α-bounded Hessian, gradient descent with constant step-size finds a point with gradient ε-close in norm to a critical point of f in at most $\mathcal{O}(\frac{1}{\varepsilon^2})$ steps, as opposed to the $\mathcal{O}(\frac{1}{\varepsilon})$ needed to approximate the global minimiser when the function is convex.

Remark 2.2.7 (Projected gradient descent on non-convex functions). It is worth noting that the previous theorem assumes the problem to be unconstrained, that is, $\mathsf{X} = \mathbb{R}^n$. One could hope to extend this proof to work on constrained optimisation problems, as in the convex case. A moment's reflection shows that this is not easily achieved. In fact, non-convex constrained optimisation is notoriously difficult, as the projection does not respect the derivatives in any reasonable way, which destroys the convergence of the algorithm. As such, these two elements do not interact well together.

Even though the proof for gradient descent for non-convex objectives assures that gradient descent will converge to a critical point, this critical point might not be a local minimum, but perhaps a saddle point or a local maximum. On the other hand, via an application of the stable manifold theorem from dynamical systems, it is possible to prove that if the initial point x_0 is sampled from a smooth distribution on $\mathbb{R}^n$, gradient descent will almost always converge in the limit to a local minimiser (Lee et al. 2016), although it may take exponentially many steps in ε (Du et al. 2017). These results inspired many others that tried to estimate Hessian information to escape saddle-points by perturbing the gradient (Allen-Zhu 2018).

In the last 5 years, there has been renewed interest in the field of non-convex optimisation due to the popularity of neural networks. Before that, there had been very little progress since the years 1965–1975 (Nesterov 2004). The new lower-bounds developed for first order (and higher-order) methods on several classes of non-convex functions—assuming certain first and second order bounds—have been of special interest (Cartis, Gould, and Toint 2010; Carmon et al. 2020). In parallel to this work, several accelerated algorithms for non-convex optimisation (Carmon et al. 2017) and algorithms in the stochastic and non-stochastic setting for functions of the form $f(x) = \frac{1}{n}\sum f_i(x)$, like the empirical risk. For a recent review of these results see the papers (Allen-Zhu 2018; Allen-Zhu and Li 2018).

Other directions of research in this area are those coming from convex relaxations, where one is able to find a convex problem that has the same minimiser as the non-convex one, such as those coming from **geometric programming**, many linear programming problems, and some eigenproblems such as the SVD decomposition. Many of these non-convex problems that happen to have a fast way to solve them while being non-convex often have a hidden Riemannian structure that makes them **geodesically convex** under some convenient metric. We will explore in depth this kind of problems in Chapter 4. For now, we will close this review section referencing what is, as the time of this writing, the largest compilation to our knowledge of papers and reviews in the area of provable non-convex optimisation

https://sunju.org/research/nonconvex/

The web-page is active as of 2020, and contains lists of relevant papers written in the last 5–10 years in most of the sub-fields of non-convex optimisation.

2.3 Neural Network Architectures

In this section, we introduce neural networks in the framework of statistical models. We give some examples of different classes of neural networks, and we show how to implement them in the Python library PyTorch. A good introductory reference for all these ideas and many more is the book (Goodfellow, Bengio, and Courville 2016).

2.3.1 Neural networks

Definition 2.3.1. Let $\Theta \subseteq \mathbb{R}^n$ be open and connected. A **neural network architecture** is a mapping $f\colon \Theta \times \mathcal{X} \to \mathcal{Y}$ that is differentiable in the first variable and that factorises into a composition of simpler functions $f^{(i)}\colon \Theta_i \times \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ for $i = 1, \dots, p$ with $\mathcal{X} \cong \mathbb{R}^{d_0}$, $\mathcal{Y} \subseteq \mathbb{R}^{d_p}$ and $\Theta = \Theta_1 \times \cdots \times \Theta_p$

$$f_\theta = f^{(p)}_{\theta_p} \circ \cdots \circ f^{(1)}_{\theta_1}.$$

We will often omit the parameters of the functions and write $f^{(i)} = f^{(i)}_{\theta_i}$. Each parametrised function $f^{(i)}$ is called a **layer**. We say that $f^{(1)}$ is the **input layer**, $f^{(i)}$ for $i = 2, \dots, p-1$ are the **hidden layers**, and $f^{(p)}$ is the **output layer**.[9] A network is said to be **deep** whenever it has three or more layers. **Deep learning** is the area that studies deep neural networks.

Each of the layers in a neural network is designed to be a (parametrised) transformation that can be computed very efficiently. In particular, most of the layers used in practice are designed so that they can be heavily parallelised, making their implementation very fast on a GPU. We will touch on this in more detail later in this section.

The size of deep neural networks has seen an exponential explosion in size in the last ten years, seeing models that have hundreds of layers and up to a hundred billion parameters the current largest one (Brown et al. 2020).

Remark 2.3.2. In most standard neural networks, Θ is just a Euclidean space. As we did in the previous section, we will assume that Θ is isomorphic to a Euclidean space in this section for simplicity, but the reader should bear in mind that the direction of the rest of the thesis will be that of considering Θ to be a connected manifold—often a matrix manifold.

2.3.2 Feedforward networks

The most classical neural network architecture is that of the feedforward network. These are models where each layer is an affine transformation of its input together with a coordinate-wise non-linear transformation.

[9] Sometimes the input and output layers are also called first and last hidden layers.

Definition 2.3.3. Let $\mathcal{X} \cong \mathbb{R}^k$, $\mathcal{Y} \cong \mathbb{R}^d$ and $\Theta \cong \mathbb{R}^{d\times k} \times \mathbb{R}^d$. A **linear layer of width** d is the affine transformation defined by

$$f \colon \Theta \times \mathcal{X} \to \mathcal{Y}$$
$$((A, b), x) \mapsto f_{A,b}(x) = Ax + b$$

We say that A is the matrix of **weights** and b is the **bias**.

Given a function $\sigma \colon \mathbb{R} \to \mathbb{R}$, we define a **non-linearity** as the associated map $\sigma \colon \mathbb{R}^d \to \mathbb{R}^d$ that comes from applying the map σ on vectors in $\mathbb{R}^d$ component-wise. We will denote both maps with the same symbol.

A **feedforward network** is a neural network where every layer is a linear layer followed by a non-linearity, $f^{(i)}_{\theta_i} = \sigma \circ f_{A_i,b_i}$, $\theta_i = (A_i, b_i)$. Usually, the non-linearity in the last layer is chosen to be the identity—*i.e.*, there is no non-linearity in the last layer.

Strengths, weaknesses, and limitations of feedforward networks A feedforward network is one of the simplest neural network architectures used in practice. It is sometimes also referred to as **multi-layer perceptron (MLP)**. The work of Cybenko (Cybenko 1989) and Hornik (Hornik 1991) shows that feedforward networks are **universal approximators**, in the sense that they are dense in the set of continuous functions supported on a compact subset of $\mathbb{R}^k$. On the other hand, an important remark is in order: While these functions may approximate any continuous function arbitrarily well, the width of a shallow network necessary to get an ε-approximator grows exponentially fast in ε (Eldan and Shamir 2016). It is this one of the reasons why feedforward networks are often replaced in practice by more specialised layers that try to encode domain-specific biases. Examples of these are CNNs (Fukushima and Miyake 1982), attention (Bahdanau, Cho, and Bengio 2015), neural machine translation (Kalchbrenner and Blunsom 2013) and transformer models (Vaswani et al. 2017) among many others. More generally, the field of **geometric deep learning** studies how to encode invariances present in the data so that the model structurally preserves them, via the use of equivariant layers (T. S. Cohen and Welling 2016), graph neural networks (Scarselli et al. 2009), and other architectures.

2.3.3 Recurrent neural networks (RNNs)

The simplest non-trivial data topology is that of **sequential data**. This comes up naturally when processing an input with variable length like audio samples, time series, or sentences in a language. In these cases, every element of the sequence is usually encoded into a vector of a fixed dimension, but the number of these vectors is unknown a priori. Using computer science nomenclature, these objects are encoded in the free monoid $(\mathbb{R}^k)^* \coloneqq \oplus_{t=0}^{\infty}(\mathbb{R}^k)^t$. Recurrent Neural Networks were designed to process this kind of sequential data, constituting a map from $(\mathbb{R}^k)^*$ into a finite-dimensional embedding space $\mathbb{R}^d$.

Definition 2.3.4. Given a sequence of inputs $(x_t)_{t=1}^T \in (\mathbb{R}^k)^*$, we define a **recurrent neural network** (RNN) with hidden size $d > 0$ as

$$h_t = \sigma(Bh_{t-1} + Cx_t) \qquad t = 1, \dots, T$$

where $B \in \mathbb{R}^{d\times d}$ and $C \in \mathbb{R}^{d\times k}$ are parameters, σ is some fixed non-linearity, and $h_0 \in \mathbb{R}^d$ is some fixed initial vector. C is called the **input kernel** and B is the **recurrent kernel**. In most applications, the starting vector h_0 is set to zero.

In classification tasks, where one needs an encoding of the whole sequence into a vector, one often chooses either h_T, or $\frac{1}{T}\sum h_t$ as the encoding of the whole sequence in $\mathbb{R}^d$. The problem with these encodings is that they are biased towards the last elements of the sequence, given that these were the elements that were encoded the last. For that reason, one often uses a **bi-directional RNN**, which accounts for running two separate RNNs, one on $(x_t)_{t=1}^T$ and another one on the reversed sequence $(x_{T-t})_{t=1}^T$ and computes the embedding as the sum of the hidden states of the two.

The intermediate vectors h_t are usually regarded as the **memory** of the RNN at time t. Each of the applications of the recurrent step is called a **time-step**, and the dimension t is called the **time dimension**.

RNNs are a rather straightforward generalisation of feedforward neural networks to sequences. On the other hand, they present some problems when the matrix B is not well-conditioned. Consider the degenerate case where $\sigma = \mathrm{Id}$. If B has eigenvalues of norm larger than 1, the gradient of B will explode in some directions for long sequences, while if B has eigenvalues of norm less than 1, the gradients of B with respect to the first time-steps will be close to zero, so the gradient will incorporate almost no information of the initial elements of the sequence. These problems are regarded as **exploding gradient** and **vanishing gradient** problems (Bengio, Simard, and Frasconi 1994). We discuss them in more detail in Section 6.1.

The **Long-Short Term Memory cell** (LSTM) (Hochreiter and Schmidhuber 1997) is a more complex recurrent architecture that tries to heuristically alleviate the vanishing gradient and exploding gradient problems via a system referred to as **gating**. The idea of this heuristic is to *control the flow of the gradients along the LSTM*, by allowing to retain some information in the internal state, while giving mechanisms to delete other information—often thought of as *forgetting*. The gating mechanism is composed of 4 vectors: **input** i, **forget** f, **output**, and **output modulation** g. The LSTM also separates the hidden state into an **internal state** c_t—sometimes referred to as **cell**—and the hidden state h_t. If we denote by $\mathrm{sigm}(t) = \frac{1}{1+e^{-t}}$ the sigmoid function and by tanh the hyperbolic tangent—keeping in mind that the range of these functions are $(0,1)$ and $(-1,1)$ respectively—the LSTM is defined by the equations

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right)$$

$$c_t = f \odot c_{t-1} + i \odot g \qquad h_t = o \odot \tanh(c_t)$$

where $\odot$ denotes the component-wise multiplication and $W \in \mathbb{R}^{4d\times(d+k)}$ is a matrix of parameters. There are other popular variations of this recurrent layer, such as the **Gated Recurrent Unit** (GRU) (Chung et al. 2014), but we will spare the reader their description, as we will not use them in the practical work in this thesis.

2.3.4 Why affine layers?

The layers that we have introduced until now are all variations of the same idea: Applying an affine transformation to the inputs and then apply a component-wise non-linearity. This is not a coincidence. This design comes from engineering considerations arising from the implementation of neural networks. Even though neural networks were first devised in the 80's (Fukushima and Miyake 1982), they were partly forgotten until the first decade of the XXI century (Ciresan et al. 2010; Krizhevsky, Sutskever, and Hinton 2012). Current interest has built upon two large engineering efforts that made neural networks accessible to users outside of academia. These two engineering advances are Graphics Process Units (GPUs) and auto-differentiation engines.

2.3.4.1 Graphics Processing Units (GPUs)

GPUs have been around for many years now, mostly tied to the design and video game industry. It has been just in the last 10 years that we have seen the field of GPU computing soar as a major tool in data analysis, allowing processing large quantities of datapoints in parallel.

A coarse but fairly accurate description of GPUs would come from thinking about them as pieces of hardware that perform operations massively in parallel much more quickly than a CPU. In general, they can apply the same operation in parallel to many datapoints. While a CPU can execute many different operations on individual pieces of data with a very low overhead between operations, a GPU shines at executing the same operation in parallel on many datapoints. We can think of a GPU as having a CPU with a very large number of cores, where the cores are tied to execute the same operation.

If we look at a matrix-vector multiplication, for a matrix of size $d \times k$, this can be decomposed into d inner-products on $\mathbb{R}^k$, each of which can be computed in parallel. Furthermore, each inner-product accounts for adding k different elements, which can be performed in $\log(d)$ time.

At the same time, when performing stochastic gradient descent as described in (2.4), in every step of the algorithm we need to compute the value of the function at B datapoints, where B is the minibatch size. This can also be performed in parallel, taking again advantage of the fast parallel processing given by GPUs. This has allowed neural networks to be able to scale the approximation of the minimisation problem to large datasets, having some of them in the order of 10^8 datapoints (Deng et al. 2009; Kuznetsova et al. 2020; Mahoney 2006).

A word on efficiency and GPUs GPUs provide an efficient way to execute the same operation on many datapoints in parallel. Each of these highly parallel operations is called a **kernel**. Launching a kernel has a much larger overhead than executing an operation on a CPU. For this reason, GPUs shine when reducing the amount of kernels launched during a program. We can think of executing a kernel as having a unitary fixed cost—that of the overhead—when executed on very few points. For this reason, computing a sequence of multiplications of sparse matrices can often be more efficiently implemented on a CPU than on a GPU, due to the lower overhead per operation of the CPU. GPUs also suffer from a penalty coming from memory latency, as each minibatch has to be moved from CPU to GPU, which can result on a computational bottleneck in some applications. The take home from all these ideas is that one has to be

careful when computing the efficiency of an algorithm implemented on a GPU, taking into account the size of the data one is working with and how parallelisable are the operations that are being executed. This latter point will be of particular importance when devising a fast implementation of the methods proposed in this thesis in Section 6.3.2.

2.3.4.2 Auto-differentiation: Forward and backward pass

In the offline learning setting, as we detailed in Section 2.1, we have access to a dataset $\{(X_i, Y_i)\}$ of m pairs of features and labels. As we mentioned before, in deep learning, in many datasets we can have m to be in the order of the millions. Computing the gradient of the empirical risk

$$R_m(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell(f(X_i, \theta), Y_i)$$

would be infeasible, as every step would require to evaluate the function f on millions of points. As an alternative, stochastic gradient descent is used. As we already mentioned, this algorithm chooses in every step a subset of B elements $\{(X_{a_i}, Y_{a_i})\}$ from the dataset to form a minibatch and averages their losses to form

$$\widehat{R}_B(\theta) \coloneqq \frac{1}{B} \sum_{i=1}^{B} \ell(f(X_{a_i}, \theta), Y_{a_i}) \tag{2.7}$$

performing the update

$$\theta \leftarrow \theta - \eta \nabla \widehat{R}_B(\theta). \tag{2.8}$$

In order to implement this algorithm in practice, it is necessary to be able to compute the gradients of the function $\theta \mapsto \ell(f(X_i, \theta), Y_i)$. It was in the seminal work (Ciresan et al. 2010) where it was first documented that this naïve looking algorithm was able to perform astonishingly well at optimising highly non-convex neural networks. Furthermore, in this same work, they showed that this optimisation process could be accelerated several orders of magnitude by implementing it on a GPU. In this first work, the authors implement themselves the necessary GPU kernels. This is a very difficult task, as programming on a GPU is notoriously technical.

In parallel to this work, there had been a line of research that involved the computation of the derivatives of functions automatically (Wengert 1964). These are called **auto-differentiation engines** or simply **autodiff engines**.

It was just then that Theano was born (Al-Rfou et al. 2016). Theano was the first library marrying auto-differentiation and GPUs first released in 2007 as a library for the programming language Lua. An autodiff engine helps the user define differentiable functions: The user writes the function using methods provided by the library and the library will automatically compute its derivative at a given point. In some sense, it implements the chain rule—more formally, the adjoint method—for a wide range of functions and constructions in a programming language. In practice, this is a rather challenging problem.

2.3.5 An implementation of a feedforward network

Let us explain the rudiments of how all the ideas in this section come together via an example written on the PyTorch library, as this will be the library that we will use all throughout Chapter 6.[10]

[10] A disclaimer to the seasoned practitioner: This is intended to be a simple working example to showcase the ideas in this chapter. It does not try to be a performant one by any possible measure.



Figure 2.1: Example of digits present in the MNIST dataset.

Example 2.3.5 (MNIST classification with a feedforward network). Consider a two layer feedforward neural network with $\texttt{relu}(x) := \max(x, 0)$ non-linearities designed to classify greyscale images of handwritten digits. A dataset that provides this exact set-up is that of MNIST (LeCun, Cortes, and Burges 1998). The MNIST dataset consists of 60.000 labelled images of size 28×28 represented by a floating-point number at every pixel describing the intensity of the grey (see Figure 2.1). As such, $\mathcal{X} = \mathbb{R}^{28\times 28}$, $\mathcal{Y} = \{0, 1, \dots, 9\}$.

The model will output a distribution on the set of 10 elements by processing the 10-dimensional output of the last layer with a softmax non-linearity, which maps a 10 dimensional vector into the 10-dimensional simplex by mapping its k-th coordinate to

$$\operatorname{softmax}(x)_k = \frac{e^{x_k}}{\sum_{i=0}^{9} e^{x_i}}.$$

In practice we will output $\log \circ \operatorname{softmax}$ for numerical stability reasons.

Loss The loss function will be the negative log-likelihood at a given label

$$\begin{aligned} \ell \colon \mathbb{R}^{10} \times \{0, 1, \dots, 9\} &\to \mathbb{R} \\ (x, k) &\mapsto -\log(\operatorname{softmax}(x)_k) \end{aligned}$$

Hyperparameters We perform stochastic gradient descent with $\eta = 10^{-3}$ and a minibatch size of 128. We will iterate over the whole dataset 80 times. The hidden size of the two layer network will be 256.

Data processing The dataset is normalised to have mean 0 and standard deviation of 1. This is a common preprocessing step. The data will also be first flattened by the model, that is, it will be converted from data in $\mathbb{R}^{28\times 28}$ into data in $\mathbb{R}^{784}$.[11]

The procedure to fit this model to the dataset can be written in PyTorch 1.6 as shown in Listing 2.1.

Let us explain what is happening in this example.

Data processing In lines 11 to 18, we download and normalise the dataset. In particular, we download it automatically, and we use the PyTorch convenience functions to apply a transform pipe to the points of this dataset. In this case, we are transforming every image so that the whole dataset has zero mean and unitary standard deviation. The values 0.1307 and 0.3081 are the empirical mean and variance of the dataset and were computed a priori.

[11] By applying this transformation, we are effectively *forgetting* the 2D structure of the data. This is far from being an optimal way to process an image, but it is the simplest.

```
1  import torch
2  import torch.nn as nn
3  import torch.nn.functional as F
4  from torchvision import datasets, transforms
5
6  # Hyperparameters
7  epochs = 80
8  batch_size = 128
9  learning_rate = 0.001
10
11 # Data processing
12 # Centre the MNIST dataset, which has mean 0.1307 and standard deviation 0.3081
13 transform = transforms.Compose([
14     transforms.ToTensor(),
15     transforms.Normalize((0.1307,), (0.3081,))
16     ])
17 dataset = datasets.MNIST('./data', train=True, download=True, transform=transform)
18 loader = torch.utils.data.DataLoader(dataset, batch_size=batch_size, shuffle=True)
19
20 # Run the model on GPU if possible
21 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
22
23 # Instantiate model and optimiser (Stochastic Gradient Descent)
24 model = nn.Sequential(
25             nn.Flatten(),
26             nn.Linear(28*28, 256),
27             nn.ReLU(),
28             nn.Linear(256, 10),
29             nn.LogSoftmax(dim=1)
30         ).to(device)
31 optim = torch.optim.SGD(model.parameters(), lr=learning_rate)
32
33 # Training loop
34 for epoch in range(epochs):
35     for idx, (batch_x, batch_y) in enumerate(loader):
36         batch_x, batch_y = batch_x.to(device), batch_y.to(device)
37
38         out = model(batch_x)
39         loss = F.nll_loss(out, batch_y)
40
41         optim.zero_grad()
42         loss.backward()
43         optim.step()
44         print("Train Epoch: {} [{}/{}]\tLoss: {:.6f}"
45                 .format(epoch, idx*batch_size, len(dataset), loss))
```

Listing 2.1: PyTorch code implementing a two layer feedforward network to classify MNIST

In this part we also wrap the dataset into a loader. The loader is an abstraction that samples batches of size `batch_size` for us from the dataset. We can iterate over it like a list, as we do in Line 35. We also set the `DataLoader` to shuffle the data every time we start iterating over it.

In Line 21, we check whether the computer has a GPU available. If so, we will be computing every operation in a GPU. If there is no GPU available, we fall back to the CPU.

Model specification In Line 24, we define a `Sequential` model. This model is one that splits exactly as a sequential composition of functions, as defined in Definition 2.3.1. We start by flattening the input, that is, applying the linear isomorphism $\mathbb{R}^{28\times 28} \cong \mathbb{R}^{784}$. After that, we apply a linear transformation (*cf.*, Definition 2.3.3) together with the $\texttt{relu}(x) = \max(x, 0)$ non-linearity, where the max is taken coordinate-wise. Finally, in Line 30 we move the model to GPU if there was one available. If not, this line does not do anything. Being able to switch between CPU and GPU with a single line of code is one of the strengths of these modern machine learning libraries.

In Line 31, we define the optimiser to be SGD with a learning rate of $\eta = 10^{-3}$.

Forward pass The gist of the code starts in the training loop. The training loop iterates over the whole dataset 80 times—also called **epochs**—in batches of size 128. As we had mentioned, we can iterate over the `loader`, which returns in every step a batch of size 128 of images and labels. In particular, we have that `batch_x` has dimensions $(128, 28, 28)$ while `batch_y` is a tensor of one dimension of size 128 of integers. As we did with the model, if there was an available GPU, we move the data from CPU to GPU.

In Line 38, we compute the image under our model of every point of the dataset. Using our previous notation, if we have a batch $\{(X_i, Y_i)\}$, this is equivalent to computing $\{f_\theta(X_i)\}$. As such, the output `out` is of size $(128, 10)$, where every vector `out[i]` is the component-wise logarithm of a point on the 10-dimensional simplex.

In Line 39, we compute the average negative log-likelihood loss over the minibatch using the logarithmic probabilities and the true labels. As such, the variable `loss` is just a floating-point number encapsulated in a PyTorch class, that is, a tensor of dimension 1 holding the value of the empirical risk of the minibatch $\widehat{R}_B(\theta)$ as defined in (2.7).

Lines 38 and 39 do much more heavy work than it looks at first sight. When these lines are called, each operation in model, that is, every matrix multiplication, every addition of a bias vector, every coordinate-wise non-linearity, *etc.*, is recorded into a graph called the **computation graph** as it is being executed. This graph contains the dependencies between all the variables in the program as a dependency tree, where the parameters of the model are in the leaves and the inner nodes of this tree are operations putting them together. The computation of $f_\theta(X_i)$ and the creation of the computation graph is what is often called the **forward pass**.

Backward pass The computation of the gradients and the update of the parameters happens in the last three lines of the loop. As we have seen, in Line 31 we have passed the parameters of our model to the optimiser. Line 41 initialises the optimiser, zeroing-out the gradients of all the parameters that it controls, this is just a technical point. Line 42 computes the gradients traversing the computation graph backwards using the chain rule—in this context often called the **adjoint method**. Finally, Line 43 updates the variables with the update step from the optimisation algorithm (*cf.*, Equation (2.8)).

Chapter 3

Differential and Riemannian Geometry

This section is intended as an introduction to some results and computations in differential geometry and Riemannian geometry that will be useful later on. We will skip the proofs of some constructions at the beginning for the sake of the exposition, as they are quite technical, but we will give references to what, in our opinion, are some particularly clean proofs of them. We will always assume all the objects to be in the smooth category, unless stated otherwise.

Outline of the chapter. The problem motivating this whole chapter is that of computing geodesics on a large class of manifolds—naturally reductive homogeneous spaces—as this will be of interest in the rest of the thesis. The chapter is organised going from more general objects to more particular ones, stopping at each layer looking into what it adds to the global picture. We finish the chapter in Section 3.5 showing how to instantiate the constructions in the chapter in a number of manifolds commonly used in optimisation.

Even though all the ideas in this chapter are well-known to geometers, we are not aware of any source that contains all of them. It is for this reason that we include at the beginning of each section a short comment on both historical references and standard references. We also include the proofs necessary to derive the equations for the geodesics in the studied spaces.

3.1 Fibred Manifolds

In this section we provide an introduction to the theory of submersions and Riemannian submersions. This can be seen as a first approach to the theory of Ehresmann connections on fibre bundles. The theory of connections on a fibred manifold as distributions on the total space was first developed in (Ehresmann 1952), while most of the formulae relating the geometry of the base space to the geometry of the total space on a Riemannian submersion is due to O'Neill (O'Neill 1966). A more detailed review of this subject can be found in (Besse 2008, Chapter 9).

We start by recalling the definition of a submersion.

Definition 3.1.1. Let $\pi\colon \overline{M} \to M$ be a map between manifolds. We say that π is a **submersion** if $(\mathrm{d}\pi)_p\colon T_p\overline{M} \to T_{\pi(p)}M$ is surjective for every $p \in \overline{M}$. In particular, if π is surjective, this is equivalent

to saying that $\mathrm{d}\pi\colon T\overline{M} \to TM$ is surjective. We say that $\overline{M}$ is the **total manifold**, and M is the **base manifold**. We will assume submersions to be surjective unless stated otherwise.

The **fibre over** $x \in M$ is defined as $\overline{M}_x \coloneqq \pi^{-1}(x)$. By the implicit function theorem, the fibres of a submersion are manifolds of dimension $\dim(\overline{M}) - \dim(M)$. A pair of manifolds together with a submersion between them is called a **fibred manifold**.

Notation. Even though the notation may suggest otherwise, the manifold $\overline{M}$ is completely unrelated to the manifold M. We choose this notation for fibred manifold as it is possible to identify vector fields X on M with certain class of vector fields $\overline{X}$ on $\overline{M}$. This notation makes this identification easier to follow.

If all the fibres are diffeomorphic, then we get the concept of a space that locally looks like a product of the base space and its fibre. This idea is formalised through fibre bundles.

Definition 3.1.2. Let $\pi\colon E \to M$ be a fibred manifold. Then $\pi\colon E \to M$ is a **fibre bundle over** M **with fibre** F if for every $x \in M$ there is an open neighbourhood U and a map $\alpha\colon \pi^{-1}(U) \to F$ such that

$$(\pi, \alpha)\colon \pi^{-1}(U) \to U \times F$$

is a diffeomorphism. We say that (π, α) is a **bundle chart**. For every $x \in M$ we have that $\alpha\,|_x\colon E_x \to F$ is a diffeomorphism.

All the fibred manifolds we will work with will be fibre bundles.

A key property of fibred manifolds is that they admit local smooth sections. These are mainly used to show that objects defined on fibre bundles are smooth.

Proposition 3.1.3 (Existence of local sections). *Let $\pi\colon \overline{M} \to M$ a fibred manifold. Then, for every $p \in \overline{M}$, there exists a neighbourhood $U \subseteq \overline{M}$, and a smooth function $\sigma\colon \pi(U) \to U$ such that $\pi \circ \sigma = \mathrm{Id}_{\pi(U)}$. This is called* ***a local section through*** *p.*

Proof. See for example (Lang 1999, Ch. II Prop. 2.2) □

Remark 3.1.4 (Fibred spaces). It is in fact possible to show via a simple application of the inverse function theorem that submersions are characterised by the property of having smooth sections. This allows us to generalise the concept of fibred manifolds to the category of topological manifolds simply by swapping smooth sections with continuous sections. This is of particular interest when studying stratified spaces, such as low-rank matrices

$$\mathbb{R}^{n\times k}_{\leq r} = \{X \in \mathbb{R}^{n\times k} \mid \operatorname{rank}(X) \leq r\}.$$

In this case, we have that this is not a manifold, but a collection of manifolds—the collection of the manifolds of fixed-rank matrices (Section 3.5.4)—called strata, with some homogeneity properties. These spaces appear naturally when considering decompositions such as the QR decomposition, which is smooth on each of the strata of this topological manifold. Sadly, we will not have time to explore this connection between topology, numerical analysis, and optimisation on manifolds in this thesis.

Remark 3.1.5 (Global sections). Global sections need not exist in general. In particular, we may not find in general an immersed copy of the base space inside the total space. To see this, consider the following circle bundle over the round sphere:

$$U\mathbb{S}^2 \coloneqq \{(x, v) \in T\mathbb{S}^2 \mid \|v\| = 1\}.$$

It is not difficult to show that this is a fibre bundle over the sphere with fibre $\mathbb{S}^1$. On the other hand, by the hairy ball theorem, it has no global sections, as a global section would give an everywhere non-zero vector field over the 2-sphere. This is a particular example of a general behaviour of non-trivial principal bundles. Note that this is an example of a bundle with no global section, but we can still find an immersed $\mathbb{S}^2$ inside its total space.

The study of the existence or lack of existence of global sections in fibred spaces and fibre bundles is of paramount importance in the field of algebraic topology. An introductory exposition to fibre bundles in this context can be found in (Steenrod 1951).

For a submersion, we define the **vertical bundle** as $V\overline{M} \coloneqq \ker \mathrm{d}\pi \subseteq T\overline{M}$. This is a vector subbundle of $T\overline{M}$.

If we also have a metric $\overline{g}$ on $\overline{M}$, we can define the **horizontal bundle** $\mathcal{H}$ as the fibre-wise orthogonal complement with respect to $\overline{g}$ of $V\overline{M}$, that is,

$$\mathcal{H}_p \coloneqq (V_p\overline{M})^\perp = (\ker(\mathrm{d}\pi)_p)^\perp = \{X \in T_p\overline{M} \mid \overline{g}(X, Y) = 0,\ \forall Y \in V_p\overline{M}\}.$$

These two vector bundles form a split $T\overline{M} = V\overline{M} \oplus \mathcal{H}$. For an element $\zeta \in T\overline{M}$ we denote its split into its horizontal and vertical components by $\zeta = \zeta_V + \zeta_{\mathcal{H}}$.

Note that since $(\mathrm{d}\pi)_p$ is a surjective linear map, $(\mathrm{d}\pi)_p \colon \mathcal{H}_p \to T_{\pi(p)}M$ is a linear isomorphism of vector spaces.

In particular, $\mathcal{H}$ is a smooth choice of a complement of $V\overline{M}$. Note that the only property that we have used of the metric $\overline{g}$ is that it is non-degenerate, so that the complement $(V\overline{M})^\perp$ together with $V\overline{M}$ span all of $T_p\overline{M}$. We can abstract these properties to define a general connection on a fibred manifold.

Definition 3.1.6. We say that a vector subbundle $\mathcal{H}$ of $T\overline{M}$ is an **Ehresmann connection** on a fibred manifold $\pi \colon \overline{M} \to M$ if it is a complement of the vertical bundle, that is, $T\overline{M} = V\overline{M} \oplus \mathcal{H}$.

The split given by an Ehresmann connection makes $\mathrm{d}\pi|_{\mathcal{H}}$ into a vector bundle morphism along π, in the sense that the following diagram commutes

$$\begin{array}{ccc} \mathcal{H} & \xrightarrow{\mathrm{d}\pi|_{\mathcal{H}}} & TM \\ {\scriptstyle \pi_{T\overline{M}}|_{\mathcal{H}}}\downarrow & & \downarrow{\scriptstyle \pi_{TM}} \\ \overline{M} & \xrightarrow{\pi} & M \end{array} \tag{3.1}$$

and $\mathrm{d}\pi|_{\mathcal{H}}$ is a fibre-wise linear isomorphism.

Definition 3.1.7. Let $\pi \colon \overline{M} \to M$ be a fibred manifold with an Ehresmann connection $\mathcal{H}$. A vector field X on M can be uniquely identified with a vector field $\overline{X}$ on $\overline{M}$ with values on $\mathcal{H}$. We call this vector field the **horizontal lift** of X. In symbols,

$$\overline{X}_p \coloneqq (\mathrm{d}\pi|_{\mathcal{H}_p})^{-1}(X_{\pi(p)}) \qquad \forall p \in \overline{M}.$$

This lift gives a **projectable** vector field, meaning that $\mathrm{d}\pi(\overline{X}) = X$ is well-defined. This construction can be readily generalised to any tensor bundle of TM, being able to lift not only vector fields, but also any kind of tensor. In technical words, a connection induces connections on all the associated bundles.

Definition 3.1.8. Let $(\overline{M}, \overline{g}), (M, g)$ be two Riemannian manifolds and let $\pi\colon \overline{M} \to M$ be a submersion between them with induced Ehresmann connection $\mathcal{H}$. We say that π is a **Riemannian submersion** if for every $p \in \overline{M}$

$$\mathrm{d}\pi|_{\mathcal{H}_p}\colon \mathcal{H}_p \to T_{\pi(p)}M$$

is a linear isometry. In other words,

$$\overline{g}_p(\zeta_1, \zeta_2) = g_{\pi(p)}((\mathrm{d}\pi)_p(\zeta_1), (\mathrm{d}\pi)_p(\zeta_2)) \qquad \forall \zeta_1, \zeta_2 \in \mathcal{H}_p.$$

We may equivalently write this condition in terms of horizontal vectors as

$$\overline{g}_p(\overline{u}, \overline{v}) = g_{\pi(p)}(u, v) \qquad \forall u, v \in T_{\pi(p)}M.$$

Remark 3.1.9 (A Riemannian submersion does not restrict to a local isometry). By the definition above, together with the existence of local sections of π, one might be tempted to think that for a point $x \in M$ there exists a section $s\colon \Sigma \to s(\Sigma)$ through $p \in \pi^{-1}(x)$ such that $\pi|_{s(\Sigma)}$ is an isometry. This is not true as, by the local version of Frobenius theorem (Lang 1999, Ch. VI Thm. 1.1), $\mathcal{H}$ is only integrable—there exists a submanifold $N \subseteq \overline{M}$ with $T_pN = \mathcal{H}_p$ for $p \in N$—whenever it is involutive around p, that is, it is closed under the Lie bracket. This means that we may only find a section giving an immersed manifold $s(\Sigma)$ with tangent bundle $\mathrm{d}s(\Sigma) \subseteq \mathcal{H}$ whenever $\mathcal{H}$ is involutive. But $\mathcal{H}$ is often not involutive. In fact, the curvature of $\mathcal{H}$ is defined as how much $\mathcal{H}$ deviates from being involutive. Hence, we may only locally integrate $\mathcal{H}$ when the connection is locally flat. In this case, $\overline{M}$ is locally a Riemannian product of manifolds.

If we have a fibred manifold $\pi\colon \overline{M} \to M$ with an Ehresmann connection $\mathcal{H}$ and a metric g on M, we may lift g to a metric tensor on $\mathcal{H}$ by setting $\overline{g}_{\mathcal{H}} \coloneqq (\mathrm{d}\pi|_{\mathcal{H}})^*(g)$ where $(-)^*$ denotes the pullback

$$\overline{g}_{\mathcal{H}_p}(u, v) \coloneqq g_{\pi(p)}(\mathrm{d}\pi(u), \mathrm{d}\pi(v)) \qquad \forall u, v \in \mathcal{H}_p.$$

Using the decomposition $T\overline{M} = V\overline{M} \oplus \mathcal{H}$, we can choose a metric tensor $\overline{g}_V$ on $V\overline{M}$ and define a metric on $\overline{M}$ declaring the horizontal and vertical bundles to be orthogonal, that is, $\overline{g} \coloneqq \overline{g}_V \oplus \overline{g}_{\mathcal{H}}$. This metric makes $\pi\colon \overline{M} \to M$ into a Riemannian submersion. We can summarise this discussion as follows:

Proposition 3.1.10 (Lifting a metric on a fibred manifold). *Let $\pi\colon \overline{M} \to M$ be a fibred manifold. Given a metric g on M, an Ehresmann connection $\mathcal{H}$ and a metric tensor $\overline{g}_V$ on $V\overline{M}$ we can form a metric that turns π into a Riemannian submersion by setting $\overline{g} = \overline{g}_V \oplus (\mathrm{d}\pi|_{\mathcal{H}})^*(g)$.*

Conversely, if we have a metric $\overline{g}$ on $\overline{M}$ that is projectable—$\overline{g}_{p_1}(\overline{u}, \overline{v}) = \overline{g}_{p_2}(\overline{u}, \overline{v})$ for every $p_1, p_2 \in \overline{M}_x$, $u, v \in T_xM$—we may push it forward to form a metric on M such that π is a Riemannian submersion.

We will show in the next sections how to construct metrics on $V\overline{M}$ in the particular case of principal bundles and homogeneous spaces, where the fibre has the structure of a Lie group.

Riemannian submersions generalise the idea of an isometry between spaces of the same dimension. In particular, they provide a strong link between the geometry of $\overline{M}$ and M.

Proposition 3.1.11. *Let $\pi\colon \overline{M} \to M$ be a Riemannian submersion. We have that:*

1. *A geodesic in $\overline{M}$ with horizontal initial conditions is horizontal everywhere. π takes horizontal geodesics on $\overline{M}$ to geodesics on M. In particular, if $(\overline{M}, \overline{g})$ is complete, so is (M, g).*
2. *Any geodesic on M may be locally lifted to a geodesic on $\overline{M}$ with horizontal initial conditions.*
3. *π is distance decreasing: $\overline{d}(p, q) \geq d(\pi(p), \pi(q))$, $\forall p, q \in \overline{M}$.*
4. *(Hermann 1960) If $(\overline{M}, \overline{g})$ is complete, any curve in M may be lifted globally to a curve in M. In particular, any geodesic in (M, g) may be lifted globally to a horizontal geodesic of $(\overline{M}, \overline{g})$.*
5. *(O'Neill 1966) For X, Y vector fields on M we have*

$$\nabla_{\overline{X}}\overline{Y} = \overline{\nabla_X Y} + \frac{1}{2}[\overline{X}, \overline{Y}]_V$$
$$[\overline{X}, \overline{Y}]_{\mathcal{H}} = \overline{[X, Y]}.$$

Proof. Item 1 can be found in (O'Neill 1983, Ch. 7 Corol. 45). Item 2 is a corollary of the formula for the connection in Item 5. Item 3 is a direct corollary of Item 1. Item 4 is proved in (Hermann 1960, Prop. 3.2). Item 5 is proved in (O'Neill 1966, Lemma 3 and Lemma 1.2) □

According to this proposition, if we have a Riemannian submersion, and we know how to compute geodesics with horizontal initial conditions on the total space, we can compute geodesic on the base space. Furthermore, since $\mathrm{d}\pi$ is surjective, any geodesic on the base space is of this form.

Before finishing this section, we look at Riemannian submersions with totally geodesic fibres, as these will be rather important in the sequel. We start by defining a totally geodesic submanifold.

Definition 3.1.12. An immersed submanifold (F, g_F) of a Riemannian manifold (M, g_M) is **totally geodesic** if geodesics on (F, g_F) are also geodesics on (M, g_M).

Example 3.1.13. On a round sphere maximum circles are geodesic submanifolds but any other circle is not.

The deviation of an immersed submanifold from being a geodesic submanifold is given by the **shape tensor** sometimes also called the **second fundamental form**.

Definition 3.1.14. Given an immersed submanifold (F, g_F) of a Riemannian manifold (M, g_M) with Levi-Civita connections ∇^F, ∇^M respectively, we define the **shape tensor** of F for two vector fields X, Y on F as the $(2, 1)$ tensor

$$II(X, Y) = \nabla^F_X Y - \nabla^M_X Y.$$

A simple computation involving Gauss' equation gives the following characterisation of totally geodesic submanifolds.

Proposition 3.1.15. *A Riemannian submanifold (F, g_F) of (M, g) is totally geodesic if and only if $II \equiv 0$.*

Proof. See (O'Neill 1983, Ch. 4 Prop. 13). □

Totally geodesic submanifolds will be important when we talk about Riemannian submersions with totally geodesic fibres. Most of the Riemannian submersions that we will encounter in the optimisation context will have this property.

In the next sections we will look at two problems: How to construct Riemannian submersions and how to compute geodesics on some total spaces of these submersions. Once we know how to do this, we may repeat this process forming a chain of submersions

$$M_1 \xrightarrow{\pi_1} M_2 \xrightarrow{\pi_2} M_3 \xrightarrow{\pi_3} \dots,$$

where, if we know how to compute geodesics on M_1, and every map is a Riemannian submersion, we know how to compute geodesics on all the other manifolds in the chain. This will be particularly useful for optimisation with orthogonal constraints as shown in Section 3.5, where we will see that the special orthogonal group, the Stiefel manifold, and the Grassmannian form such a tower. We will also leverage this in Chapter 6 to implement optimisation on different manifolds by implementing it on one manifold, and then pushing the geodesics forward along Riemannian submersions.

3.2 Invariant Metrics and Principal Bundles

In this section we will study manifolds together with a group action. In particular, we will be interested in studying conditions for the existence of an invariant metric under this action. This will take us naturally to the definition of principal bundles which, in turn, will provide us with a way to construct Riemannian submersions. Most of the theory presented in this section and the next one can be generalised to study connections on principal and associated bundles. The presentation in this section differs from the classical one in that we emphasise the Riemannian side of principal bundles as spaces on which we can find a metric invariant under a Lie group action that descends into the quotient of the manifold by the action.

Principal bundles were first proposed in full generality by Palais, who introduced the study of proper actions on manifolds, as a generalisation of actions by compact groups (Palais 1961). The results of this section can mostly be found across the books (Kobayashi and Nomizu 1963; Kobayashi and Nomizu 1969), written by two of the fathers of the subject.

We start by defining some notation and recalling some rudiments of Lie groups.

Notation. Let $\mu\colon G \times M \to M$ be an action of a Lie group G on a manifold M. We will write $\mu_g(x) = \mu^x(g) = \mu(g, x)$ for the different partial maps defined by it. In particular, μ_g is a diffeomorphism with inverse $\mu_{g^{-1}}$.

We will denote **left and right actions** of a Lie group onto itself by L_g and R_g, and the **conjugacy action** of a Lie group by $c_g \coloneqq L_g \circ R_{g^{-1}}$.

For a Lie group G, H, we will denote their Lie algebras as $\mathfrak{g}, \mathfrak{h}$. Given a vector $X \in \mathfrak{g}$, we denote the **left-invariant vector field** associated to X by

$$X^{\mathtt{L}}_g \coloneqq (\mathrm{d}L_g)_e(X).$$

Left-invariant vector fields provide a linear isomorphism between T_gG and $\mathfrak{g}$ and they allow us to define left-invariant tensors on the whole manifold by defining them just on the Lie algebra. For example, if we have an inner product g_e on the Lie algebra, we can define a left-invariant metric as

$$g^{\mathtt{L}}_g(X^{\mathtt{L}}_g, Y^{\mathtt{L}}_g) \coloneqq g_e(X, Y) \qquad X, Y \in \mathfrak{g}.$$

We recall that the **Lie exponential** is defined as the integral of left-invariant vector fields, that is, if $v \in \mathfrak{g}$, $\operatorname{expm}(tX) = \gamma(t)$ we have that

$$\dot{\gamma}(t) \coloneqq X^{\mathtt{L}}_{\gamma(t)}.$$

We will denote it by expm, to differentiate it from the Riemannian exponential map.

We denote the **orbit of a point** $x \in M$ as

$$G(x) \coloneqq \{\mu_g(x) \mid g \in G\},$$

and the **stabiliser of** x, also referred to as the **isotropy group at** x, as

$$G_x \coloneqq \{g \in G \mid \mu_g(x) = x\}.$$

Note that for continuous actions $G_x = (\mu^x)^{-1}(\{x\})$ is closed.

Remark 3.2.1 (Left and right actions). In the sequel we will be using left and right actions. The use of one or the other is merely due to convenience, since we may turn a left action μ_g into a right action by considering the map $g \mapsto \mu_{g^{-1}}$. We will reserve right actions for actions we use to quotient by, and left actions to talk about actions that act on this quotient.

We are interested in studying metrics that are invariant under the action of a group, as these metrics will descend into the quotient of the manifold by the group action.

Definition 3.2.2. Given a manifold M together with a right action μ by a Lie group G, we say that a metric g on M is G**-invariant** if

$$g_{\mu_g(p)}(\mathrm{d}\mu_g(u), \mathrm{d}\mu_g(v)) = g_p(u, v) \qquad \forall g \in G, p \in M, u, v \in T_pM,$$

in other words, the action μ_g is an isometry for every $g \in G$. Furthermore, if the group of isometries $\mu_G = \{\mu_g \mid g \in G\}$ is a closed subgroup of $\operatorname{Iso}(M, g)$, we say that G **acts by isometries**.

Remark 3.2.3 (Effective actions). Note that μ_G is isomorphic to G if $\mu_g = \mathrm{Id}$ implies that $g = e$. We call these **effective actions**. If an action is not effective, we can define the **ineffective kernel** of the action $N = \bigcap_{x \in M} G_x$. This is a normal subgroup of G, and in this setting μ_G is isomorphic to G/N as Lie groups. In particular, the action of G/N on the manifold M is equivalent to that of G, so in most situations we can safely assume that the action is effective, although at times it is convenient to work with non-effective actions, when doing explicit computations in concrete manifolds.

Remark 3.2.4 (Closed subgroups of isometries). By a theorem of Myers and Steenrod 1939, the isometry group $\operatorname{Iso}(M, g)$ is a Lie group. Hence, the requirement that μ_G is a closed subgroup of $\operatorname{Iso}(M, g)$ implies that μ_G is itself a Lie group. We will see in shortly that having a closed subgroup of isometries will allow us to *quotient* by it—*i.e.*, putting a unique smooth structure on the space of orbits such that it makes the projection a submersion—which may not be possible when the group is not closed.

The first kind of actions that we will consider are proper actions. An action is said to be **proper** if the map from $G \times M$ to $M \times M$, $(g, x) \mapsto (\mu_g(x), x)$ is proper, in the sense that the preimage of a compact set is compact. Proper actions give us a way to construct G-invariant metrics. Even though this result goes back to Palais, it was recently shown that this metric can be chosen to be complete.

Theorem 3.2.5 (Kankaanrinta 2005). *Let M be a manifold with a proper G-action μ. Then, there exists a complete metric g which makes μ_G into a closed subgroup of* $\mathrm{Iso}(M, g)$.

The second kind of actions that we will look at are free actions. A **free action** is one that has no non-trivial fixed points. In other words, if $\mu_g(x) = x$ for an $x \in M$, then $g = e$. Another way of looking at this is that every isotropy group is trivial. Free isometric actions are actions by which we can quotient and obtain a smooth manifold.

Theorem 3.2.6 (Free slice theorem). *Let G be a closed subgroup of* $\mathrm{Iso}(M, g)$ *that acts freely. Then, the set of orbits $M/G = \{G(x) \mid x \in M\}$ can be given a unique manifold structure and a Riemannian metric such that $\pi \colon M \to M/G$ is a Riemannian submersion.*

Proof. See (Petersen 2016, Theorem 5.6.21). □

Combining these two theorems we have the following corollary.

Corollary 3.2.7. *Let M be a manifold together with a free proper action. There exists a G-invariant metric on M that descends into a metric on M/G which makes $\pi \colon M \to M/G$ into a Riemannian submersion.*

Proper actions are a reasonably general setting on which to study invariant metrics.

Theorem 3.2.8 (Isometric actions are proper). *Let (M, g) be a Riemannian manifold and G be a closed subgroup of* $\mathrm{Iso}(M, g)$. *Then, its action on M is proper.*

Proof. See (Alexandrino and Bettiol 2015, Theorem 3.62). □

Remark 3.2.9 (Isometric actions vs. effective and proper). Theorem 3.2.8 gives a converse to Theorem 3.2.5. Between these two theorems, we have an equivalence between effective and proper G-actions on a manifold M and closed subgroups G of $\mathrm{Iso}(M, g)$ where g is a complete metric.

Remark 3.2.10 (Non-free isometric actions). In some situations, it is not strictly necessary for the action to be free but it certainly makes things easier. If the action is not free, one still has an open and dense subset of the manifold whose quotient by the action is a manifold. This is part of the content of the **principal orbit theorem**, and the start of the theory of Riemannian foliations (Alexandrino and Bettiol 2015, Thm. 3.82).

Let us now define principal bundles, which are fibre bundles whose fibre is a Lie group, together with a compatible fibre-preserving action.

Definition 3.2.11. Let $\pi \colon P \to M$ be a fibre bundle with fibre a Lie group G. We say that P is a **principal bundle** if we have a free and transitive (on the fibres) fibre-preserving right action $\mu \colon P \times G \to P$, and an atlas such that the bundle charts $\alpha \colon \pi^{-1}(U) \to G$ are equivariant with respect to the action,

$$\alpha(\mu(p, g)) = \alpha(p)g \qquad \forall p \in \pi^{-1}(U), g \in G.$$

This makes the charts compatible with the action μ on P and the action of multiplication on the right on G.

We now show that principal bundles are the right abstraction to look at metrics that are G-invariant by which we can quotient.

Theorem 3.2.12 (Construction of principal bundles). *Let M be a manifold with a free and proper G-action. Then, $\pi\colon M \to M/G$ has the structure of a principal bundle.*

Proof. See (Alexandrino and Bettiol 2015, Theorem 3.34). A sketch of the proof goes as follows:

Since the action is free and proper, by Theorem 3.2.6 the map $\pi\colon M \to M/G$ is a submersion, so every fibre $M_x \coloneqq \pi^{-1}(\{x\})$ is a manifold. These fibres are, by definition, the orbits of the action and, since the action is free, they are all isomorphic to G, so a principal bundle is a fibre bundle with fibre G and a fibre-preserving free and proper action. Adapting the proof of Theorem 3.2.6 one proves that the bundle charts are equivariant. □

And we have the following converse.

Proposition 3.2.13 (Characterisation of principal bundles). *The action μ of a principal bundle is free and proper.*

Proof. See (Alexandrino and Bettiol 2015, Proposition 3.33). These properties follow from the fact that an action of a Lie group on itself is free and proper, and using the equivariance of the bundle charts. □

In particular, every principal bundle admits an invariant metric that descends into the base space and makes π into a Riemannian submersion.

Until now, we have argued that principal bundles are spaces that accept G-invariant metrics, and that if we have a G-invariant metric on the total space, it descends into a metric on the base space. We will now see some sort of converse to this result. In particular, we will see how to lift a metric from the base space into the total space such that the metric on P is G-invariant. In general, we described in Section 3.1 how, given a metric on a base space and a connection on the total space, we may lift it to a metric on the total space of a fibred manifold by declaring $\mathrm{d}\pi$ to be an isometry and choosing a metric tensor on the vertical bundle. In the case of principal bundles, the fact that the fibres look like Lie groups allows us to simplify the process of choosing a metric tensor on VP to choosing one on the Lie algebra.

We start by defining a connection on a principal bundle.

Definition 3.2.14. A **connection on a principal G-bundle** $\pi\colon P \to M$ (or simply a **principal connection**) $\mathcal{H} \subseteq TP$ is a G-equivariant Ehresmann connection, that is, a G-equivariant sub-bundle complementary to VP. In symbols,

$$(\mathrm{d}\mu_g)_p(\mathcal{H}_p) = \mathcal{H}_{\mu(p,g)} \qquad \forall p \in P, g \in G.$$

It is direct to see that a G-invariant metric on P induces a connection on P defining $\mathcal{H}_p \coloneqq (V_pP)^\perp$, and it is exactly the G-invariance of the metric what gives the equivariance. In particular, since principal bundles accept G-invariant metrics, every principal bundle admits a principal connection. We may use these to lift a metric from M to P making $\pi\colon P \to M$ into a Riemannian submersion.

Theorem 3.2.15 (Lifting a metric on a principal bundle (Vilms 1970)). *Let $\pi\colon P \to M$ be a principal G-bundle. Given a metric g on M, an inner product g_e on $\mathfrak{g}$ and a principal connection $\mathcal{H}$, there exists a unique metric $\overline{g}$ on P such that π is a Riemannian submersion with totally geodesic fibres isometric to $(G, g_e{}^{\mathrm{L}})$—the left-invariant metric on G associated to g_e—and horizontal distribution $\mathcal{H}$.*

Proof. Given the principal connection $\mathcal{H}$, we can put a metric on the horizontal space by pulling back the metric on M as $g_{\mathcal{H}} = (\mathrm{d}\pi|_{\mathcal{H}})^*(g)$. We can then declare $\mathcal{H}$ and VP to be orthogonal, and put a metric on VP by transferring the right-invariant metric on G to P via the bundle charts. If $\alpha\colon \pi^{-1}(U) \to G$ is a bundle chart, the metric is defined locally on $\pi^{-1}(U)$ as

$$\overline{g}|_{\pi^{-1}(U)} = \alpha^*(g_e{}^{\mathrm{L}}) \oplus (\mathrm{d}\pi|_{\mathcal{H}\cap T\pi^{-1}(U)})^*(g|_U).$$

This is well-defined as, since the charts are right-equivariant, we have that for any two bundle charts α_U, α_V with $U \cap V \neq \emptyset$,

$$\alpha_V \circ \alpha_U^{-1} = L_g$$

for some $g \in G$ (see *e.g.*, Naber 2011, Lemma 4.2.1).

The fact that the fibres in this construction are totally geodesic submanifolds of P is proved in (Vilms 1970, Theorem 3.5).

The uniqueness of this construction comes from the fact that the fibres are totally geodesic, so they have to be orthogonal to the connection subbundle. □

Remark 3.2.16 (Associated bundles). The construction above goes through in the more general setting of associated bundles, as proved in the original paper. On the other hand, the simpler case of principal bundles is enough for what we will need in the sequel.

Let us now define a family of vector fields which will be useful later.

Definition 3.2.17. For a principal G-bundle $\pi\colon P \to M$ with action μ we define the **infinitesimal generators of the action** as

$$\xi\colon P \times \mathfrak{g} \to VP$$
$$(p, v) \mapsto \xi_v(p) \coloneqq (\mathrm{d}\mu^p)_e(v)$$

Example 3.2.18. In the simplest case of the principal bundle $\pi\colon G \to \{e\}$, $\xi_v(g) = (\mathrm{d}L_g)_e(v)$. In other words, ξ_v are just left-invariant vector fields on G, that is, $\xi_v = v^{\mathrm{L}}$.

The infinitesimal generators of the action take values on VP, since for a $p \in P$, $(\mathrm{d}\pi)_p \circ (\mathrm{d}\mu^p)_e = \mathrm{d}(\pi \circ \mu^p)_e = 0$, where we have used that the action μ^p is fibre-preserving, and thus $\pi \circ \mu^p$ is constant. The infinitesimal generators provide with bundle-isomorphism between VP and $P \times \mathfrak{g}$, and hence, they show that the vertical bundle is always trivial.

Proposition 3.2.19. *Let $\pi\colon P \to M$ be a principal G-bundle, then $(\mathrm{d}\mu^p)_e\colon \mathfrak{g} \to V_pP$ is a linear isomorphism, so that the vertical space factorises as $VP \cong P \times \mathfrak{g}$. We also have that the infinitesimal generators induce a Lie algebra homomorphism*

$$\xi_{[u,v]} = [\xi_u, \xi_v] \qquad \forall u, v \in \mathfrak{g}.$$

Proof. For a $p \in P$, V_pP can be identified with the tangent to the orbit of G at p. This orbit is diffeomorphic to G, as the action is free, so V_pP and $\mathfrak{g}$ have the same dimension. If we prove that $(\mathrm{d}\mu^p)_e$ is injective, then it is a linear isomorphism. Let $v \in \mathfrak{g}$ such that $(\mathrm{d}\mu^p)_e(v) = 0$ for $p \neq e$. This is equivalent to saying that

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \mu(p, \exp\mathrm{m}(tv)) = 0.$$

Thus, p is locally a fixed point of the map $\mu^{\exp\mathrm{m}(tv)}$, but since the action is free this only happens when $v = 0$ so $(\mathrm{d}\mu^p)_e$ is injective.

To prove that ξ is an Lie algebra homomorphism, we have by equivariance that

$$\mu_g \circ \mu^p(\exp\mathrm{m}(tv)) = \mu^{\mu_g(p)}(g^{-1}\exp\mathrm{m}(tv)g) \quad \forall v \in \mathfrak{g}, g \in G, p \in P$$

and differentiating at $t = 0$ we get the following infinitesimal equivariance property:

$$(\mathrm{d}\mu_g)_p(\xi_v(p)) = \xi_{\mathrm{Ad}_{g^{-1}}(v)}(\mu_g(p)) \qquad \forall g \in G, p \in P, v \in \mathfrak{g}. \tag{3.2}$$

Finally, letting $g = \exp\mathrm{m}(-tu)$, substituting p with $\mu(\exp\mathrm{m}(tu), p)$ in (3.2), and differentiating

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} (\mu_{\exp\mathrm{m}(-tu)})_*(\xi_v(\mu(\exp\mathrm{m}(tu), p))) = \frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \xi_{\mathrm{Ad}_{\exp\mathrm{m}(tu)}(v)}(p) \qquad \forall p \in P, u, v \in \mathfrak{g}.$$

The left-hand side is the definition of $[\xi_u, \xi_v](p)$, while the right-hand side is equal to $\xi_{[u,v]}(p)$, after using that the derivative of a linear map is the map itself. □

Remark 3.2.20 (Lifting a metric on a principal G-bundle for G compact). In the case when G is compact, the construction of the metric on VP in Theorem 3.2.15 may be greatly simplified by the use of infinitesimal generators. As it was shown in Equation (3.2), $(\mathrm{d}\mu_g)_p$ may be evaluated using infinitesimal generators. In particular, it acts as the adjoint action on $\mathfrak{g}$. If we have an $\mathrm{Ad}(G)$-invariant inner product on $\mathfrak{g}$, we may then pull it back to a G-invariant metric tensor on VP. This metric is much simpler to work with and compute, as it does not make use of the bundle charts.

As we will see in Proposition 3.3.9, on compact Lie groups one may construct $\mathrm{Ad}(G)$-invariant metrics, so we may carry this construction whenever G is compact. Luckily, we will just study the setting of G-compact in the sequel, as all Riemannian homogeneous spaces are of this form.

3.3 Homogeneous Spaces

We now come back to the computation of geodesics on different manifolds. The discussion in the previous section tells us that, by Corollary 3.2.7, if one has a G-invariant metric on a principal bundle, then this metric descends to a metric on the quotient $P/G \cong M$, turning the projection into a Riemannian submersion. Since a Riemannian submersion maps horizontal geodesics into geodesics, if we know how to compute geodesics on P, then we can compute geodesics on M just by projecting the horizontal ones.

In this section, we will address the questions of how to implement these ideas in concrete families of manifolds. In particular, we will look at families of manifolds on which we can compute the geodesics easily, so that we can get the geodesics on the quotient simply by projecting onto it.

The theory of (left-) invariant metrics on a space with a transitive action was pioneered by Nomizu 1954, which was the first to realise that, if one has a transitive action, it can be used to translate geometric problems on a manifold to algebraic problems on a tangent space. An introduction to reductive spaces in terms of representations can be found in (Kobayashi and Nomizu 1969, Ch. X Sec. 1 and 2). An early introduction to the subject of reductive homogeneous spaces avoiding representations and delving deeper into the theory of the canonical connection of a reductive space can be found in (Kowalski 1980, Ch. 1). The theory of reductive structures was first systematically studied by Lichnerowicz in (Lichnerowicz 1958). A concise and self-contained introduction to a number of the ideas in this section and the next one can also be found in the excellent text (Cheeger and Ebin 2008, Chapter 3).

We start by looking at the simplest example. Let H be a closed subgroup of a Lie group G, consider the right action of H on G by right multiplications. It is not difficult to see that this action is proper and free, so we have our first specific family of principal bundles.

Proposition 3.3.1. *Let H be a closed subgroup of a Lie group G acting on G via right multiplications. Then, $\pi\colon G \to G/H$ is a principal H-bundle.*

Manifolds that are a quotient of a Lie group by a closed subgroup are particularly important and are called **homogeneous spaces**.

Definition 3.3.2. A manifold M is a G-**homogeneous space** if it is diffeomorphic to a manifold of the form G/H with H a closed subgroup of a Lie group G.

The name of homogeneous spaces comes from the fact that these are exactly the manifolds that admit transitive actions, as we will show in a second.

Definition 3.3.3. An G-action is **transitive** on M if for every $x, y \in M$, there exists a $g \in G$ such that $\mu_g(x) = y$.

When G is a Lie group acting on a smooth manifold, a transitive action μ_g allows us to move any point to any other point. In particular, since μ_g is a diffeomorphism (its inverse is $\mu_{g^{-1}}$), these manifolds look the same from a smooth point of view at every point.

Proposition 3.3.4 (Construction and characterisation of homogeneous spaces). *Let H be a closed subgroup of G. The G-action by left multiplications induced on the quotient G/H is transitive, so G/H is a G-homogeneous space.*

Conversely, let M be a G-homogeneous space with action μ, and fix a point $x_0 \in M$. Then, M is diffeomorphic to the quotient of G by the stabiliser of x_0, $H \coloneqq G_{x_0}$ by right multiplications, through the diffeomorphism

$$\begin{aligned} \widetilde{\mu}^{x_0}\colon G/H &\to M \\ \pi(g) &\mapsto \mu^{x_0}(g) \end{aligned}.$$

Remark 3.3.5 (Maps that descend to the quotient). A map $\Phi\colon P \to P$ on a principal bundle map that commutes with the action μ_g descends into a map on the quotient $\widetilde{\Phi} \coloneqq \pi \circ \Phi$. We will use this in the setting of the principal bundle of a homogeneous space $\pi\colon G \to G/H$, with R_h for $h \in H$ as the bundle right-action and $\Phi = L_g$. The left-action then descends into a left-action $\widetilde{L}_g$ on G/H since left and right multiplication on a Lie group always commute.

For a homogeneous space M, there might be different groups that act transitively on it. Therefore, its representation as a quotient of Lie groups may not be unique. To see this, we shall first introduce a large family of examples for homogeneous spaces.

Definition 3.3.6. A Riemannian manifold (M, g) is a **Riemannian homogeneous space** if its isometry group $\mathrm{Iso}(M, g)$ acts transitively on M.

Example 3.3.7 (Non-unique quotient representation of a homogeneous space). Let $M = \mathbb{R}^n$ with the canonical metric, the full isometry group is generated by rotations, reflections, and translations, but the subgroup of translations acts transitively on $\mathbb{R}^n$ and is a closed subgroup of $\mathrm{Iso}(\mathbb{R}^n, g_{\mathbb{R}^n})$. In other words, both the full isometry group and the subgroup of translations act transitively on $\mathbb{R}^n$. Hence, if we denote the translations on $\mathbb{R}^n$ by $\mathrm{T}(n)$, we have two representations of $\mathbb{R}^n$ as $\mathbb{R}^n \cong \mathrm{T}(n) \cong \mathrm{Iso}(\mathbb{R}^n, g_{\mathbb{R}^n})/\mathrm{O}(n)$, where we have used that the group of translations has a trivial stabiliser.

Riemannian homogeneous spaces have a particularly simple geometry. In particular, by definition, for any two points there exists an isometry that transforms one into the other. Hence, quantities that are invariant under isometries, such as the curvature tensor or the Levi-Civita connection, may be computed at all points just by computing them at one point and translating them using the isometries of the manifold. In other words, from the perspective of Riemannian geometry, these manifolds look the same at every point, so it is enough to study them at one of their points—or in a neighbourhood of one point—to understand their geometry at any given point.

The stabilisers of Riemannian manifolds are always compact. This means that a Riemannian homogeneous space can always be represented as G/H with H compact.

Proposition 3.3.8 (Necessary condition for Riemannian homogeneous spaces). *Let (M, g) be a Riemannian manifold and let G be a closed subgroup of $\mathrm{Iso}(M, g)$, then its stabiliser H at a point $x_0 \in M$ is compact.*

Proof. By Theorem 3.2.8 an action by isometries is proper, and since $H = (\mu^{x_0})^{-1}(\{x_0\})$, H is compact. □

The converse is also true. In order to prove it, we first need the following important construction on Lie groups called **the averaging trick**.

Proposition 3.3.9 (Bi-invariant metric by a compact subgroup). *Let G be a Lie group and H a compact subgroup. Then G admits a left-G-invariant metric that is also right-H-invariant*

Proof. Let $\omega_{\mathfrak{h}}$ be a non-zero alternating multilinear map on $\mathfrak{h}$ in $k = \dim \mathfrak{h}$ variables (*i.e.*, $\omega_{\mathfrak{h}} \in \bigwedge^k \mathfrak{h}^*$). Since $\dim \bigwedge^k \mathfrak{h}^* = 1$, this form is unique up to a multiplicative constant. We can then extend it to all of H by pushing it forward along R_h, defining $(\omega_H)_h \coloneqq (R_h)_*(\omega_{\mathfrak{h}})$. This makes ω_H into a right-invariant top-dimensional form on H, and its associated measure is called the **right Haar measure on** H.

Let $\langle \cdot, \cdot \rangle$ be any inner product on $\mathfrak{g}$. We can turn this inner product into an $\mathrm{Ad}(H)$-invariant inner product on $\mathfrak{g}$ by averaging it over the elements of the group using the right Haar measure ω_H

$$g_{\mathfrak{g}}(u, v) \coloneqq \int_H \langle \mathrm{Ad}_h(u), \mathrm{Ad}_h(v) \rangle \, \omega_H \qquad u, v \in \mathfrak{g}.$$

Note that this integral is well-defined since H is compact. The $\mathrm{Ad}(H)$-invariance follows from the right-invariance of ω

$$g_{\mathfrak{g}}(\mathrm{Ad}_{h'}(u), \mathrm{Ad}_{h'}(v)) = \int_H \langle \mathrm{Ad}_{hh'}(u), \mathrm{Ad}_{hh'}(v) \rangle \, \omega_H = g_{\mathfrak{g}}(u, v) \qquad \forall u, v \in \mathfrak{g}, \forall h' \in H.$$

Finally, we can extend this inner product to a metric on all of G by pushing it forward along L_g, $g_g \coloneqq (L_g)_*(g_{\mathfrak{g}})$. This makes g into a left-G-invariant metric, but since it is $\mathrm{Ad}(H)$-invariant and it is left-H-invariant, it is right-H-invariant as well. □

We call a metric that is both left-invariant and right-invariant **bi-invariant**. When G is itself compact, we may choose $H = G$ in the theorem above, getting the following result.

Corollary 3.3.10 (Bi-invariant metrics on compact Lie groups). *A compact Lie group G admits a bi-invariant metric.*

Remark 3.3.11 (Correspondence between invariant metrics and $\mathrm{Ad}(H)$-invariant inner products). It is easy to see by going in the opposite direction in the construction of Proposition 3.3.9 that left-G-invariant right-H-invariant metrics on G are in one-to-one correspondence with $\mathrm{Ad}(H)$-invariant inner products on $\mathfrak{g}$.

With this construction in hand, we may prove the converse of Proposition 3.3.8.

Proposition 3.3.12 (Sufficient condition for Riemannian homogeneous spaces). *Let M be a manifold with a transitive action. If M has a compact stabiliser $H = G_{x_0}$ then, there is a metric g that makes (M, g) into a Riemannian homogeneous space.*

Proof. Following the construction in Proposition 3.3.9, we may construct a metric on G such that it is left-G-invariant and right-H-invariant. Since it is right-H-invariant, it descends into a metric g on $M \cong G/H$, and since it is left-G-invariant, the metric on M is G-invariant, that is, (M, g) is a Riemannian homogeneous space. □

Remark 3.3.13 (A homogeneous space may not be Riemannian homogeneous). It is not the case that every homogeneous space admits a metric that makes it into a Riemannian homogeneous space. For example, the family of matrices of a fixed rank never admits a Riemannian homogeneous structure outside the trivial case, as noted in Section 3.5.4.

As we have seen, on a Riemannian homogeneous space M, after fixing a point $x_0 \in M$, its isometry group may be endowed with a metric that is right-H-invariant for $H = G_{x_0}$. In other words, this metric descends into the quotient G/H, making it isometric to M by construction. Furthermore, since this metric can be chosen to be left-G-invariant, it makes the diffeomorphism from Proposition 3.3.4 into an isometry and the map $\pi \colon G \to G/H$ into a Riemannian submersion.

As a Riemannian homogeneous space is a principal bundle with a metric that descends onto the quotient, we know that geodesics on G/H are just the projection of geodesics on G with a given initial horizontal vector. Hence, one piece that we are missing to be able to compute geodesics on Riemannian homogeneous spaces is to figure out what constitutes a horizontal vector for these metrics. In order to be able to do this, we will work with a class of Riemannian homogeneous spaces for which the horizontal

bundle is particularly simple. We have already implicitly worked with these spaces during the proof of Proposition 3.3.9, and now we will give them a name.

We aim to be able to left-translate a tensor that descends from T_eG to $T_{eH}G/H \cong T_{x_0}M$ onto all of M. Since we are taking the quotient by right action of the isotropy group H, any tensor that descends into the quotient should be right-H-invariant. Since we will be looking at tensors that are $\mathrm{d}\widetilde{L}_g$-invariant, we are in particular interested in looking at tensors that are $\mathrm{d}\widetilde{L}_h$-invariant for $h \in H$. These two things together take us into looking at tensors that are $\mathrm{Ad}(H)$-invariant. We can formalise this idea by looking at the quotient isotropy representation $(\mathrm{d}\widetilde{L}_h)_e \coloneqq (\mathrm{d}\pi \circ \mathrm{d}L_h)_e$. To compute it, we look first at the quotient action on the left

$$\widetilde{L}_h(g) \coloneqq \pi(L_h(g)) = hgH = hgh^{-1}H = \pi(c_h(g)) \qquad g \in G, h \in H$$

and differentiating this relation at $g = e$ we get

$$(\mathrm{d}\widetilde{L}_h)_e(v) = (\mathrm{d}\pi)_e(\mathrm{Ad}_h(v)) \qquad h \in H, v \in \mathfrak{g}.$$

In other words, we see that the differential of the diffeomorphisms $\widetilde{L}_h$ can be described via the adjoint representation of H on $\mathfrak{g}$. Since $\mathfrak{h}$ is clearly $\mathrm{Ad}(H)$-invariant, the study of tensors that are $\mathrm{d}\widetilde{L}_h$-invariant heavily simplifies when $\mathfrak{g}$ splits into two complementary vector spaces that are $\mathrm{Ad}(H)$-invariant.

Definition 3.3.14. We say that a homogeneous space G/H is **reductive** if there is a linear complement $\mathfrak{g} = \mathfrak{h} \oplus \mathfrak{m}$ such that

$$\mathrm{Ad}_h(\mathfrak{m}) \subseteq \mathfrak{m} \qquad \forall h \in H.$$

Remark 3.3.15. The subspace $\mathfrak{m}$ does not need to be an ideal, as it may not be closed under the Lie bracket.

We now give a more geometric interpretation of the usefulness of the reductive condition. But first we recall a basic geometric fact about Lie groups.

Remark 3.3.16 (Lie groups have trivial tangent bundle). Looking at the principal bundle $\pi\colon G \to \{e\}$ of G acting on itself via right multiplication, we already saw in Proposition 3.2.19 that the infinitesimal generators of the action—left-invariant vector fields in this case—provide a vector bundle isomorphism $\xi\colon G \times \mathfrak{g} \to VG$. Since in this case $VG = TG$, this proposition translates to the classical fact that left-invariant vector fields give a trivialisation of the tangent bundle of a Lie group, that is, $\xi(v) = v^{\mathrm{L}}$.

Remark 3.3.17 (A distinguished principal connection on a reductive homogeneous space). Since G is a principal H-bundle, a principal connection gives a split $TG = \mathcal{H} \oplus VG$. We have that, through left-invariant vector fields, $VG \cong G \times \mathfrak{h}$. What the reductive condition gives us is a distinguished principal connection $\mathcal{H}_g \coloneqq (\mathrm{d}L_g)_e(\mathfrak{m})$. Since this is a principal connection, it descends into the quotient G/H to give a bundle morphism along π between $\mathcal{H} \cong G \times \mathfrak{m}$ and $T(G/H)$ which is an isomorphism on the fibres, as described in (3.1). Furthermore, the equivariance allows us to identify an element in $T(G/H)$ with an element in $\mathfrak{m}$ modulo a choice of an element in $h \in H$, coming from the fact that $\mathrm{Ad}_h(\mathfrak{m}) = \mathfrak{m}$, but $\mathrm{Ad}\,|_{\mathfrak{m}} \neq \mathrm{Id}_{\mathfrak{m}}$. In practice, when identifying an element $(x, v) \in T(G/H)$ with an element in $\mathfrak{m}$, we will have to choose an element of the fibre $\pi^{-1}(x) \cong H$. This choice will not matter in the definition of the metric tensor on G, as the inner products that we consider on $\mathfrak{m}$ will be $\mathrm{Ad}(H)$-invariant (*cf.* Proposition 3.3.9).

We have access to all these desirable properties in the setting of Riemannian homogeneous spaces, as every Riemannian homogeneous space is reductive.

Proposition 3.3.18. *Every Riemannian homogeneous space is reductive.*

Proof. A Riemannian homogeneous space can be represented as G/H with H a compact subgroup of G (Proposition 3.3.8). Following Proposition 3.3.9, fixing an inner product on $\mathfrak{g}$, we may average it into an $\mathrm{Ad}(H)$-invariant inner product on $\mathfrak{g}$. The choice $\mathfrak{m} \coloneqq \mathfrak{h}^{\perp}$ is a reductive complement of $\mathfrak{h}$. □

The language of reductive homogeneous spaces allows us to give a more concise version of the ideas already presented in Proposition 3.3.9. In particular, it allows us to give a simpler version of the construction in Theorem 3.2.15 in the context of reductive homogeneous spaces.

Theorem 3.3.19 (Lifting a metric on a reductive homogeneous space). *Let (M, g) with $M \cong G/H$ be a reductive homogeneous space and let $g_{\mathfrak{h}}$ be an $\mathrm{Ad}(H)$-invariant inner product on $\mathfrak{h}$ (which exists when H is compact). There exists a unique metric $\overline{g}$ on G that makes $\pi\colon G \to M$ into a Riemannian submersion with totally geodesic fibres isometric to $(H, {g_{\mathfrak{h}}}^{L})$.*

Proof. We may lift the metric g into a left-G-invariant metric on the canonical distribution $\mathcal{H} = \mathfrak{m}^{L}$ as done in Theorem 3.2.15 by defining $g_{\mathcal{H}} = (\mathrm{d}\pi|_{\mathcal{H}})^{*}(g)$. At the same time, the $\mathrm{Ad}(H)$-invariant inner product on $\mathfrak{h}$ may be left-transported as in the proof of Proposition 3.3.9 to a right-G left-H-invariant product on $VG = \mathfrak{h}^{L}$. The metric $\overline{g}$ comes from declaring the horizontal and vertical distributions to be orthogonal, that is, $\overline{g} = {g_{\mathfrak{h}}}^{L} \oplus g_{\mathcal{H}}$. This metric is left-$G$-invariant and right-$H$-invariant by construction and makes π into a Riemannian submersion. The proof that the fibres of this construction are totally geodesic follows from Theorem 3.2.15. □

It is worth noting that a more general version of this theorem holds as first proved in the paper (Bérard-Bergery 1975) (in French). If we have $K \subseteq H$ closed and compact subgroups of a Lie group G, we can still form a Riemannian submersion with totally geodesic fibres of the form

$$\pi\colon G/K \to G/H$$
$$gK \mapsto gH$$

although in this case we end up with a fibre bundle with fibre the homogeneous space H/K, rather than a principal bundle. The details can be found in English in (Besse 2008, Thm. 9.80).

3.4 Naturally Reductive Homogeneous Spaces

In this section, we look at a particularly nice example of Riemannian homogeneous spaces for which the Levi-Civita connection of the metric coincides with the natural torsion-free affine connection on G/H. These are called **naturally reductive homogeneous spaces**, and were introduced in (Nomizu 1954). An introductory treatment with a more modern notation can be found in (Kobayashi and Nomizu 1969, Ch. X Sec. 3).

We now have all the necessary tools to start computing geodesics on Riemannian homogeneous spaces. If we have a Lie group G with a compact subgroup H we can put a left-G-invariant and $\mathrm{Ad}(H)$-invariant

metric on G. This metric descends into a left-invariant metric on G/H making $\pi\colon G \to G/H$ a Riemannian submersion. Hence, if we know how to compute geodesics for this metric on G, we know that the geodesics on G/H are exactly projection of the geodesics on G with initial conditions on $\mathfrak{m}$, where $\mathfrak{m}$ is the orthogonal complement of $\mathfrak{h}$.

The process to compute geodesics with initial conditions $(gH, v) \in T(G/H)$ would be as follows:

1. Identify v with an element in $\mathfrak{m}$ as follows: $v_{\mathfrak{m}} \coloneqq (\mathrm{d}L_{g^{-1}})_g((\mathrm{d}\pi|_{\mathcal{H}_g})^{-1}(v)) \in \mathfrak{m}$.
2. Compute the geodesic $\gamma_e(t)$ on G with horizontal initial conditions $(e, v_{\mathfrak{m}})$.
3. Transport $\gamma_e(t)$ to T_gG using the isometry L_g: $\gamma_g \coloneqq (L_g)_*(\gamma_e)$.
4. Project the geodesic back to the manifold: $\pi \circ \gamma_g$.

Note that in the first step we have implicitly chosen an element in $\pi^{-1}(gH) \cong H$—which we have conveniently denoted by g—as the element in the fibre onto whose tangent space to lift v. As we mentioned before, this does not change the geodesics, as the metric is $\mathrm{Ad}(H)$-invariant.

The only step missing in implementing the plan above is the second one. In other words, we just need examples of reductive homogeneous spaces on which we can compute horizontal geodesics. To do so, we first need the following proposition which will allow us to translate the Lie derivative on the manifold to the Lie derivative on the Lie algebra via left-invariant vector fields.

Proposition 3.4.1. *Let G be a Lie group. The map $X \mapsto X^L$ is a Lie algebra homomorphism*

$$[X, Y]^L = [X^L, Y^L] \qquad \forall X, Y \in \mathfrak{g}.$$

Proof. Note that $X^{\mathrm{L}} = (\mathrm{d}L_g)_e(X) = \left.\dfrac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} g\exp\mathrm{m}(tX)$ comes from the differential of a right action evaluated at g. In other words, $X^{\mathrm{L}} = \xi(X)$ and the result follows from Proposition 3.2.19. □

Now, we may compute the Levi-Civita connection for a left-invariant metric on G.

Proposition 3.4.2. *Let g be a left-invariant metric on G. Then, the Levi-Civita connection at the identity takes the form*

$$(\nabla_{X^L} Y^L)_e = \frac{1}{2}[X, Y] + U(X, Y) \qquad \forall X, Y \in \mathfrak{g},$$

where U is the symmetric bilinear form on $\mathfrak{g}$ defined by

$$2g(U(X, Y), Z) = -g(X, [Y, Z]) - g(Y, [X, Z]) \qquad \forall X, Y, Z \in \mathfrak{g},$$

or, more succinctly,

$$U(X, Y) \coloneqq -\frac{1}{2}(\mathrm{ad}^*_X(Y) + \mathrm{ad}^*_Y(X)) \qquad \forall X, Y \in \mathfrak{g},$$

where ad^*_X is the adjoint with respect to the inner product on $\mathfrak{g}$.

Proof. By the Koszul formula (see *e.g.*, Petersen 2016, Thm. 2.2.2), for any $X, Y, Z \in \mathfrak{g}$

$$\begin{aligned} 2g(\nabla_{X^{\mathrm{L}}} Y^{\mathrm{L}}, Z^{\mathrm{L}}) = {} & X^{\mathrm{L}}(g(Y^{\mathrm{L}}, Z^{\mathrm{L}})) + Y^{\mathrm{L}}(g(X^{\mathrm{L}}, Z^{\mathrm{L}})) - Z^{\mathrm{L}}(g(X^{\mathrm{L}}, Y^{\mathrm{L}})) \\ & - g(X^{\mathrm{L}}, [Y^{\mathrm{L}}, Z^{\mathrm{L}}]) - g(Y^{\mathrm{L}}, [X^{\mathrm{L}}, Z^{\mathrm{L}}]) + g(Z^{\mathrm{L}}, [X^{\mathrm{L}}, Y^{\mathrm{L}}]). \end{aligned}$$

Since $X^{\mathrm{L}}, Y^{\mathrm{L}}, Z^{\mathrm{L}}$ are left-invariant fields, the function $p \mapsto g(X^{\mathrm{L}}, Y^{\mathrm{L}})_p$ is constant, so the first three terms are zero. Using Proposition 3.4.1 and evaluating at the identity we get the formula. □

Remark 3.4.3. It is not true in general that $\nabla_{X^{\mathrm{L}}} Y^{\mathrm{L}}$ is itself left-invariant due to the factor U. This is the reason for which we could compute the connection just at the identity.

Following this proposition, if we have a Riemannian homogeneous space G/H, the vector bundle morphism $\mathrm{d}\pi\colon \mathfrak{m} \times G \to T(G/H)$ along π induces the mapping

$$X^\star \coloneqq \mathrm{d}\pi(X^{\mathrm{L}}). \qquad X \in \mathfrak{m}$$

With these vector fields, we may compute the connection at $eH \in G/H$.

Proposition 3.4.4. *Let $(G/H, g)$ be a Riemannian homogeneous space then*

$$(\nabla_{X^\star} Y^\star)_{eH} = \frac{1}{2}([X,Y]_{\mathfrak{m}})^\star_{eH} + (U_{\mathfrak{m}}(X,Y))^\star_{eH} \qquad \forall X, Y \in \mathfrak{m}$$

where $U_{\mathfrak{m}}\colon \mathfrak{m} \times \mathfrak{m} \to \mathfrak{m}$ is the projection of the bilinear form U from Proposition 3.4.2 onto $\mathfrak{m}$.

Proof. The result follows from the expression for the Koszul formula as in Proposition 3.4.2 after noting that $\mathrm{d}\pi(X^{\mathrm{L}}) = \mathrm{d}\pi((X_{\mathfrak{m}})^{\mathrm{L}})$ so that $[X^\star, Y^\star] = ([X,Y]_{\mathfrak{m}})^\star$. □

This proposition suggests the following definition:

Definition 3.4.5. A **naturally reductive homogeneous space** is a Riemannian homogeneous space for which $U_{\mathfrak{m}} \equiv 0$, or, equivalently, if the projection onto $\mathfrak{m}$ of the adjoint $(\mathrm{ad}_X)_{\mathfrak{m}}\colon \mathfrak{m} \to \mathfrak{m}$ is skew-symmetric for every $X \in \mathfrak{m}$. In symbols,

$$\langle [X,Y]_{\mathfrak{m}}, Z\rangle = -\langle Y, [X,Z]_{\mathfrak{m}}\rangle \qquad \forall X, Y, Z \in \mathfrak{m}.$$

Remark 3.4.6 (Pseudo-Riemannian metrics on G). The definition of a naturally reductive space just depends on ad^*, which, in turn, just depends on the choice of reductive structure $\mathfrak{g} = \mathfrak{h} \oplus \mathfrak{m}$ and a suitable inner product on $\mathfrak{m}$. In particular, as we lift the metric from M to G, we have a choice on the $\mathrm{Ad}(H)$-invariant bilinear map on $\mathfrak{h}$. Throughout this chapter, we have assumed that this bilinear map was an inner product, but it is possible to have a pseudo-Riemannian left-G-invariant and right-H-invariant metric on G that descends into a Riemannian metric on G/H. This setting includes examples such as the different non-compact Grassmannian and general non-compact symmetric spaces. For simplicity, we will not explore these spaces, as, to the best of our knowledge, these spaces are not of great interest in the setting of optimisation, with the notable exception of the hyperbolic space.

Remark 3.4.7 (A geometric motivation for naturally reductive geometric spaces). We saw in Remark 3.3.17 that reductive homogeneous spaces have a distinguished principal connection. For a homogeneous space, the associated bundle induced by the isotropy representation is the tangent bundle of G/H, that is, $G \times_H T_{x_0} M \cong T(G/H)$. Hence, this principal connection defines a parallel transport system on G/H, which is given for a curve $\gamma(t)$ and an horizontal lift of it $g(t) \in G$ by

$$\begin{aligned}\mathbb{P}_{\gamma,t}\colon T_{\gamma(0)}G/H &\to T_{\gamma(t)}G/H\\ X &\mapsto \mathrm{d}(\pi \circ L_{g(t)} \circ L_{g(0)^{-1}})_{g(0)}((\mathrm{d}\pi|_{\mathcal{H}_{g(0)}})^{-1}(X)).\end{aligned}$$

The associated affine connection to this parallel transport system is called the **canonical connection**. This affine connection is given at the identity for a curve $\gamma(0) = eH$, $\dot\gamma(0) = X$ by

$$(\nabla_{X^\star} Y^\star)_{eH} \coloneqq \frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \mathbb{P}^{-1}_{\gamma,t}(Y^\star(t)) = ([X,Y]_{\mathfrak{m}})^\star_{eH}.$$

This affine connection is left-invariant and complete, with geodesics of the form $\gamma(t) = \pi(g\exp(tX))$ for $X \in \mathfrak{m}$ and $g \in G$ by construction. This connection has non-zero torsion given by the formula $T(X,Y)_{eH} = ([X,Y]_{\mathfrak{m}})^{\star}_{eH}$ (Kobayashi and Nomizu 1969, Ch. X Thm. 2.6).

For any affine connection ∇ we can define a torsion-free connection $\widetilde{\nabla}$ with the same geodesics by subtracting a half of its torsion from it, $\widetilde{\nabla} \coloneqq \nabla - \frac{1}{2}T$ (Ambrose, Palais, and I. M. Singer 1960). By applying this construction to the canonical connection, given that the torsion is also left-invariant, we get a torsion-free left-invariant affine connection on G/H with the same geodesics as the canonical connection, which is called the **natural torsion-free connection**. Now, on a Riemannian homogeneous space we have two distinguished torsion-free connections, namely the Levi-Civita connection associated to the metric and the natural torsion-free connection. Naturally reductive homogeneous spaces solve the question of when do these two connections agree. Since g is left-invariant, its Levi-Civita connection ∇^{LC} will also be left-invariant. Since $\widetilde{\nabla}$ is also left-invariant, we will have that $\widetilde{\nabla} = \nabla^{\mathrm{LC}}$ if and only if they agree at eH on the geodesics of $\widetilde{\nabla}$, that is, on the vector fields $X^{\star}$. But Proposition 3.4.4 gives a formula for ∇^{LC} at eH, from which we deduce that $\widetilde{\nabla} = \nabla^{\mathrm{LC}}$ if and only if $U_{\mathfrak{m}} \equiv 0$. In other words, the connection of a Riemannian homogeneous space (M,g) is the natural torsion-free connection of its reductive structure if and only if (M,g) is naturally reductive. This was the original motivation for their introduction by Nomizu. Note that, in general, g could be a pseudo-Riemannian metric, making (M,g) a pseudo-Riemannian homogeneous space. We restrict our attention to the Riemannian setting, as, for optimisation, we need a metric space structure on M to be able to prove convergence results.

Remark 3.4.8 (Killing vector fields). Sometimes, the formulas for the connection in Proposition 3.4.4 are given in terms of Killing vector fields on a Riemannian homogeneous space (M,g). In that case, letting $G = \mathrm{Iso}(M,g)$, and fixing a point $p \in M$, it is possible to identify each $X \in \mathfrak{m}$ with a Killing vector field X^{+} on G/H that does not vanish at p. In this case, this assignment is a Lie algebra antihomomorphism $[X,Y]^{+} = -[X^{+},Y^{+}]$, and the formula for the Levi-Civita connection of two Killing vector fields is the same but with a minus sign. This picture is particularly simple in the setting of naturally reductive homogeneous spaces as Killing vector fields are then given by the projection of *right*-invariant vector fields on G, as will follow from Proposition 3.4.10. On the other hand, the left-invariant vector fields are more amenable for the computation of geodesics.

The formula for the Levi-Civita connection of the projection of a left-G-invariant right-H-invariant metric on G/H takes a particularly simple form when $(G/H,g)$ is naturally reductive.

Proposition 3.4.9. *Let G be a Lie group and H a compact Lie subgroup. Let g be a left-G-invariant $\mathrm{Ad}(H)$-invariant metric on G so that $(G/H,\pi_*(g))$ is a naturally reductive homogeneous space. Then*

$$\nabla_{X^{L}}Y^{L} = \frac{1}{2}[X,Y]^{L} \qquad \forall X,Y \in \mathfrak{m}.$$

Proof. By the Koszul formula we have that for $X,Y,Z \in \mathfrak{g}$

$$2g(\nabla_{X^{L}}Y^{L},Z^{L}) = -g(X^{L},[Y^{L},Z^{L}]) - g(Y^{L},[X^{L},Z^{L}]) + g(Z^{L},[X^{L},Y^{L}]).$$

By linearity, if we prove that the formula holds for $Z \in \mathfrak{m}$ and for $Z \in \mathfrak{h}$ then we are done.

Assume that $X, Y \in \mathfrak{m}$ and $Z \in \mathfrak{m}$. By the naturally reductive condition and the left-invariance of g the first two terms cancel getting

$$2g(\nabla_{X^{\mathrm{L}}} Y^{\mathrm{L}}, Z^{\mathrm{L}}) = g([X, Y]^{\mathrm{L}}, Z^{\mathrm{L}})$$

so the formula holds on its horizontal part.

Assume now that $Z \in \mathfrak{h}$. Since g is $\mathrm{Ad}(H)$-invariant, computing the derivative at zero of

$$g(\mathrm{Ad}_{\mathrm{expm}(tZ)}(X), \mathrm{Ad}_{\mathrm{expm}(tZ)}(Y)) = g(X, Y),$$

we get that ad_Z is skew-symmetric so the two first terms of the Koszul formula also cancel, and the formula holds on its vertical part as well. □

We are now ready to compute the geodesics for naturally reductive homogeneous spaces, and, in particular, for compact Lie groups with a bi-invariant metric.

Proposition 3.4.10 (Geodesics on naturally reductive spaces). *Let G be a Lie group with a left-invariant metric such that $U \equiv 0$. The geodesic at the identity γ_e with initial condition $X \in \mathfrak{g}$ is given by the one-parameter subgroups $\gamma_e(t) = \mathrm{expm}(tX)$.*

More generally, if G/H is a naturally reductive homogeneous space, the geodesics at an arbitrary point $gH \in G/H$ are of the form

$$\gamma_{gH}(t) = \pi(g \,\mathrm{expm}(tX_{\mathfrak{m}})) = g \,\mathrm{expm}(tX_{\mathfrak{m}})H.$$

Proof. If $U \equiv 0$, we have $\nabla_{X^{\mathrm{L}}} Y^{\mathrm{L}} = \frac{1}{2}[X, Y]^{\mathrm{L}}$ (Proposition 3.4.2). In particular, the vector field X^{L} is self-parallel. The integral curves of a left-invariant vector field is given by the Lie exponential, by definition of the Lie exponential.

The same argument goes through after selecting $X, Y \in \mathfrak{m}$ and applying the formula for the connection on two left-invariant vector fields on a naturally reductive homogeneous space (Proposition 3.4.9). Since these are horizontal geodesics and $\pi \colon G \to G/H$ is a Riemannian submersion we get the result. □

A large family of naturally reductive homogeneous spaces comes from having a Lie group G together with a bi-invariant metric.

Definition 3.4.11. A **normal Riemannian homogeneous space** is a Riemannian homogeneous space (M, g) with a presentation G/H such that the metric is induced by a bi-invariant metric on G.

Normal Riemannian homogeneous spaces are examples of naturally reductive spaces.

Proposition 3.4.12. *Let G be a Lie group with a bi-invariant metric, then*

$$\langle \mathrm{ad}_X(Y), Z \rangle = -\langle Y, \mathrm{ad}_X(Z) \rangle \qquad \forall X, Y, Z \in \mathfrak{g},$$

or, more compactly, $\mathrm{ad}_g^* = -\mathrm{ad}_g$ for every $g \in G$. In particular, a normal Riemannian homogeneous space is naturally reductive and its connection on left-invariant vector fields is given by

$$\nabla_{X^L} Y^L = \frac{1}{2}[X, Y]^L \qquad \forall X, Y \in \mathfrak{g}.$$

Proof. Differentiate $\langle \mathrm{Ad}_{\exp(tX)}(Y), \mathrm{Ad}_{\exp(tX)}(Z)\rangle = \langle Y, Z\rangle$ at $t = 0$. For the formula for the connection, consider the Koszul formula and use that $\mathrm{ad}_X^* = -\mathrm{ad}_X$ for every $X \in \mathfrak{g}$. □

Remark 3.4.13. We already proved in Corollary 3.3.10 that compact groups admit bi-invariant metrics. Using the same construction, without the need of averaging, one proves that Abelian groups also admit a bi-invariant metric. A basic result in Lie group theory says that any Abelian group is the direct product of a torus with $\mathbb{R}^n$. Hence, Lie groups of the form $G \cong K \times \mathbb{R}^n$, for K compact, admit a bi-invariant metric. In fact, these are the only groups with bi-invariant metrics.

Theorem 3.4.14 (Classification of groups with bi-invariant metrics). *A Lie group admits a bi-invariant metric if and only if it is isomorphic to a direct product $G \times H$, with G compact and H Abelian.*

Proof. See (Milnor 1976, Lemma 7.5). □

In the case when G is compact and semisimple, the bi-invariant scalar product is unique up to a scaling factor, and it is given by $-B$, where B is the Killing form. These normal homogeneous spaces are called **standard Riemannian homogeneous spaces** and this metric is called the **canonical metric**.

3.5 Matrix Manifolds

All the theory developed in this chapter comes together particularly nicely when G is a matrix Lie group. In this case, if G/H is a naturally reductive homogeneous space, we may simplify the computation of geodesics to the computation of the Lie exponential on G. It turns out that the Lie exponential is just the exponential of matrices, so this effectively transforms all the necessary computations into linear algebra problems. In this section we will see how some of the most important families of manifolds used in optimisation fall within the different categories of manifolds described before.

We start by recalling the definition of a matrix Lie group.

Definition 3.5.1. A complex (resp. real) **matrix Lie group** is a closed subgroup of $\mathrm{GL}(n, \mathbb{C})$ (resp. $\mathrm{GL}(n, \mathbb{R})$).

On matrix Lie groups, the Lie exponential is just the regular exponential of matrices.

Proposition 3.5.2. *The Lie exponential map on a real or complex matrix Lie group is given by the exponential of matrices.*

Proof. The matrix exponential $\gamma(t) = \exp(tX)$ can be expressed as the solution of the matrix differential equation

$$\gamma'(t) = X\gamma(t) \qquad \gamma(0) = \mathrm{I}_n$$

for $X \in \mathbb{C}^{n\times n} = \mathfrak{gl}(n, \mathbb{C})$. Since, for matrix groups, $(\mathrm{d}L_X)_{\mathrm{I}_n}(A) = XA$, we have that this is exactly the differential equation that defines the exponential map as an integral curve of a left-invariant vector field. □

Let us now show how the whole theory for computing geodesics is put together in the case of some commonly used manifolds in optimisation.

3.5.1 The Stiefel manifold

We define the **Stiefel manifold** as the space of matrices with orthonormal columns

$$\mathrm{St}(n,k) \coloneqq \{X \in \mathbb{R}^{n\times k} \mid X^\intercal X = \mathrm{I}_k\} \qquad 1 \leq k \leq n.$$

Throughout this section we will assume $n > k$ as, for $n = k$, $\mathrm{St}(n,k) \cong \mathrm{O}(n)$, which is disconnected. Hence, when doing orthogonal optimisation on $\mathbb{R}^{n\times n}$, we will just work on $\mathrm{SO}(n)$.

Calling something a manifold does not make it a manifold, so let us show that it is in fact a homogeneous space by showing that it has a transitive action by a Lie group. This shows, in particular, that it is a smooth manifold (Proposition 3.3.4).

Consider the left action of $\mathrm{SO}(n)$ on $\mathrm{St}(n,k)$ by multiplication on the left. This action is transitive, since for $U_1, U_2 \in \mathrm{St}(n,k)$ there exists a matrix $\overline{U} \in \mathrm{SO}(n)$ such that $L_{\overline{U}}(U_1) = U_2$. To see this, complete U_1, U_2 into two orthogonal matrices $\overline{U}_1, \overline{U}_2 \in \mathrm{SO}(n)$ and consider $\overline{U} \coloneqq \overline{U}_1^{-1}\overline{U}_2$.

Let $x_0 = \begin{pmatrix} \mathrm{I}_k \\ 0_{n-k,k} \end{pmatrix} \in \mathrm{St}(n,k)$. The isotropy group of the action above at x_0—the elements of $\mathrm{SO}(n)$ that fix x_0—is

$$H = \mathrm{I}_k \otimes \mathrm{SO}(n-k) = \left\{ \begin{pmatrix} \mathrm{I}_k & 0_{k,n-k} \\ 0_{n-k,k} & Q \end{pmatrix} \;\middle|\; Q \in \mathrm{SO}(n-k) \right\}.$$

Since $H \cong \mathrm{SO}(n-k)$, we have that $\mathrm{St}(n,k) = \mathrm{SO}(n)/\mathrm{SO}(n-k)$, where the projection $\pi\colon \mathrm{SO}(n) \to \mathrm{St}(n,k)$ maps an orthogonal matrix to its first k columns (Proposition 3.3.4).[1]

Consider the inner product on $\mathfrak{so}(n) = \mathrm{Skew}(n)$ given by $\overline{g}(A,B)_e \coloneqq \frac{1}{2}\operatorname{tr}(A^\intercal B)$.[2] Since this is $\mathrm{Ad}(\mathrm{SO}(n))$-invariant, it extends naturally to a bi-invariant metric on all of $\mathrm{SO}(n)$ via left-invariant vector fields, setting $\overline{g}(UA,UB)_U \coloneqq \overline{g}(A,B)_e$ (Corollary 3.3.10). Since this metric on the total space is bi-invariant, $\mathrm{St}(n,k)$ is a normal Riemannian homogeneous space (Proposition 3.4.12).

Denote by $\overline{U} = \begin{pmatrix} U & U_\perp \end{pmatrix}$ the decomposition of a matrix $\overline{U} \in \mathrm{SO}(n)$ into its first k columns and the rest so that $\pi(\overline{U}) = U$. We want to compute the Riemannian exponential of a matrix $\Delta \in T_U \mathrm{St}(n,k)$. To do so, we will start by computing $\mathfrak{h}$ and setting $\mathfrak{m} = \mathfrak{h}^\perp$ (Proposition 3.3.18)

$$\mathfrak{h} = 0_{k,k} \otimes \mathfrak{so}(n-k) \qquad \mathfrak{m} = \left\{ \begin{pmatrix} S & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} \;\middle|\; S \in \mathfrak{so}(k), A \in \mathbb{R}^{(n-k)\times k} \right\}.$$

As the space is reductive, we can represent a tangent vector $\Delta \in T_U \mathrm{St}(n,k)$ by a vector in $X_\mathfrak{m} \in \mathfrak{m}$ as $\Delta = (\mathrm{d}\pi)_{\overline{U}}(\overline{U}X_\mathfrak{m})$, and the exponential map is given by

$$\exp_U(\Delta) = \pi(\overline{U}\operatorname{expm}(X_\mathfrak{m})) = \overline{U}\operatorname{expm}\begin{pmatrix} S & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} P_{n,k}, \tag{3.3}$$

where $P_{n,k} \coloneqq \begin{pmatrix} \mathrm{I}_k \\ 0_{k,n-k} \end{pmatrix}$ is the matrix representation of the projection onto the first k columns (Proposition 3.4.10). The differential of π at a matrix B is then given by $\mathrm{d}\pi(B) = BP_{n,k}$, so

$$\Delta = \mathrm{d}\pi(\overline{U}X_\mathfrak{m}) = US + U_\perp A \in T_U \mathrm{St}(n,k).$$

In view of this, we may represent a tangent space to the Stiefel manifold as

$$T_U \mathrm{St}(n,k) = \{US + U_\perp A \mid S \in \mathfrak{so}(k), A \in \mathbb{R}^{(n-k)\times k}\}.$$

[1] Note that here we are writing π to refer to the map that is the composition of the canonical projection onto $\mathrm{SO}(n)/\mathrm{SO}(n-k)$ and the isotropy action at x_0 to avoid unnecessarily cluttering the notation.

[2] The factor of $\frac{1}{2}$ is there so that $\mathrm{St}(n,1)$ is isometric to the round sphere of radius 1.

Remark 3.5.3 (The choice of completion $U_\perp$). Note that $\overline{U} \in \pi^{-1}(U) \cong H$ (Proposition 3.3.1). Hence, a choice of $U_\perp$ to complete U into an orthogonal matrix $\overline{U}$ accounts for a choice of $h \in H \cong \mathrm{SO}(n-k)$, which would transform $\mathfrak{m}$ as $\mathrm{Ad}_h(\mathfrak{m})$. In this case, as we are fixing a basis of $\mathfrak{m}$ and we are computing the basis of $T_U \operatorname{St}(n,k)$ in terms of it, this choice materialises as a change of basis on $T_U \operatorname{St}(n,k)$. This change of basis of the tangent space does not affect the computation of the geodesics or any other metric-related object, as the metric comes from an $\mathrm{Ad}(H)$-invariant inner product on $\mathfrak{g}$.

We could directly use (3.3) to compute the exponential map on $\operatorname{St}(n,k)$, but this formula has two problems. First, it requires the matrix exponential of the $n \times n$ matrix $X_\mathfrak{m}$. As this matrix has rank at most $2k$, for small k, we would like to have a formula that scales down with k. Second, we need to compute $U_\perp$ if we only have access to U. This is equivalent to computing a QR decomposition of an $n \times (n-k)$ matrix, which we would like to avoid.

Assume for the rest of the section that $n > 2k$. We now present a method proposed in (Gallier and Quaintance 2020) that avoids the first problem, but not the second. The idea is to decompose A into its thin QR decomposition

$$A = Q_A R_A \qquad Q_A \in \operatorname{St}(n-k,k), R_A \in \mathbb{R}^{k\times k}.$$

We then compute

$$\begin{aligned}\overline{U} \operatorname{expm} \underbrace{\begin{pmatrix} S & -Q_A R_A^\intercal \\ Q_A R_A & 0_{n-k,n-k} \end{pmatrix}}_{n\times n} P_{n,k} &= \overline{U} \begin{pmatrix} \mathrm{I}_k & 0_{k,k} \\ 0_{n-k,k} & Q_A \end{pmatrix} \operatorname{expm} \begin{pmatrix} S & -R_A^\intercal \\ R_A & 0_{k,k} \end{pmatrix} \begin{pmatrix} \mathrm{I}_k & 0_{k,n-k} \\ 0_{k,k} & Q_A^\intercal \end{pmatrix} P_{n,k} \\ &= \begin{pmatrix} U & U_\perp Q_A \end{pmatrix} \operatorname{expm} \underbrace{\begin{pmatrix} S & -R_A^\intercal \\ R_A & 0_{k,k} \end{pmatrix}}_{2k\times 2k} P_{n,k},\end{aligned}$$

where in the first equality we have used that, since expm is analytic,

$$\operatorname{expm}(BXB^\intercal) = B \operatorname{expm}(X) B^\intercal \qquad \forall B \in \operatorname{St}(n,k), X \in \mathbb{R}^{k\times k}.$$

The cost of this formula is a thin QR of an $(n-k)\times k$ matrix, the exponential of a $2k \times 2k$ matrix and the computation of $U_\perp$, which can be done through another thin QR of an $n \times (n-k)$ matrix.

We will now show how to avoid the computation of $U_\perp$. This may be achieved via a trick first introduced in (Edelman, Arias, and Smith 1998). In their paper they just outline the method and do not explicit the computations. We will see that, given the abstract theory presented before, the computations will follow naturally.

The trick to avoid the computation of $U_\perp$ will come from finding a different parametrisation of tangent spaces of $\operatorname{St}(n,k)$. To do this, consider the natural embedding of $\operatorname{St}(n,k)$ into $\mathbb{R}^{n\times k}$. For a given matrix on the ambient space $C \in \mathbb{R}^{n\times k}$, and any point $U \in \operatorname{St}(n,k)$, the canonical inner product of $\mathbb{R}^{n\times k}$ gives a decomposition into the normal and tangential part of C

$$C = \pi_U(C) + \pi_U^\perp(C).$$

We will use the projection π_U to parametrise the tangent space in terms of matrices in $\mathbb{R}^{n\times k}$. In order to do this, we shall compute formulas for these projections. By differentiating the formula $U^\intercal U = \mathrm{I}_k$ that

defines $\mathrm{St}(n,k)$ we get an implicit formula for the tangent space of $\mathrm{St}(n,k)$

$$T_U \,\mathrm{St}(n,k) = \{\Delta \in \mathbb{R}^{n\times k} \mid U^\intercal \Delta \in \mathfrak{so}(k)\}.$$

It is direct to see that this implicit representation is equivalent to the explicit representation of the tangent space of $\mathrm{St}(n,k)$ given before. Now, consider a matrix N in the normal space $N_U \,\mathrm{St}(n,k)$, where *normal* is taken with respect to the canonical inner product of $\mathbb{R}^{n\times k}$. $N \in \mathbb{R}^{n\times k}$ is in the normal space if and only if $\langle N, \Delta\rangle = 0$ for every $\Delta \in T_U \,\mathrm{St}(n,k)$, so we may choose a matrix of the form $N = UB$ for an arbitrary symmetric matrix B, as

$$\langle \Delta, UB\rangle = \mathrm{tr}(\Delta^\intercal UB) = \langle U^\intercal \Delta, B\rangle = 0 \qquad \forall \Delta \in T_U \,\mathrm{St}(n,k),$$

where we have used that symmetric and skew-symmetric matrices are orthogonal. Counting dimensions, we see that every matrix in the normal space is of this form, concluding that the normal space can be expressed as

$$N_U \,\mathrm{St}(n,k) = \{UB \in \mathbb{R}^{n\times k} \mid B \in \mathrm{Sym}(k)\}.$$

From here, we get the formulas for the normal and tangential projections from $\mathbb{R}^n$

$$\begin{aligned}
\pi_U^\perp(C) &= U\mathrm{sym}(U^\intercal C),\\
\pi_U(C) &= C - \pi_U^\perp(C) = U\mathrm{skew}(U^\intercal C) + (\mathrm{I}_n - UU^\intercal)C,
\end{aligned}$$

where we have written $\mathrm{sym}(C) \coloneqq \frac{1}{2}(C + C^\intercal)$, $\mathrm{skew}(C) \coloneqq \frac{1}{2}(C - C^\intercal)$ for the orthogonal projections onto the symmetric and skew-symmetric matrices.

Using π_U, we can represent the matrix $\Delta \in T_U \,\mathrm{St}(n,k)$ as the projection of a matrix $C \in \mathbb{R}^{n\times k}$,

$$\Delta = U\mathrm{skew}(U^\intercal C) + (\mathrm{I}_n - UU^\intercal)C.$$

This decomposition is not unique, as there are many matrices $C \in \mathbb{R}^{n\times k}$ that project onto the same matrix $\pi_U(C)$. Comparing this decomposition with that in terms of $\mathfrak{m}$, since both are orthogonal decompositions, we see that they are related by the equations

$$U_\perp A = (\mathrm{I}_n - UU^\intercal)C \qquad S = \mathrm{skew}(U^\intercal C).$$

Finally, we just have to apply the same QR trick that we used before:

$$\begin{aligned}
\overline{U}\,\mathrm{expm}\underbrace{\begin{pmatrix} S & -A^\intercal \\ A & 0_{n-k,n-k}\end{pmatrix}}_{n\times n} P_{n,k} &= \begin{pmatrix} U & \mathrm{I}_n\end{pmatrix}\begin{pmatrix} \mathrm{I}_k & 0_{k,k} \\ 0_{n,k} & U_\perp\end{pmatrix}\mathrm{expm}\begin{pmatrix} S & -A^\intercal \\ A & 0_{n-k,n-k}\end{pmatrix} P_{n,k}\\
&= \begin{pmatrix} U & \mathrm{I}_n\end{pmatrix}\mathrm{expm}\begin{pmatrix} \mathrm{skew}(U^\intercal C) & -((\mathrm{I}_n - UU^\intercal)C)^\intercal \\ (\mathrm{I}_n - UU^\intercal)C & 0_{k,k}\end{pmatrix} P_{n,k}\\
&= \begin{pmatrix} U & Q_C\end{pmatrix}\mathrm{expm}\underbrace{\begin{pmatrix} \mathrm{skew}(U^\intercal C) & -R_C^\intercal \\ R_C & 0_{k,k}\end{pmatrix}}_{2k\times 2k} P_{n,k},
\end{aligned}$$

where we have written $Q_C \in \mathrm{St}(n,k), R_C \in \mathbb{R}^{k\times k}$ for the QR decomposition of $(\mathrm{I}_n - UU^\intercal)C \in \mathbb{R}^{n\times k}$. This formula is entirely in terms of C and U, circumventing the computation of $U_\perp$. Furthermore, it just involves the exponential of a $2k \times 2k$ matrix. By doing this, we inadvertently lose the differentiability of

the process, as the QR decomposition is not smooth for matrices that are not full rank. This will render these tricks useless for the applications in this thesis, but they are rather interesting in their own sake nonetheless.

We summarise this section in the following proposition.

Proposition 3.5.4 (Exponential map on the Stiefel manifold). *Let $\overline{U} \in \mathrm{SO}(n)$ and $U = \pi(\overline{U}) \in \mathrm{St}(n,k)$ be points on the total space and the manifold respectively. Let $X_{\mathfrak{m}} = \left(\begin{smallmatrix} S & -A^\intercal \\ A & 0_{n-k,n-k} \end{smallmatrix}\right) \in \mathfrak{m}$ be the matrix on the Lie algebra that defines the tangent vector (matrix) $\Delta = (\mathrm{d}\pi)_{\overline{U}}(\overline{U} X_{\mathfrak{m}}) \in T_U \mathrm{St}(n,k)$. The Riemannian exponential map with respect to the canonical metric is given by*

$$\exp_U(\Delta) = \pi(\overline{U}\operatorname{expm}(X_{\mathfrak{m}})) = \overline{U}\operatorname{expm}\begin{pmatrix} S & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} P_{n,k},$$

where $P_{n,k} \coloneqq \left(\begin{smallmatrix} \mathrm{I}_k \\ 0_{k,n-k} \end{smallmatrix}\right)$ is the matrix representation of the projection onto the first k columns.

If we represent $X_{\mathfrak{m}}$ using a matrix $C \in \mathbb{R}^{n\times k}$ so that $U_\perp A = (\mathrm{I}_n - UU^\intercal)C$ and $S = \operatorname{skew}(U^\intercal C)$, using a QR decomposition of $Q_C R_C = (\mathrm{I}_n - UU^\intercal)C \in \mathbb{R}^{n\times k}$, we can write the exponential map as

$$\exp_U(\Delta) = \begin{pmatrix} U & Q_C \end{pmatrix} \operatorname{expm}\begin{pmatrix} \operatorname{skew}(U^\intercal C) & -R_C^\intercal \\ R_C & 0_{k,k} \end{pmatrix} P_{n,k}.$$

Furthermore, this formula is only differentiable with respect to C whenever $(\mathrm{I}_n - UU^\intercal)C$ has maximal rank k.

3.5.2 The Grassmannian

The Grassmannian $\mathrm{Gr}(n,k)$ is the set of all k-dimensional subspaces of $\mathbb{R}^n$. Its construction as a Riemannian homogeneous space is essentially the same as in the case of the Stiefel manifold, so we will not give as many details here as we gave in the previous section.

The homogeneous space structure of the Grassmannian, analogously to the case of the Stiefel manifold, comes from considering the isotropy group of the transitive left $G = \mathrm{SO}(n)$ action on the class of x_0 with $x_0 = \left(\begin{smallmatrix} \mathrm{I}_k \\ 0_{n-k,k} \end{smallmatrix}\right) \in \mathrm{St}(n,k)$. In this case $H = \mathrm{S}(\mathrm{O}(k) \otimes \mathrm{O}(n-k))$ and $\mathrm{Gr}(n,k) \cong \mathrm{SO}(n)/\mathrm{S}(\mathrm{O}(k)\times\mathrm{O}(n-k))$, where

$$\mathrm{S}(\mathrm{O}(k) \otimes \mathrm{O}(n-k)) \coloneqq (\mathrm{O}(k) \otimes \mathrm{O}(n-k)) \cap \mathrm{SO}(n).$$

Considering the $\mathrm{Ad}(G)$-invariant inner product $\langle A_1, A_2\rangle = \frac{1}{2}\operatorname{tr}(A_1^\intercal A_2)$ on $\mathfrak{g}$, we get

$$\mathfrak{h} = \mathfrak{so}(k) \otimes \mathfrak{so}(n-k) \qquad \mathfrak{m} = \left\{ \begin{pmatrix} 0_{k,k} & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} \;\middle|\; A \in \mathbb{R}^{(n-k)\times k} \right\}$$

and left-translating it into a bi-invariant metric on $\mathrm{SO}(n)$, it descends into a metric on $\mathrm{Gr}(n,k)$ that makes $\mathrm{Gr}(n,k)$ into a normal Riemannian homogeneous space.[3] The Riemannian exponential is then given by

$$\exp_{[U]}(A_{\mathfrak{m}}) = \pi(\overline{U}\operatorname{expm}(A_{\mathfrak{m}}))$$

for a $\overline{U} \in \mathrm{SO}(n)$ with $\pi(\overline{U}) = [U]$. The elements of the Grassmannian are typically represented by a representative of their class $U \in \mathrm{St}(n,k)$.

[3] The Grassmannian with this metric is not only a normal Riemannian homogeneous space but also a symmetric space, but we will not use this fact here.

We can stablish an isomorphism between $\mathrm{Gr}(n,k)$ and $\mathrm{Gr}(n,n-k)$ by sending a space to its complement. Hence, we will assume in what follows that $n \geq 2k$.

It is possible to compute the exponential on the Grassmannian by performing just one SVD of an $(n-k)\times k$ matrix, as noted in (Edelman, Arias, and Smith 1998). This is similar to the QR trick performed in the Stiefel manifold case. Let $A \in \mathbb{R}^{(n-k)\times k}$, and let $A_{\mathfrak{m}} \in \mathfrak{m}$ be the associated matrix on $\mathfrak{m}$. If $A = U_A \Sigma_A V_A^\intercal$ is a SVD of A with $U_A \in \mathrm{St}(n-k,k), \Sigma_A \in \mathbb{R}^{k\times k}, V_A \in \mathrm{SO}(k)$, then

$$A_{\mathfrak{m}} = \begin{pmatrix} 0_{k,k} & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} = \begin{pmatrix} V_A & 0_{k,n-k} \\ 0_{n-k,k} & U_A \end{pmatrix} \begin{pmatrix} 0_{k,k} & -\Sigma_A^\intercal \\ \Sigma_A & 0_{k,k} \end{pmatrix} \begin{pmatrix} V_A^\intercal & 0_{k,n-k} \\ 0_{n-k,k} & U_A^\intercal \end{pmatrix}.$$

Now, representing a k-subspace $[U] \in \mathrm{Gr}(n,k)$ giving a base $U \in \mathrm{St}(n,k)$ of it, if $U_\perp \in \mathrm{St}(n,n-k)$ is a completion of U into an orthogonal matrix,

$$\exp_{[U]}(A_{\mathfrak{m}}) = \begin{pmatrix} UV_A & U_\perp U_A \end{pmatrix} \begin{pmatrix} \cos(\Sigma_A) \\ \sin(\Sigma_A) \end{pmatrix} V_A^\intercal,$$

where Σ_A is a diagonal matrix, so the sine and cosine are applied element-wise to the diagonal of Σ_A.

Note that, if we consider a matrix $C \in \mathbb{R}^{n\times k}$, we may use the same trick as we did for the Stiefel manifold to avoid having to compute $\overline{U}$. Indeed, consider the SVD $U_C \Sigma_C V_C^\intercal = (\mathrm{I}_n - UU^\intercal)C$. We have that for an element $X = (\mathrm{I}_n - UU^\intercal)C \in T_{[U]}\,\mathrm{Gr}(n,k)$,

$$\exp_{[U]}(X) = \begin{pmatrix} UV_C & U_C \end{pmatrix} \begin{pmatrix} \cos(\Sigma_C) \\ \sin(\Sigma_C) \end{pmatrix} V_C^\intercal.$$

In GPU implementations, this method is hardly every faster than the naïve computation of the exponential of matrices, given that computing the SVD is orders of magnitude slower than computing a matrix exponential. Even then, it is still possible to perform the QR trick described in the previous section and compute the Riemannian exponential map on the Grassmannian by computing a QR decomposition $Q_C R_C = (\mathrm{I}_n - UU^\intercal)C$, so that

$$\exp_{[U]}(X) = \overline{U}\,\mathrm{expm}\begin{pmatrix} 0_{k,k} & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} P_{n,k} = \begin{pmatrix} U & Q_C \end{pmatrix} \mathrm{expm}\begin{pmatrix} 0_{k,k} & -R_C^\intercal \\ R_C & 0_{k,k} \end{pmatrix} P_{n,k}.$$

This trick for the Grassmannian was not presented in (Edelman, Arias, and Smith 1998).

We summarise this discussion in a proposition.

Proposition 3.5.5 (Exponential map on the Grassmannian manifold). *Let $\overline{U} \in \mathrm{SO}(n)$ and $[U] = \pi(\overline{U}) \in \mathrm{Gr}(n,k)$ be points on the total space and the manifold respectively with a representative $U \in \mathrm{St}(n,k)$. Let $X_{\mathfrak{m}} = \left(\begin{smallmatrix} 0_{k,k} & -A^\intercal \\ A & 0_{n-k,n-k} \end{smallmatrix}\right) \in \mathfrak{m}$ be the matrix on the Lie algebra that defines the tangent vector (matrix) $\Delta = (\mathrm{d}\pi)_{\overline{U}}(\overline{U}X_{\mathfrak{m}}) \in T_U\,\mathrm{Gr}(n,k)$. The Riemannian exponential map with respect to the canonical metric is given by*

$$\exp_{[U]}(\Delta) = \pi(\overline{U}\,\mathrm{expm}(X_{\mathfrak{m}})) = \overline{U}\,\mathrm{expm}\begin{pmatrix} 0_{k,k} & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} P_{n,k}$$

where $P_{n,k} \coloneqq \left(\begin{smallmatrix} \mathrm{I}_k \\ 0_{k,n-k} \end{smallmatrix}\right)$ is the matrix representation of the projection onto the first k columns.

If we represent $X_{\mathfrak{m}}$ using a matrix $C \in \mathbb{R}^{n\times k}$ so that $U_\perp A = (\mathrm{I}_n - UU^\intercal)C$ and $\mathrm{skew}(U^\intercal C) = 0$, using a QR decomposition and an SVD of $Q_C R_C = U_C \Sigma_C V_C^\intercal = (\mathrm{I}_n - UU^\intercal)C \in \mathbb{R}^{n\times k}$, we can write the exponential map as

$$\exp_{[U]}(\Delta) = \begin{pmatrix} UV_C & U_C \end{pmatrix} \begin{pmatrix} \cos(\Sigma_C) \\ \sin(\Sigma_C) \end{pmatrix} V_C^\intercal = \begin{pmatrix} U & Q_C \end{pmatrix} \mathrm{expm}\begin{pmatrix} \mathrm{skew}(U^\intercal C) & -R_C^\intercal \\ R_C & 0_{k,k} \end{pmatrix} P_{n,k},$$

respectively. Furthermore, this formula is only differentiable with respect to C whenever $(\mathrm{I}_n - UU^\intercal)C$ has maximal rank k, and in the case of the SVD whenever $X_\mathfrak{m}$ rank has, additionally, non-repeated singular values.

3.5.3 Symmetric positive definite matrices

The case of the symmetric positive definite matrices $\mathrm{Sym}^+(n)$ is slightly different from the previous two. We include all the details here as it is surprisingly difficult to find this derivation in the literature.[4]

This manifold has a transitive left action by the non-compact group of orientation preserving invertible matrices $\mathrm{GL}^+(n)$ given by

$$\mu\colon \mathrm{GL}^+(n) \times \mathrm{Sym}^+(n) \to \mathrm{Sym}^+(n)$$
$$P, A \mapsto PAP^\intercal$$

The action is well-defined, since $\langle Av, v\rangle > 0$ for every $v \in \mathbb{R}^n\backslash\{0\}$ if and only if $\langle PAP^\intercal v, v\rangle = \langle AP^\intercal v, P^\intercal v\rangle > 0$ for every $v \in \mathbb{R}^n\backslash\{0\}$ given that $P^\intercal \in \mathrm{GL}(n)$. It is direct to see that the action is transitive, since any matrix in $A \in \mathrm{Sym}^+(n)$ is diagonalisable as $A = Q\Sigma Q^\intercal$ for $Q \in \mathrm{SO}(n)$, so any matrix can be taken to the identity as $\mu(\Sigma^{-1/2}Q^\intercal, A) = \mathrm{I}_n$.

The isotropy group of this action is $H = \mathrm{SO}(n)$. Considering the canonical scalar product rescaled[5]

$$\langle\!\langle X, Y\rangle\!\rangle \coloneqq 4\langle X, Y\rangle = 4\,\mathrm{tr}(X^\intercal Y) \qquad X, Y \in \mathfrak{gl}(n) \cong \mathbb{R}^{n\times n}$$

we get that the split $\mathfrak{g} = \mathfrak{m} \oplus \mathfrak{h}$ is the usual split of a matrix into its symmetric and skew-symmetric parts via the projection with respect to the canonical product, that is, $\mathfrak{h} = \mathfrak{so}(n) \cong \mathrm{Skew}(n)$ and $\mathfrak{m} = \mathrm{Sym}(n)$. Furthermore, since this is an $\mathrm{Ad}(H)$-invariant product, we may extend it to a product on $\mathrm{GL}^+(n)$ via left translations as

$$\overline{g}(B_1, B_2)_P \coloneqq \langle\!\langle P^{-1}B_1, P^{-1}B_2\rangle\!\rangle \qquad P \in \mathrm{GL}^+(n), B_1, B_2 \in T_P\,\mathrm{GL}^+(n).$$

In this case, the isotropy representation defining the isomorphism in Proposition 3.3.4, rather than being a projection onto some components as it the case for the Stiefel manifold and the Grassmannian, is given by the map

$$\mu^{\mathrm{I}_n}\colon \mathrm{GL}^+(n)/\,\mathrm{SO}(n) \to \mathrm{Sym}^+(n)$$
$$P\cdot\mathrm{SO}(n) \mapsto PP^\intercal.$$

Therefore, the submersion from $\mathrm{GL}^+(n)$ onto $\mathrm{Sym}^+(n)$ associated to the action μ is given by

$$\mu^{\mathrm{I}_n}\circ\pi\colon \mathrm{GL}^+(n) \to \mathrm{Sym}^+(n)$$
$$P \mapsto PP^\intercal.$$

Differentiating at the identity, we get the linear isomorphism between $\mathfrak{m}$ and a tangent space of $\mathrm{Sym}^+(n)$

$$\mathrm{d}(\mu^{\mathrm{I}_n}\circ\pi\circ L_{A^{1/2}})_{\mathrm{I}_n}\colon \mathfrak{m} \to T_A\,\mathrm{Sym}^+(n)$$
$$\Delta \mapsto 2A^{1/2}\Delta A^{1/2}.$$

[4] In this case, one may take a shortcut since $\mathrm{Sym}^+(n)$ is not only a naturally reductive homogeneous space, but also a symmetric space. One may then use the theory of symmetric spaces to compute the geodesics, but this somewhat hides the ideas of the construction of a Riemannian submersion that underlay this computation.

[5] This rescaling makes up for a constant that appears when differentiating the action μ.

The construction of the metric on a naturally reductive homogeneous space goes on to make this map into a linear isometry to turn $\mu^{\mathrm{I}_n} \circ \pi$ into a Riemannian submersion. For this to happen, since this map has inverse

$$\begin{aligned} \eta_A \colon T_A \operatorname{Sym}(n) &\to \mathfrak{m} \\ \Delta &\mapsto \frac{1}{2} A^{-1/2} \Delta A^{-1/2} \end{aligned}$$

we have that the product on $T_A \operatorname{Sym}^+(n)$ that turns the projection into a submersion is

$$g(\Delta_1, \Delta_2)_A \coloneqq \langle\!\langle \eta_A(\Delta_1), \eta_A(\Delta_2) \rangle\!\rangle = \langle A^{-1}\Delta_1, A^{-1}\Delta_2 \rangle \qquad \Delta_1, \Delta_2 \in T_A \operatorname{Sym}^+(n).$$

Applying now the formula for the geodesics, we have that for an element $\Delta \in T_A \operatorname{Sym}^+(n) \cong \operatorname{Sym}(n)$

$$\begin{aligned} \exp_A(\Delta) &= (\mu^{\mathrm{I}_n} \circ \pi)(A^{1/2} \operatorname{expm}(\tfrac{1}{2} A^{-1/2} \Delta A^{-1/2})) \\ &= A^{1/2} \operatorname{expm}(A^{-1/2} \Delta A^{-1/2}) A^{1/2} \\ &= A \operatorname{expm}(A^{-1}\Delta), \end{aligned}$$

where in the last equality we have used that the exponential and the conjugation by a matrix commute.

We summarise these computations in the following proposition.

Proposition 3.5.6 (Exponential map on $\operatorname{Sym}^+(n)$). *Let $A \in \operatorname{Sym}^+(n)$ and $\Delta \in T_A \operatorname{Sym}^+(n) \cong \operatorname{Sym}(n)$ be a point and a vector respectively. The Riemannian exponential map with respect to the metric $g(\Delta_1, \Delta_2)_A = \langle A^{-1}\Delta_1, A^{-1}\Delta_2 \rangle$ is given by*

$$\exp_A(\Delta) = A^{1/2} \operatorname{expm}(A^{-1/2} \Delta A^{-1/2}) A^{1/2} = A \operatorname{expm}(A^{-1}\Delta).$$

3.5.4 Fixed-rank matrices

The manifold $\mathbb{R}_r^{n \times k}$ of $n \times k$ matrices of fixed rank r is a manifold that is often considered in the context of low-rank optimisation. This manifold is a homogeneous manifold as it has a transitive left action from $\operatorname{GL}(n) \times \operatorname{GL}(k)$ given by

$$\begin{aligned} \mu \colon (\operatorname{GL}(n) \times \operatorname{GL}(k)) \times \mathbb{R}_r^{n \times k} &\to \mathbb{R}_r^{n \times k} \\ ((A, B), X) &\mapsto A X B^{-1}. \end{aligned}$$

It is clear that the action is well-defined by looking at X as a linear map from $\mathbb{R}^k$ to $\mathbb{R}^n$ looking at A, B as linear isomorphisms on these spaces. It is transitive, as, after an SVD, any matrix $X \in \mathbb{R}_r^{n \times k}$ may be taken to the matrix $x_0 = \left(\begin{smallmatrix} \mathrm{I}_r & 0_{r,k-r} \\ 0_{n-r,r} & 0_{n-r,k-r} \end{smallmatrix}\right)$. The isotropy group of this action is then given by

$$H = \left\{ \begin{pmatrix} C & A_1 \\ 0_{n-r,r} & A_2 \end{pmatrix}, \begin{pmatrix} C & 0_{r,k-r} \\ B_1 & B_2 \end{pmatrix} \middle| \begin{array}{lll} C \in \operatorname{GL}(r), & A_1 \in \mathbb{R}^{r \times (n-r)}, & B_1 \in \mathbb{R}^{(k-r) \times r} \\ A_2 \in \operatorname{GL}(n-r), & B_2 \in \operatorname{GL}(k-r) \end{array} \right\} \subseteq \operatorname{GL}(n) \times \operatorname{GL}(k)$$

so we have that $\mathbb{R}_r^{n \times k} \cong (\operatorname{GL}(n) \times \operatorname{GL}(k))/H$.

As opposed to the previous examples, this manifold is much more difficult to work with, as it does not admit the structure of a Riemannian homogeneous space. In particular, it is not reductive outside of the case $\mathbb{R}_n^{n \times n} \cong \operatorname{GL}(n)$ (Munthe-Kaas and Verdier 2016, Prop. 5.7). This is one of the reasons why doing low-rank optimisation is particularly challenging, as we do not have at our disposal all the tools developed above to compute the geodesics.

Chapter 4

Fibred Manifolds in First Order Optimisation

In this chapter we show how the theory of Riemannian submersions and the ideas developed in Chapter 3 can be used to prove convergence results for optimisation problems on some manifolds in terms of problems on simpler manifolds by looking at submersions and Riemannian submersions. We also present a general treatment of the **dynamic trivialisation framework**. This framework was originally published in the paper (Lezcano-Casado 2019) at NeurIPS 2019. The proofs of convergence for this framework and an in-depth literature review are postponed to Chapter 5.

Outline of the chapter. In Section 4.1, we set up the notation for the rest of the chapter, and we include a short historical introduction to the field. In Section 4.2, we give some reasons for why submersions are of interest in the context of first-order optimisation, and we discuss the relation between Riemannian submersions and retractions. In Section 4.3 we prove all the necessary tools to give convergence rates on a base manifolds in terms of convergence rates on a total space, when these two are connected by a Riemannian submersion. In Section 4.4, we take a closer look at the **static trivialisation framework** when using the Riemannian exponential map to pullback the problem to a Euclidean space. In Section 4.5, we introduce the **dynamic trivialisation framework**, and outline the practical advantages that it provides. These advantages will be put in practice all throughout Chapter 6 during the design of the library for optimisation on manifolds GeoTorch presented in that chapter.

4.1 Optimisation on Manifolds

Consider the problem

$$\min_{x \in \mathsf{X}} f(x)$$

where $\mathsf{X} \subseteq \mathbb{R}^n$. In this section we will consider the setting in which X has a differentiable manifold structure, and we will denote this set by M. More generally, we will consider the intrinsic view of the problem, where M is just an abstract differentiable manifold, writing the problem as

$$\min_{x \in M} f(x). \tag{4.1}$$

In order to be able to perform first order optimisation, we will need some extra structure on M such as a Riemannian metric or an affine connection. We will consider this extra structure as we need it through the chapter. Even more, we will see that the point that makes manifold optimisation simpler than general constrained optimisation is the existence of a group of symmetries. In the cases when this group of symmetries is large enough, we will be able to make use of the tools presented in the last chapter. This approach will allow us to prove convergence results for this particularly well-behaved set of constrained optimisation problems.

The idea of taking advantage of the Riemannian structure of an embedded manifold, seen as a curved space locally approximated by their tangent spaces was first introduced in (Luenberger 1972). Luenberger sees a manifold as a set of constraints $M = \{x \in \mathbb{R}^n \,|\, h(x) = 0\}$ for a map $h \colon \mathbb{R}^n \to \mathbb{R}^m$ for which zero is a regular value. He then presents two extrinsic projected gradient descent algorithms: Orthogonal projection onto the manifold, and projecting following the perpendicular to the tangent space, seeing the tangent space as an affine subspace of the ambient space. He also considers the intrinsic algorithm of gradient descent along geodesics. About these different perspectives, Luenberger notes:

> Developing a convergence analysis for the gradient projection method leads quickly to horrendous approximations and almost endless assumptions. Such analyses hardly compare with the relative neatness and elegance of the corresponding analysis of the unconstrained case.

but he also adds:

> It should not be inferred that this paper suggests that when implementing such a scheme one should try to move along a geodesic. The geodesic method is proposed for theoretical purposes only. It should be noted, however, that all the methods for selecting the descent curve discussed above are asymptotically equivalent in the sense that the curves on M, determined by gradient projection or the geodesic leaving in the direction of the projected gradient, are all tangent at the starting point. This means that as the step size goes to zero the methods all move approximately along the same curve. This in turn implies that the algorithms all possess the same asymptotic convergence properties. In all cases the simplest procedure should be used. For computation the simplest is the second of the projection methods; for analysis it is certainly the geodesic method.

In other words, following a geodesic in the direction of steepest descent or following a curve that is tangent to the geodesic is asymptotically equivalent. This idea has driven the research in the field for the last 50 years under the name of **retractions**.

Definition 4.1.1. A map $r \colon TM \to M$ is a **retraction** if for every $p \in M$, the map $r_p \colon T_pM \to M$ satisfies

$$r_p(0) = p \qquad \text{and} \qquad (\mathrm{d}r_p)_0 = \mathrm{Id}.$$

By construction, a curve defined by a retraction $r_p(tv)$ for $t \in [0, 1]$ is tangent at p to the geodesic with initial conditions $(p, v) \in TM$. In particular, the Riemannian exponential map is a retraction, and as

such, retractions can be thought of as a first order approximation to the Riemannian exponential map on (M, g) for any metric g.

Note that in the definition of retraction there is no condition on M being an embedded manifold of a Euclidean space nor there is any mention to a metric structure on M. For this reason, to be able to talk about notions such as distances or directions of steepest descent, it is convenient to consider a Riemannian metric on M so that (M, g) is a Riemannian manifold. Armed with these two tools, it is possible to define what has been the most popular first order algorithm for optimisation on manifolds: **Gradient descent along a retraction**

$$x_{t+1} = r_{x_t}(-\eta \nabla f(x_t)). \tag{4.2}$$

This algorithm has been instantiated for most manifolds of interest in the area of optimisation by finding convenient retractions for each of them. For an introduction to these methods and their applications, we refer the reader to the book (Absil, Mahony, and Sepulchre 2009).

Even though we needed a Riemannian structure on M to be able to define the gradient in (4.2), the choice of a retraction is independent of this Riemannian metric. This is the main reason underlying Luenberger's remark that proving convergence for these methods is far from clean. As Luenberger mentions, things considerably simplify when r is chosen to be the Riemannian exponential map. In this case, it is possible to give some meaningful theorems, as the exponential map may be studied through the properties of the metric tensor. We will explore these ideas further in Chapter 5.

Concurrently with Luenberger's work, Brockett brought techniques from differential geometry to numerical analysis to study differential equations on Lie groups, with applications to control theory (Brockett 1972). His approach was much more abstract, considering intrinsic properties of the manifolds, rather than looking at them as manifolds embedded in $\mathbb{R}^n$. It was his PhD student Smith who, in his thesis, applied these techniques for the first time in the context of optimisation on manifolds, generalising for the first time conjugate gradient descent and the Newton's method to abstract manifolds (Smith 1993). These ideas were written aimed at a numerical analysis reader in the influential paper (Edelman, Arias, and Smith 1998).

We follow this latter approach, concentrating mostly on the global perspective as done in Chapter 3, given that we believe that it strongly emphasises the underlying geometry of the problem.

We postpone to Section 5.1.2 a full literature review, as presenting a literature review before presenting the algorithmic framework may make it unnecessarily difficult to follow.

4.2 Fibred Manifolds and First Order Structure

Consider a map $\varphi\colon \overline{M} \to M$ where $\overline{M}$ is another differentiable manifold. We may transform problem (4.1) into

$$\min_{p \in \overline{M}} f(\varphi(p)). \tag{4.3}$$

We define this to be a **pullback problem** associated to the original problem (4.1). We will also refer to this pullback problem as a **static trivialisation** of the initial problem.[1]

[1]The "trivialisation" part of the name comes from the fact that, in the case when $\dim \overline{M} = \dim M$, which will be quite important later on when $\overline{M} = T_pM$, if φ does not have critical points on a neighbourhood, then it defines a local trivialisation

We state now the two main examples that first drew our attention to this family of problems.

Example 4.2.1 (Homogeneous spaces). Consider a homogeneous space $\pi\colon G \to G/H$. The function π allows pulling back an optimisation problem from a space which may have a potentially difficult topology to a Lie group, where we may exploit its group structure. This was a recurrent idea in Chapter 3 and in this chapter we will see how we may apply this to optimisation on manifolds.

Example 4.2.2 (Tangent space). Consider $\overline{M} = T_pM$ for a fixed $p \in M$. We will see that, if we consider a complete Riemannian manifold (M, g) with non-positive sectional curvature, we may choose $\varphi = \exp_p$ so that the map $\exp_p\colon T_pM \to M$ is a submersion. In the general setting, even though it is not a global submersion, we can still control the problems that may arise for the manifolds of interest using standard tools from geometry, as we will see in Section 4.4. The failure of $\exp_p$ at being a global submersion will naturally lead to the concept of **dynamic trivialisations** studied in Section 4.5.

If we pullback the problem to a Euclidean space, there are some practical advantages. The main advantage is that on a Euclidean space we have many different optimisation techniques which exploit its vector space structure: Momentum methods, adaptive methods, quasi-Newton methods, ... We will see that this gives a great advantage over traditional optimisation methods when dealing with non-convex optimisation problems in the deep learning context in Section 6.4. We will enumerate a number of other different advantages that this approach presents at the end of Section 4.5.

We now turn our attention to the study the general properties of static trivialisations in terms of those of the original problem.

If we are going to compare the pullback problem with the original problem, we should at least ask for φ to be **surjective**. If φ is surjective, then both minimisation problems are equivalent

$$\min_{p\in\overline{M}} f(\varphi(p)) = \min_{x\in M} f(x).$$

In this setting, if the original problem has a unique minimiser $x^* \in M$ then, any minimiser of the pullback problem $p^* \in M$ will be related to x^* through φ as $\varphi(p^*) = x^*$. In general, if φ is surjective, φ will map global minimisers of the pullback problem to global minimisers of the original problem. In other words, surjective maps do not modify the 0th order structure of the problem.

Given that in our study we will be looking at first order optimisation methods, it would be reasonable to ask φ to also preserve the first order structure of the f in some sense. For example, if we want to parametrise $\mathbb{R}_{>0}$ in terms of $\mathbb{R}$, we could choose $\varphi(t) = \log(1 + \exp(t))$ as the parametrisation. On the other hand, we could also consider a function like the one seen in Figure 4.1. Both of them are smooth surjective mappings from $\mathbb{R}$ into $\mathbb{R}_{>0}$, but it should be clear that it may not be the best idea to use the latter one, as it will create new local minima or saddle points in pullback problem that did not exist in the original one.

More abstractly, by the chain rule we have:

$$\mathrm{d}(f \circ \varphi) = \mathrm{d}f \circ \mathrm{d}\varphi.$$

around that point, which we may use to simplify gradient computations and use optimisers from $\mathbb{R}^n$. These ideas were studied in the context of discretisation of differential equations on Lie groups, where one has access to global trivialisations of the tangent bundle (Magnus 1954; Iserles and Nørsett 1999; Iserles, Munthe-Kaas, et al. 2000). The "static" part comes opposed to the **dynamic trivialisations**, which will be introduced in Section 4.5.

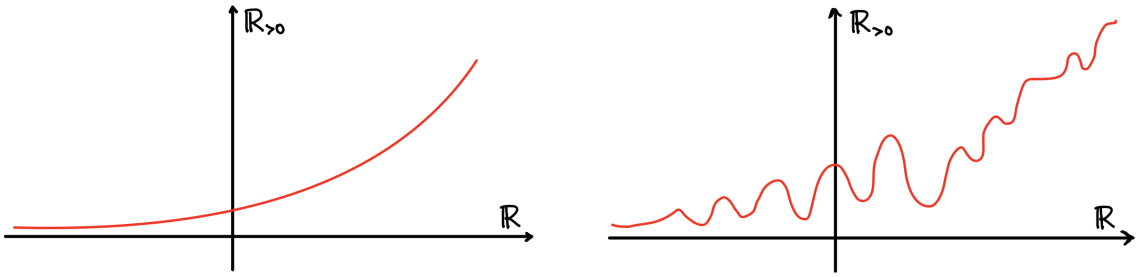

Figure 4.1: Two surjective maps from $\mathbb{R}$ to $\mathbb{R}_{>0}$ with vastly different properties when used in optimisation problems.

In local coordinates around a point, this is just a vector-matrix multiplication. Now, if we want $\mathrm{d}(f \circ \varphi)$ to have critical points if and only if $\mathrm{d}f$ has critical points, this may only happen if $\mathrm{d}\varphi$ has a right-inverse at every point, that is, if $\mathrm{d}\varphi$ is surjective at every point. In other words, since we wanted φ to be also surjective, we have that the maps that do not create new critical points are exactly **submersions**.

Proposition 4.2.3 (Submersions do not create spurious critical points). *Let $\varphi\colon \overline{M} \to M$ be a surjective map between manifolds. The following are equivalent:*

1. *The map φ is a submersion.*
2. *For every function f on M and every point $p \in \overline{M}$, p is a critical point of $f \circ \varphi$ if and only if $\varphi(p)$ is a critical point of f.*

Note that Proposition 4.2.3 does not mean that the critical points of f and $f \circ \varphi$ are in bijection through φ, but rather that for every point critical point $x \in M$ we get a fibre of critical points $\overline{M}_x \coloneqq \varphi^{-1}(x)$. In other words, the critical points of $f \circ \varphi$ are somewhat "concentrated in the same way as those from f", so that φ does not create new spurious critical points.

Another way of looking at this idea is through the lens of local sections. If φ is a submersion, for any two points such that $\varphi(p) = x$, there exists a map $\sigma\colon \varphi(U) \to U$ on a neighbourhood U of p such that $\varphi \circ \sigma = \mathrm{Id}_{\varphi(U)}$ (Proposition 3.1.3). In particular, if x is an isolated critical point of f on a neighbourhood W, then p will be an isolated critical point of $f \circ \varphi$ on the immersed manifold $\Sigma = U \cap \sigma(W) \subseteq \overline{M}$. Note that this submanifold Σ need not be open in $\overline{M}$, so it will not be a neighbourhood of p whenever $\dim(\overline{M}) > \dim(M)$. In this case, the fibre will not be discrete—will have dimension $\dim(\overline{M}) - \dim(M)$—and the point p will not be an isolated critical point of $f \circ \varphi$ on $\overline{M}$.

Let us look at all these ideas in a simple example to help visualise them.

Example 4.2.4 (Submersion onto the circle). Let $\pi\colon \mathbb{R}^2 \backslash \{0\} \to \mathbb{S}^1$ be the orthogonal projection onto the unit circle in $\mathbb{R}^2$. The differential of this map at a point projects a vector onto its tangent component, so π is a submersion. We can see several local sections of this submersion at a point $p \in \mathbb{R}^2 \backslash \{0\}$ in Figure 4.2. This fibred manifold is particularly simple, as the total space splits as a product of the base space and the fibre $\mathbb{R}^2 \backslash \{0\} \cong \mathbb{S}^1 \times \mathbb{R}_{>0}$. This means that this is a trivial fibre bundle, so it also has global sections, as seen in Figure 4.2.

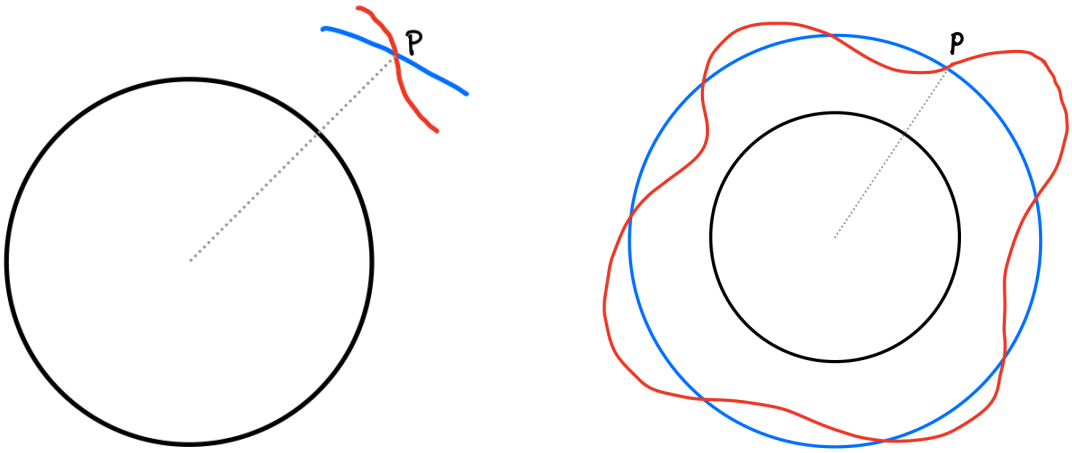

Figure 4.2: Examples of local and global sections of the fibre bundle $\pi\colon \mathbb{R}^2\backslash\{0\} \to \mathbb{S}^1$.

Consider a function $f\colon \mathbb{S}^1 \to \mathbb{R}$ with an isolated critical point $x \in \mathbb{S}^1$. We may pullback this function to $\mathbb{R}^2\backslash\{0\}$ using π. In this case, a critical point $x \in \mathbb{S}^1$ is transformed into a whole fibre of critical points of $f \circ \pi$. On the other hand, none of the fibres next to that one have any critical points.

Another slightly different submersion onto the circle that is sometimes of help is that given from its universal cover. If we have a function f on the circle, we may pull it back to a 1-periodic function on the real line via the map $\rho\colon \mathbb{R} \to \mathbb{S}^1 \subseteq \mathbb{C}$ given by $\rho(t) = e^{2\pi i t}$. This submersion is different in nature to the previous one given that, since $\dim \mathbb{R} = \dim \mathbb{S}^1$, it has a discrete fibre. As such, this submersion maps isolated critical points to isolated critical points.

This second example hints at how submersions help to define functions on manifolds via functions on other more amenable spaces. We will exploit this idea in the sequel when working with homogeneous spaces $\pi\colon G \to G/H$.

Remark 4.2.5 (Higher order generalisation). We have motivated the use of submersions in the context of first order optimisation via the existence of 1st order right-inverses. It is worth noting that the idea of a function being surjective can be phrased in categorical terms as the function having a 0th order right-inverse, that is, an inverse in the category of sets. This should be not surprising, as the difference between f and the pullback $f \circ \varphi$ is just a right composition.

This hints at how to extend these ideas and the ideas presented in the sequel to higher order methods. On the other hand, this extension gets quite technical very quickly as, to be able to talk about higher-order derivatives either one has work with jet bundles, or one has to fix connections—or Riemannian metrics—on $\overline{M}$ and M to be able to talk about $\nabla \mathrm{d}\varphi$. It is for this reason that we leave the higher order treatment for future research.

There are also examples of very simple submersions that are widely used in the machine learning and deep learning community.

Example 4.2.6 (Submersions in machine learning). Let $M \cong \mathbb{R}^n$ and consider component-wise functions like the hyperbolic tangent $\tanh\colon \mathbb{R}^n \to (-1,1)^n$ or the sigmoid function $\sigma\colon \mathbb{R}^n \to (0,1)^n$. These

examples are not only submersions but also diffeomorphisms. We already saw their use in the LSTM model in Section 2.3.3.

For now, we have looked at the use of submersions just through the lens of differential geometry. We still need to talk about metric structures to be able to make sense of notions relevant to convergence and gradients. Before doing so, let us fix some notation and recall some standard constructions.

Assume that we have two metrics $g, \overline{g}$ on M and $\overline{M}$ respectively. For these metrics, denote by $\alpha\colon TM \to T^*M$ (resp. β) the isomorphism between the tangent and cotangent bundle given by the metric, $\alpha(X) \coloneqq g(X, -)$. For a given map $\psi\colon \overline{M} \to M$ we denote its fibre-wise dual as

$$\begin{aligned} \mathrm{d}\psi'\colon T^*M &\to T^*\overline{M} \\ \eta &\mapsto \eta \circ \mathrm{d}\psi \end{aligned}$$

and its fibre-wise adjoint with respect to g and $\overline{g}$ as $\mathrm{d}\psi^*\colon TM \to T\overline{M}$, which is defined for a vector field X on M and a vector field Y along ψ as

$$\overline{g}(X, (\mathrm{d}\psi)^*(Y)) \coloneqq g(\mathrm{d}\psi(X), Y).$$

Proposition 4.2.7. *Let $\psi\colon \overline{M} \to M$ be a smooth map—not necessarily a submersion—between Riemannian manifolds with metrics $\overline{g}$ and g respectively. The following relation holds*

$$\beta \circ \mathrm{d}\psi^* = \mathrm{d}\psi' \circ \alpha.$$

Proof. Let Y be a vector field on $\overline{M}$ and X be a vector field along ψ,

$$(\mathrm{d}\psi' \circ \alpha)(X)(Y) = g(X, \mathrm{d}\psi(Y)) = \overline{g}(\mathrm{d}\psi^*(X), Y) = (\beta \circ \mathrm{d}\psi^*)(X)(Y). \qquad \square$$

Using this proposition, we can compute the gradient with respect to the new parametrisation. This is sometimes referred to as **the adjoint method** in numerical analysis.

Corollary 4.2.8 (The adjoint method). *Let $\psi\colon \overline{M} \to M$ be a smooth map between Riemannian manifolds and f be a function on M. Then,*

$$\nabla(f \circ \psi) = \mathrm{d}\psi^*(\nabla f). \tag{4.4}$$

Proof. This is direct using the previous proposition

$$\nabla(f \circ \psi) \coloneqq \beta^{-1}(\mathrm{d}(f \circ \psi)) = (\beta^{-1} \circ \mathrm{d}\psi')(\mathrm{d}f) = \mathrm{d}\psi^*(\nabla f). \qquad \square$$

If $\varphi\colon \overline{M} \to M$ is a Riemannian submersion, since $\mathrm{d}\varphi|_{\mathcal{H}}$ is a linear isometry and $\mathrm{d}\varphi|_{V\overline{M}} = 0$, we have that

$$\mathrm{d}\varphi \circ \mathrm{d}\varphi^* = \mathrm{Id}_{TM}.$$

In other words, the map $\mathrm{d}\varphi^*$ takes values in the horizontal bundle $\mathcal{H}$ (Definition 3.1.6) and is a linear inverse of $\mathrm{d}\varphi|_{\mathcal{H}}$—the non-zero part of $\mathrm{d}\varphi$—and a linear right-inverse of $\mathrm{d}\varphi$. This formula together with Proposition 4.2.7 is particularly useful to check when a map φ is a Riemannian surjection on low dimensional manifolds if we have access to a local representation of g and $\overline{g}$, as we do not need to compute the horizontal space associated to $\overline{g}$.

It is also worth noting that, while the right-hand side of (4.4) seems to depend on the choice of metric on M, it does not. Even then, when computing gradients recursively in autodiff frameworks, it is convenient to fix a metric on every intermediate manifold and compute the adjoints with respect to these fixed metrics, so that we can compose the gradient given by some layer with the adjoint with respect to these metrics of the next layer. Of course, in most situations this metric is just the canonical metric on $\mathbb{R}^n$, so computing the adjoint simply accounts for transposing the Jacobian.

Let us now see how submersions provide a natural way to define a retraction on M in terms of one on $\overline{M}$. Assume that we have a retraction $\overline{r}$ on $\overline{M}$, that is, a method to perform optimisation on $\overline{M}$. This induces discrete dynamics on M as the pushforward of the dynamics on $\overline{M}$ for a step-size $\eta > 0$

$$p_{t+1} = \overline{r}_{p_t}(-\eta \nabla f(p_t))$$
$$x_{t+1} = \psi(p_{t+1})$$

This allows us to define the step map on M.

Definition 4.2.9. Let $\psi\colon \overline{M} \to M$ be a smooth map between Riemannian manifolds and let $\overline{r}$ be a retraction on $\overline{M}$. We define the **step map of $\overline{r}$ along ψ at** $p \in \overline{M}$ as

$$\Psi_p^{\overline{r}} \coloneqq \psi \circ \overline{r}_p \circ (\mathrm{d}\psi)_p^*\colon T_{\psi(p)}M \to M.$$

In this way, we can write the dynamics on M as

$$x_{t+1} = \Psi_{p_t}^{\overline{r}}(-\eta \nabla f(x_t))$$

where p_t is defined as before. A question that arises now is whether the map $\Psi_p^{\overline{r}}\colon TM \to M$ is a retraction for every $p \in \overline{M}$.

Proposition 4.2.10. *Let $\psi\colon \overline{M} \to M$ be a smooth map between Riemannian manifolds and let $\overline{r}$ be a retraction on $\overline{M}$. Then $\Psi_p^{\overline{r}}$ is a retraction for every p if and only if ψ is a—not necessarily surjective—Riemannian submersion.*

Proof. It is clear that $\Psi_p^{\overline{r}}(x, 0) = x$. For the second condition, differentiating $\Psi_p^{\overline{r}}$, we have that the map is a retraction if and only if

$$\mathrm{d}\psi_p \circ \mathrm{d}\psi_p^* = \mathrm{Id}_{T_{\psi(p)}M}$$

or equivalently, whenever ψ is a not necessarily surjective submersion. □

In the simplest case, when the dimension of $\overline{M}$ and M agree, this proposition says that the gradient is preserved just by local isometries, which is exactly what one would expect.

In the general case, we have seen in Proposition 3.1.10 that, if we have a submersion $\varphi\colon \overline{M} \to M$ and a metric on M, we may lift this metric to a metric on $\overline{M}$ that makes $\varphi\colon \overline{M} \to M$ into a Riemannian submersion after choosing an Ehresmann connection on $\overline{M}$ and a metric tensor on $V\overline{M}$. As such, Proposition 4.2.10 shows that this process gives dynamics induced by a gradient descent performing optimisation on $\overline{M}$ rather than on M after lifting the gradient on every step to a horizontal gradient on $\overline{M}$ via $(\mathrm{d}\varphi)^*$.

On the other hand, there are metrics on $\overline{M}$ that do not descend into a metric on M—those metrics that are not constant on the fibres. This shows that, through this method, we can get new dynamics that cannot be achieved just by performing gradient descent with a retraction on M with respect to any metric.

There is a third interesting setting. If φ is a diffeomorphism, and we have two distinguished metrics $\overline{g}, g$, but these metrics do not make φ into a Riemannian submersion, optimising $f \circ \varphi$ accounts for optimising f with respect to the pushforward metric $\varphi_*(\overline{g})$ on M. Therefore, precomposing by φ accounts for a change of metric on M. More generally, we may find this scenario when we have a metric on $\overline{M}$ that does descend into a metric on M.

To summarise, we have three scenarios of particular interest:

1. $\pi\colon \overline{M} \to M$ is a Riemannian submersion
2. It is not a Riemannian submersion, but the metric on $\overline{M}$ descends into a metric on M
3. The metric on $\overline{M}$ does not descend into a metric on M.

We will study the first case in Section 4.3 and the second one in Section 4.4.

The third case is particularly interesting in the setting of linear networks, as it has been shown in (Arora, N. Cohen, and Hazan 2018) that overparametrisation in the linear setting yields dynamics that fall in this setting—dynamics that cannot be achieved by Riemannian gradient descent—and furthermore, they achieve acceleration-like convergence. The follow-up work (Bah et al. 2019) looks at these results from the lens of Riemannian geometry. This allows the authors to prove stronger results than those in the original paper. Even though we think that the ideas presented in this thesis would help to draw a more geometric picture of the behaviour described in these papers, we will not pursue this research direction in this thesis.

Before delving into the first two points, we continue the simple example of the circle to showcase some of these scenarios.

Example 4.2.11 (Projection onto the circle). We continue with the example of the orthogonal projection $\pi\colon \mathbb{R}^2\backslash\{0\} \to \mathbb{S}^1$. Both the total space and the base space involved in this submersion have a distinguished metric, *i.e.*, that induced by their identification as subsets of $\mathbb{R}^2$. These metrics do not turn π into a Riemannian submersion. This is clear, since Riemannian submersions map horizontal geodesics to geodesics, and the projection of geodesics of $\mathbb{R}^2\backslash\{0\}$ with horizontal initial conditions—straight lines orthogonal to the fibre—just cover half of $\mathbb{S}^1$, while the round metric on $\mathbb{S}^1$ is complete.

The standard metric on $\mathbb{R}^2\backslash\{0\}$ in polar coordinates is represented as $\mathrm{d}r^2 + r^2\mathrm{d}\theta^2$, while the differential of π is given by $\mathrm{d}\pi = \partial_t \otimes \mathrm{d}\theta$, where ∂_t is the line element on $\mathbb{S}^1$. The pullback of the canonical metric on the circle $\mathrm{d}t^2$ onto the horizontal bundle spanned by ∂_θ that makes $\mathrm{d}\pi|_{\mathcal{H}}$ into a linear isometry is given by $\pi^*(\mathrm{d}t^2) = \mathrm{d}\theta^2$—rather than $r^2\mathrm{d}\theta^2$, as the metric has. As described in Section 3.1, we find that a family of metrics that make π into a Riemannian submersion are those of the form $g(r,\theta)^2\mathrm{d}r^2 + \mathrm{d}\theta^2$ for a strictly positive function. For example, for $g \equiv 1$ we get a cylinder.

The horizontal geodesics for any metric of this form starting at (r, θ) are just circles of radius r and therefore, they project onto geodesics on $\mathbb{S}^1$, as expected. Note that this construction of a family of metrics that make a submersion into a Riemannian submersion is exactly that described in Proposition 3.1.10.

The standard metric on $\mathbb{R}^2 \backslash \{0\}$ does not descend into a metric on $\mathbb{S}^1$ and as such, the dynamics given by optimising $f \circ \pi$ with respect to this metric would not be given by any retraction on the circle.

This example is not too interesting, as the metric that makes the projection into a Riemannian submersion is the one that "copies" the metric onto each circle of the total space. This is due to the fact that the bundle is trivial. When the bundle is not trivial, this construction gives much more interesting examples, as we will see in the next section.

4.3 Riemannian Submersions

Riemannian submersions are the simplest of all the previous scenarios. They are a natural generalisation of isometries, and thus, they are as well-behaved as one could expect. Riemannian submersions allow us to prove a fairly general convergence theorem, which can be interpreted as saying that if we know how to optimise in the total space $(\overline{M}, \overline{g})$ and we have a Riemannian submersion $\varphi \colon \overline{M} \to M$, it is sensible to consider the optimisation algorithm along the step-map, as described in the previous section.

Theorem 4.3.1. *Let $\varphi \colon \overline{M} \to M$ be a Riemannian submersion, $\overline{r}$ a retraction on $\overline{M}$ and f a function on M. If for an initial point $p_0 \in \overline{M}$ and a step $\eta > 0$ the sequence*

$$p_{t+1} = \overline{r}_{p_t}(-\eta \nabla (f \circ \varphi)(p_t))$$

converges to a critical point p^* of $f \circ \varphi$ on $\overline{M}$ as

$$d_{\overline{M}}(p_t, p^*) \leq \frac{C}{t^r}$$

for some constants $C, r > 0$, the sequence of points

$$x_t = \varphi(p_t) \tag{4.5}$$

converges to the critical point $x^* \coloneqq \varphi(p^*)$ of f on M at the same rates:

$$d_M(x_t, x^*) \leq \frac{C}{t^r}.$$

Proof. Since p^* is a critical point of $f \circ \varphi$ and φ is a submersion, x^* is a critical point of f (Proposition 4.2.3). The result then follows from the fact that a Riemannian submersion is distance decreasing (Proposition 3.1.11). □

What this theorem says is that if we have convergence rates on $(\overline{M}, g)$ for a family of functions $\overline{\mathcal{C}}_{(\overline{M}, \overline{g})}$, and the pullback by the Riemannian submersion φ respects that family of functions—if $f \in \mathcal{C}_{(M,g)}$ then $f \circ \varphi \in \overline{\mathcal{C}}_{(\overline{M}, \overline{g})}$—then, we have the same convergence rates for $\mathcal{C}_{(M,g)}$ by applying the algorithm to $f \circ \varphi$ and using φ to project the iterates. The families of functions do not even need to be "the same" on both manifolds.

A tighter but more abstract way of saying this is that if we know how to prove convergence of an algorithm for the pullback of a family of functions $\varphi^*(\mathcal{C}_{(M,g)})$, we may pushforward the algorithm to solve problems in $\mathcal{C}_{(M,g)}$.

We will now see how to implement this idea for the class of functions of α-bounded Hessian functions on a manifold (Definition 2.2.4) in terms of functions that are L-Lipschitz and of α-bounded Hessian. We therefore set

$$\begin{aligned}\mathcal{C}^{L,\alpha}_{(M,g)} &= \{f \in C^2(M) \mid \|\nabla f\| \leq L,\ \|\mathrm{Hess}\, f\| \leq \alpha\}\\ \overline{\mathcal{C}}^{\alpha'}_{(\overline{M},\overline{g})} &= \{f \in C^2(\overline{M}) \mid \|\mathrm{Hess}\, f\| \leq \alpha'\}\end{aligned}$$

where the bounds on the gradient and Hessian are to be regarded uniformly point-wise.

In what follows, we will show that when $\varphi\colon \overline{M} \to M$ is a Riemannian submersion with some further properties, we actually have the equality

$$\varphi^*(\mathcal{C}^{L,\alpha}_{(M,g)}) = \overline{\mathcal{C}}^{\alpha'}_{(\overline{M},\overline{g})}$$

for a concrete value of $\alpha' > 0$ that just depends on L, α, and φ. We will explicitly bound the φ-dependant quantity explicitly for some manifolds of interest.

We will devote all of Chapter 5 to the study of convergence rates for functions in the class $\overline{\mathcal{C}}^{\alpha'}_{(\overline{M},\overline{g})}$ in terms of the curvature of $(\overline{M},\overline{g})$. This, together with Theorem 4.3.1 yields convergence rates on some homogeneous spaces in terms of their curvature.

We start by recalling the usual bounds on the second order information on the pullback of a function. If a function f is L-Lipschitz of α-bounded Hessian and if we are have bounds for φ such as $\|\mathrm{d}\varphi\| \leq L_\varphi$ and $\|\nabla \mathrm{d}\varphi\| \leq \alpha_\varphi$, we have that

$$\|\nabla \mathrm{d}(f \circ \varphi)\| \leq \alpha L_\varphi^2 + L\alpha_\varphi.$$

As such, all we have to do to implement the plan above is to give first and second order bounds when φ is a Riemannian submersion.

We start by looking at a general submersion with totally geodesic fibres, like those presented throughout Chapter 3. Bounds similar to these can be found in (O'Neill 1966; Vilms 1970).

Proposition 4.3.2 (First and second order bounds of a submersion). *Let $\varphi\colon \overline{M} \to M$ be a Riemannian submersion with totally geodesic fibres with Levi-Civita connections $\overline{\nabla}, \nabla$ respectively. We have that*

$$\|\mathrm{d}\varphi\| \leq 1 \tag{4.6}$$

$$\|\nabla \mathrm{d}\varphi|_{T_p\overline{M}}\| = 2 \max_{\substack{X \in \mathcal{H}_p, V \in V_p\overline{M}\\ \|X\|=\|V\|=1}} \|(\overline{\nabla}_X \tilde{V})_{\mathcal{H}_p}\| \tag{4.7}$$

Here $Y_{\mathcal{H}_p}$ denotes the horizontal part of each vector $Y \in T_p\overline{M}$ and $\tilde{V}$ is any smooth local extension of $V \in V_pM$ to a vector field around p.

Proof. The first bound is direct as for $X \in T\overline{M}$

$$\|\mathrm{d}\varphi(X)\| = \|\mathrm{d}\varphi(X_{\mathcal{H}} + X_V)\| = \|\mathrm{d}\varphi(X_{\mathcal{H}})\| = \|X_{\mathcal{H}}\| \leq \|X\|.$$

To prove the second formula, recall that for vector fields X, Y on $\overline{M}$ the Hessian of φ is defined as

$$\nabla \mathrm{d}\varphi(X,Y) \coloneqq \nabla_{\mathrm{d}\varphi(X)}(\mathrm{d}\varphi(Y)) - \mathrm{d}\varphi(\overline{\nabla}_X Y). \tag{4.8}$$

We will split the computation into its horizontal and vertical components.

If $V \in V\overline{M}$ is a vertical vector, the first term of $\nabla \mathrm{d}\varphi(V,V)$ in (4.8) is zero since φ is constant on the fibres so that $\mathrm{d}\varphi(V) = 0$. On the other hand, the hypothesis that the fibres are totally geodesic means that $\overline{\nabla}_V V \in V\overline{M}$ (Proposition 3.1.15) so that $\nabla \mathrm{d}\varphi(V,V) = 0$.

If $X \in \mathcal{H}$ is a horizontal vector field, let $\overline{\gamma}$ be the horizontal geodesic with $\dot{\overline{\gamma}}(0) = X$, and let $\gamma = \varphi(\overline{\gamma})$ be projection onto a geodesic on M. Then, evaluating (4.8) along $\overline{\gamma}$ we have

$$\nabla \mathrm{d}\varphi(\dot{\overline{\gamma}}, \dot{\overline{\gamma}}) = \nabla_{\dot{\gamma}}\dot{\gamma} - \mathrm{d}\varphi(\overline{\nabla}_{\dot{\overline{\gamma}}}\dot{\overline{\gamma}}) = 0.$$

If V is a vertical vector field and $X \in \mathcal{H}_p$, then

$$\nabla \mathrm{d}\varphi(X,V) = -\mathrm{d}\varphi(\overline{\nabla}_X V) \tag{4.9}$$

and

$$\|\nabla \mathrm{d}\varphi(X,V)|_{T_p\overline{M}}\| = \|(\overline{\nabla}_X V)_{\mathcal{H}_p}\|.$$

Putting the horizontal and vertical case together, if $X \in T\overline{M}$ is an arbitrary vector, we can decompose it into its horizontal and vertical part so that $\|\nabla \mathrm{d}\varphi(X,X)\| = 2\|\nabla \mathrm{d}\varphi(X_{\mathcal{H}}, X_V)\|$. Using that $\Phi(X,Y) \coloneqq \nabla \mathrm{d}\varphi(X,Y)$ is a symmetric bilinear form, by polarisation, the triangle inequality and Cauchy–Schwarz

$$4\|\Phi(u,v)\| \leq \|\Phi|_{\mathrm{diag}}\|(\|u+v\|^2 + \|u-v\|^2) = 4\|\Phi|_{\mathrm{diag}}\|$$

where $\Phi|_{\mathrm{diag}}(u) \coloneqq \Phi(u,u)$ so that

$$\|\nabla \mathrm{d}\varphi|_{T_p\overline{M}}\| = \max_{\substack{u,v\in T_p\overline{M}\\ \|u\|=\|v\|=1}} \|(\nabla \mathrm{d}\varphi)_p(u,v)\| = \max_{\substack{u\in T_p\overline{M}\\ \|u\|=1}} \|(\nabla \mathrm{d}\varphi)_p(u,u)\| = 2 \max_{\substack{X\in\mathcal{H}_p, V\in V_p\overline{M}\\ \|X\|=\|V\|=1}} \|(\nabla \mathrm{d}\varphi)_p(X,V)\|. \qquad \square$$

For a normal Riemannian homogeneous space (Definition 3.4.11) these bounds can be simplified to an algebraic computation on the Lie algebra.

Proposition 4.3.3. *If (M,g) is a normal Riemannian homogeneous space with $\varphi\colon G \to M$ the canonical projection,*

$$\|\nabla \mathrm{d}\varphi\| = \max_{\substack{X\in\mathfrak{m}, Y\in\mathfrak{h}\\ \|X\|=\|Y\|=1}} \|[X,Y]\|$$

where $\mathfrak{h}$ and $\mathfrak{m}$ are as in Definition 3.3.14

Proof. We start by noting that, since (M,g) is a normal Riemannian homogeneous space, φ is a Riemannian submersion with totally geodesic fibres (Theorem 3.3.19), so that we are in the setting of Proposition 4.3.2.

In order to compute the norm of the Hessian of φ at an arbitrary point in G, given that the only non-zero part is that given by Equation (4.9), it is enough to compute the derivative of a vertical vector field with $V_p = V \in T_pM$ in the direction of a horizontal vector field. Furthermore, as we are working with a Riemannian homogeneous space, the norm of the Hessian of φ will not depend on the point. For this reason, it will be enough to compute it at the identity $e \in G$. But at the identity we can use that if V

is a vertical vector, $V \in \mathfrak{h}$ and V^{L} is a vertical vector field. The result follows from the formula for the connection on a normal Riemannian homogeneous space (Proposition 3.4.12) and the fact that due to the reductive condition $[\mathfrak{m}, \mathfrak{h}] \subseteq \mathfrak{m}$ we do not need to project onto $\mathfrak{m}$. □

Now, we may apply these results to the Stiefel manifold and the Grassmannian since, as described in Section 3.5, they are both normal Riemannian homogeneous spaces.

Proposition 4.3.4. *Let (M, g) be either the Stiefel manifold $\mathrm{St}(n, k)$ or the Grassmannian manifold $\mathrm{Gr}(n, k)$ as a bundle $\pi\colon \mathrm{SO}(n) \to M$ with the metric induced by the scalar product $\langle A, B\rangle = \frac{1}{2}\operatorname{tr}(A^{\intercal}B)$. Consider a function $f\colon M \to \mathbb{R}$ that is L-Lipschitz and of α-bounded Hessian. We then have that $f \circ \pi$ is of $(\sqrt{2}L + \alpha)$-bounded Hessian on $\mathrm{SO}(n)$.*

Proof. The chain rule and Cauchy–Schwarz give the bound

$$\|\mathrm{Hess}(f \circ \pi)\| \leq \|\mathrm{Hess}\, f\|\|\mathrm{d}\pi\|^2 + \|\mathrm{d}f\|\|\nabla \mathrm{d}\pi\| \leq \alpha + L \max_{\substack{X \in \mathfrak{m}, Y \in \mathfrak{h} \\ \|X\| = \|Y\| = 1}} \|[X, Y]\|$$

where we have used Propositions 4.3.2 and 4.3.3 and the bounds on f.

Since the total space $\mathrm{SO}(n)$ of the Stiefel manifold and the Grassmannian is a matrix manifold, the bracket on its Lie algebra is just the commutator of skew-symmetric matrices. As shown in Ge 2014, Lemma 2.5 for the Frobenius norm $\|A\|_F = \sqrt{\operatorname{tr}(A^{\intercal}A)}$ we have the bound

$$\|[A, B]\|_F \leq \|A\|_F \|B\|_F \qquad \forall A, B \in \mathfrak{so}(n).$$

Using that $\|A\| = \frac{1}{\sqrt{2}}\|A\|_F$ we get that

$$\|[A, B]\| \leq \sqrt{2}\|A\|\|B\| \qquad \forall A, B \in \mathfrak{so}(n)$$

and the result follows. □

Remark 4.3.5. The result in Ge 2014 also gives the equality cases for this bound of the norm. It is direct to see that no two matrices $[m, h]$ with $m \in \mathfrak{m}, h \in \mathfrak{h}$ achieve this bound. On the other hand, consider the case $\mathrm{Gr}(8, 4)$ and the matrices

$$A = \begin{pmatrix} 0 & -\lambda & 0 & 0 \\ \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda \\ 0 & 0 & \lambda & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 0 & 0 & \lambda \\ 0 & 0 & \lambda & 0 \\ 0 & -\lambda & 0 & 0 \\ -\lambda & 0 & 0 & 0 \end{pmatrix}$$

so that

$$m = \begin{pmatrix} 0_{4,4} & -A^{\intercal} \\ A & 0_{4,4} \end{pmatrix} \in \mathfrak{m} \qquad h = \begin{pmatrix} 0_{4,4} & 0_{4,4} \\ 0_{4,4} & B \end{pmatrix} \in \mathfrak{h}$$

then we see that

$$\|[m, h]\| = 2\lambda^2 \leq \|m\|\|h\| = 4\lambda^2$$

so the bound in the Hessian of π is pessimistic by at most a factor of 2. We have the same upper-bound for the constants for $\mathrm{St}(n, k)$ and $\mathrm{Gr}(n, k)$ for $n \geq 8$ and $k \geq 4$, via the totally geodesic immersions of $\mathrm{St}(8, 4)$ and $\mathrm{Gr}(8, 4)$ into the larger ones. Similar results can be obtained whenever $n < 8$ or $k < 4$.

4.4 Static Trivialisations

Let us now explore the static trivialisation framework from Equation (4.3) for the setting when $\dim(\overline{M}) = \dim(M)$. In this case, we will be forward a metric from $\overline{M}$ along φ into a new metric on M. Here we will present the framework in the fully general case. For a—perhaps illuminating—example in the setting of optimisation with orthogonal constraints see Section 6.1.2.

4.4.1 Diffeomorphisms

The simplest case comes when we have access to a diffeomorphism between the spaces $\varphi\colon \overline{M} \to M$. If we have a metric $\overline{g}$ on $\overline{M}$, we may then form the pushforward metric $\varphi_*(\overline{g})$ that makes φ into an isometry. If also have a retraction $\overline{r}$ on $\overline{M}$, and a metric g on M, we may define a step map on M as

$$\begin{aligned} \varphi_*(\overline{r})\colon TM &\to M \\ (x, v) &\mapsto (\varphi \circ \overline{r}_{\varphi^{-1}(x)} \circ (\mathrm{d}\varphi)^*_{\varphi^{-1}(x)})(v) \end{aligned}.$$

This construction is essentially the same as that described in Definition 4.2.9. The difference is that, in this case, given that we have a diffeomorphism, the fibre is trivial and the map does not depend on the lifted point. On the other hand, Proposition 4.2.10 still applies, and we have that this will be a retraction if and only if φ is a local isometry, that is, if $\varphi_*(\overline{g}) = g$.

The setting of having a diffeomorphism is not particularly interesting, as the existence of diffeomorphisms between manifolds is too strong a topological constraint. In the following section, we will see how the Riemannian exponential map is a submersion for some manifolds and, in general, it is a diffeomorphism in a *large enough* set. This allows us to safely pull back problems from a manifold with a potentially difficult topology to a Euclidean space.

4.4.2 The Riemannian exponential

Consider the pullback problem along the Riemannian exponential

$$\min_{v \in T_pM} f(\exp_p(v)).$$

We will assume throughout this section that we have a complete metric g on a connected manifold M. In this setting, $\exp_p$ is surjective and

$$\min_{x \in M} f(x) = \min_{v \in T_pM} f(\exp_p(v)).$$

Note that problem (4.3) does not come equipped a priori with a choice of metric. As such, we have the freedom to select one that makes our life easier. Complete metrics exist for any differentiable manifold (Nomizu and Ozeki 1961) but the geometry of a complete metric may not be simple in our manifold of choice. In particular, solving the second order non-linear differential equation to compute the Riemannian exponential map may be rather challenging if the metric does not have enough symmetries. Luckily, as we saw in Section 3.5 most of the manifolds that we are interested on in optimisation are rather amenable in this aspect, with the notable exception of fixed rank matrices (Section 3.5.4).

4.4.2.1 Hadamard manifolds

A reasonable question could be: When is the exponential map a diffeomorphism, so that we can pushforward the flat metric on the tangent space onto a metric onto the manifold? Hadamard manifolds have this desirable property.

Definition 4.4.1. A connected and complete Riemannian manifold (M, g) is **Hadamard** if its sectional curvature is everywhere non-positive.

The importance of Hadamard manifolds comes from the Cartan–Hadamard theorem.

Theorem 4.4.2 (Cartan–Hadamard theorem). *In a Hadamard manifold (M, g), for every $p \in M$, the map $\exp_p$ is a submersion. In particular, if the manifold is simply connected, $\exp_p$ is a diffeomorphism.*

We provide a proof of this classical theorem as a corollary of the expository results in Section 5.3. Examples of manifolds in this family that are of interest in optimisation are the hyperbolic space or the space of symmetric positive definite matrices with the left-invariant coming from its representation as a symmetric space $\mathrm{Sym}^+(n) \cong \mathrm{GL}^+(n)/\mathrm{SO}(n)$. After a choice of frame—a linear isomorphism between $\mathbb{R}^n$ and T_pM—we may effectively work on M as if it was $\mathbb{R}^n$ provided that we know how to compute the adjoint of the differential of the exponential map. This is something that we will explore in Section 4.4.2.3. For Hadamard manifolds, we may pullback the minimisation problem from a curved space into a Euclidean space on which we can use more sophisticated optimisation methods, such as momentum or adaptive methods.

Remark 4.4.3 (Numerical stability of the Riemannian exponential map on negatively curved manifolds). For manifolds that are strictly negatively curved, the exponential map grows exponentially fast in the sense described in Rauch's theorem (Theorem 5.3.16). For this reason, when using the exponential map to pullback a problem to a Euclidean space on a Hadamard manifold, one often encounters numerical stability issues which might have to be approached separately via a careful numerical implementation of the algorithms (see *e.g.*, Yu and Sa 2019).

Remark 4.4.4 (Other manifolds for which the Riemannian exponential is a diffeomorphism). Hadamard manifolds are a large family of manifolds on which the exponential map is a diffeomorphism but they are not the only ones. If the exponential map is a diffeomorphism at a point, the manifold may still have parts of positive curvature—*e.g.*, at the vertex of a paraboloid. In fact, there are examples of manifolds that are not Hadamard where the exponential map at any point is a covering map (Gulliver 1975).

4.4.2.2 The general case

For a general Riemannian manifold, the exponential may not be a covering map due to the existence of conjugate points. Although we will present an introduction to conjugate points in Section 5.3.1, we recall here one possible definition.

Definition 4.4.5. A point q **is conjugate to a point** p if the map $\exp_p$ is not regular at q, *i.e.*, its differential at a vector $v \in \exp_p^{-1}(\{q\}) \subseteq T_pM$ is not full rank.

Even though the Riemannian exponential is not in general a submersion, it is a local diffeomorphism around zero. The Gauss' lemma gives that the differential of the exponential map at zero is the identity. As such, by the inverse function theorem, $\exp_p$ is a diffeomorphism on a neighbourhood of $0 \in T_pM$. An obvious question now is: How large is this neighbourhood?

This is of course a well studied question in Riemannian geometry. We summarise here some relevant results from the literature.

Definition 4.4.6. Let (M, g) be a complete Riemannian manifold. We define the **segment domain**, and its interior as

$$\begin{aligned}\operatorname{seg}(p) &\coloneqq \{v \in T_pM \mid \exp_p(tv)\colon [0,1] \to M \text{ is length minimising}\}\\ \operatorname{seg}^0(p) &\coloneqq \{tv \in T_pM \mid t \in [0,1), v \in \operatorname{seg}(p)\}\end{aligned}$$

We also define the **tangent cut locus at** p and the **cut locus at** p as

$$\begin{aligned}\operatorname{Tcut}(p) &\coloneqq \operatorname{seg}(p) \setminus \operatorname{seg}^0(p)\\ \operatorname{cut}(p) &\coloneqq \exp_p(\operatorname{Tcut}(p))\end{aligned}$$

By Hopf–Rinow, $\exp_p(\operatorname{seg}(p)) = M$. More generally, the segment domain and the cut locus have the following properties.

Proposition 4.4.7 (Properties of the cut locus). *The segment domain has the following properties:*

1. $\operatorname{seg}^0(p)$ *is the interior of* $\operatorname{seg}(p)$. *In particular,* $\operatorname{seg}^0(p)$ *is open*
2. $\exp_p$ *is a diffeomorphism on* $\operatorname{seg}^0(p)$
3. *If* $v \in \operatorname{Tcut}(p)$, *then either* $\exp_p$ *fails to be injective at* $x = \exp_p(v)$, *or its differential fails to be surjective, or both*

Proof. See for example (Petersen 2016, Section 5.7.3). □

In plain words, $\operatorname{seg}^0(p)$ is a maximal star-shaped open neighbourhood of zero on T_pM on which the exponential map is a diffeomorphism.

Due to the third property of the cut locus, we see that points in the cut locus are those that are either joined by two geodesics of the same length starting at p (cut points) or conjugate points of p. We will look at how to bound the first occurrences of the latter and its applications to convex geometry on manifolds in Section 5.3.4.

We may now ask what is the topology of the cut locus. By Proposition 4.4.7, it is a closed set in M, but we do not know how big is it. The cut locus is a remarkably slippery object of study given that, in general, it is not differentiable. For example, for surfaces, the cut locus of a point has structure of a local tree, in the sense that every point in the cut locus has a neighbourhood W such that the connected component of $W \cap \operatorname{cut}(p)$ on which z lies is a tree (Myers 1935). Here by tree we mean that any two points can be joined by a unique injective curve. This tree, on the other hand, may be infinite of a fractal-like shape. This was later generalised in (Buchner 1977), where it was proved that locally, the cut locus of an n-dimensional manifold is homeomorphic to an $(n-1)$-dimensional simplicial complex.

Regarding the size of the cut locus, we have the following theorem by Itoh and Tanaka.

Theorem 4.4.8 (Itoh and Tanaka 1998). *Let M be a connected and complete Riemannian manifold of dimension n. For a point $p \in M$ the Hausdorff dimension of* $\operatorname{Tcut}(p)$ *is either* 0 *or* $n-1$*, and the Hausdorff dimension of* $\operatorname{cut}(p)$ *is an integer strictly smaller than n.*

Let us put now this result in the language of measures, which is more familiar to the optimisation community.

Corollary 4.4.9. $\operatorname{Tcut}(p)$ *has Lebesgue measure zero on T_pM and* $\operatorname{cut}(p)$ *has measure zero with respect to the Borel measure on M. If M is orientable,* $\operatorname{cut}(p)$ *has measure zero with respect to the measure induced by the Riemannian volume form.*

We can argue that, although the cut locus can introduce problems in practice, the problematic set is not too large in a measure-theoretic sense. On the other hand, even though the cut locus is of measure zero, it might have quite a wild topology. For M compact, the cut locus is connected as $M\backslash\{p\}$ is a deformation retract onto $\operatorname{cut}(p)$ given by flowing along geodesics emanating from p. On the other hand, in the non-compact case, the cut locus may have countably many components (Chavel 1970). Luckily, as we have mentioned at the beginning of this section, the manifolds on which we are interested in optimisation are particularly well-behaved, so this is never a problem in practice, as we will see empirically in Section 6.4. It is also worth noting that, when setting up the static trivialisation framework, we will often sample a point $p \in M$ according to some measure and then start at the point $v_0 = 0 \in T_pM$. For this reason, we will start the optimisation at the farthest point from the conjugate locus, which is advantageous in practice.

4.4.2.3 The adjoint of the Riemannian exponential

In order to implement the pullbacks along the exponential map in first order optimisation, it is necessary to compute the adjoint of the differential of the exponential map. For general Riemannian manifolds, this can be computed in terms of the radial vector field defined by geodesics. We will not use this result, but we believe that it may be of interest.

Proposition 4.4.10 (McAlpin 1965). *Let (M, g) be a Riemannian manifold and let $(p, v) \in TM$ such that $\gamma(t) = \exp_p(tv)$ does not have conjugate points for $t \in [0, 1]$ then*

$$(\mathrm{d}\exp_p)^*_v = (\mathrm{d}\exp_{\gamma(1)})_{-\dot\gamma(1)}$$

A proof of this result can be found in (Lang 1999, IX, Lemma 3.4).

Let us now compute the adjoint of the differential of the exponential on naturally reductive homogeneous spaces. For these spaces, we may simplify its computation to the computation of the adjoint of the Lie exponential on G.

Proposition 4.4.11. *Let $(G/H, g)$ be a naturally reductive homogeneous space and fix a point and a vector $(x, v) \in T(G/H)$. Let $\zeta_g = (\mathrm{d}\pi)_g \circ (\mathrm{d}L_g)_e|_{\mathfrak{m}} \colon \mathfrak{m} \to T_{gH}(G/H)$ be the linear isometry given by the naturally reductive structure. Then, letting $v_{\mathfrak{m}} \coloneqq \zeta_g^{-1}(v)$ for a $g \in \pi^{-1}(x)$ and denoting $\tilde g \coloneqq \operatorname{expm}(v_{\mathfrak{m}})$,*

$$(\mathrm{d}\exp_p)^*_v = \zeta_g \circ (\mathrm{d}\operatorname{expm})^*_{v_{\mathfrak{m}}} \circ (\mathrm{d}L_g)^*_{\tilde g} \circ (\mathrm{d}\pi)^*_{g\tilde g}$$

Proof. The formula follows by differentiating the expression for the Riemannian exponential in Proposition 3.4.10

$$\exp_x(v) = (\pi \circ L_g \circ \operatorname{expm} \circ \zeta_g^*)(v)$$

and taking adjoints. □

When G is a matrix Lie group, we may give a more computationally concrete formula for the adjoint of the Riemannian exponential. In this case, L_g is just left multiplication, which makes computing its adjoint with respect to the canonical basis direct. To compute the adjoint for the exponential, we use the fact that in this case, for matrix Lie groups, the Lie exponential is just the exponential of matrices (Proposition 3.5.2) and, in particular, an analytic map. For an analytic matrix map, we have a particularly simple formula for the adjoint. As we are working on $\mathbb{R}^{n\times n}$, we will identify $\mathrm{d}L_X$ with L_X to simplify the notation.[2]

Theorem 4.4.12. *Consider an analytic matrix function*

$$\begin{aligned} \psi\colon \mathbb{R}^{n\times n} &\to \mathbb{R}^{n\times n} \\ X &\mapsto \sum_{n=0}^{\infty} \frac{a_n}{n!} X^n \end{aligned}$$

We then have that, for the canonical inner product on $\mathbb{R}^{n\times n}$

$$(\mathrm{d}\psi)_X^* = (\mathrm{d}\psi)_{X^\intercal}. \qquad X \in \mathbb{R}^{n\times n}$$

Proof. We can compute the differential of ψ as

$$(\mathrm{d}\psi)_X(E) = \sum_{n=0}^{\infty} \Big(\frac{a_n}{n!} \sum_{i=0}^{n} X^i E X^{n-i} \Big).$$

By linearity, it is enough to compute the adjoint of functions of the form $X \mapsto X^i E X^{n-i}$.

Observe that the adjoint of the linear map given by left multiplication $L_A(X) = AX$ is exactly $L_{A^\intercal}$

$$\langle L_A(X), Y\rangle \coloneqq \operatorname{tr}((AX)^\intercal Y) = \operatorname{tr}(X^\intercal A^\intercal Y) = \langle X, L_{A^\intercal}(Y)\rangle.$$

In the case of right multiplication, we also get $R_A^* = R_{A^\intercal}$.

Finally, we just have to apply this formula to the functions $L_{X^i}(E) = X^i E$ and $R_{X^{n-i}}(E) = EX^{n-i}$, and noting that $X \mapsto X^i E X^{n-i} = L_{X^i}(R_{X^{n-i}}(E))$ we get the result. □

Remark 4.4.13 (Complex setting). The generalisation of this result to complex functions is direct, just computing the differential of the analytic function with conjugate coefficients in its Taylor series. In this case, one can interpret this theorem by saying that "the adjoint of the differential is the differential of the conjugate at the adjoint", noting the two different meanings of the word " adjoint" in the sentence.

One can also formulate the theorem for a holomorphic function defined just on an open subset $U \subseteq \mathbb{C}$, and define the function on matrices on the set of matrices such that their spectrum is contained in U, hence making sense also of functions like $\log(X)$.

[2] After obtaining this result, we thought that this should be folklore in some areas like functional analysis or numerical analysis. In fact, this result can be found without proof in (Higham 2008, p.66).

Putting this together with Proposition 4.4.11, we are able to compute the adjoint of the Riemannian map for manifolds such as the Stiefel manifold, the Grassmannian.

Theorem 4.4.14. *Let $(G/H, g)$ be a naturally reductive homogeneous space with G a subgroup of $\mathrm{U}(n)$ (resp. $\mathrm{SO}(n)$) and g the quotient metric induced from the canonical metric on $\mathbb{C}^{n\times n}$ (resp. $\mathbb{R}^{n\times n}$). With the notation in Proposition 4.4.11,*

$$(\mathrm{d}\exp_p)^*_v = \zeta_g \circ (\mathrm{d}\operatorname{expm})_{v^{\mathrm{H}}_{\mathfrak{m}}} \circ (\mathrm{d}L_g)^*_{\tilde{g}} \circ (\mathrm{d}\pi)^*_{\tilde{g}}$$

where $-^{\mathrm{H}}$ denotes the conjugate transpose.

In Section 6.3.2, we will explore how to numerically approximate both the exponential and its adjoint efficiently on a GPU.

The conjugate locus on naturally reductive homogeneous spaces has a particularly simple structure, given that it is just the projection as the conjugate locus of the Lie group. If this is a matrix Lie group then the conjugate locus can be computed by algebraic means, as this is set of points on which the differential of the exponential of matrices fails to be surjective. Using the formula for the differential of the exponential and looking at its eigenvalues, it is easy to show the following.

Theorem 4.4.15. *Let G be a matrix Lie group. The exponential of matrices is analytic, with analytic inverse on a bounded open neighbourhood of the origin given by*

$$U = \{A \in \mathfrak{g} \mid |\mathrm{Im}(\lambda_i(A))| < \pi\}.$$

Proof. See for example (Rossmann 2002, Proposition 7'). □

In particular, this gives us a neighbourhood of radius π—or in general a whole band—around the identity where the exponential is a diffeomorphism.

Remark 4.4.16 (Symmetric spaces). For compact groups with a bi-invariant metric, where at the identity the Lie exponential and the Riemannian exponential coincide, this gives a tight bound for the conjugate radius. These computations may be extended to symmetric spaces by means of Jacobi fields. On a symmetric space $(G/H, g)$, as the curvature tensor is covariantly constant, the Jacobi equation—*i.e.*, the equation that the differential of the exponential satisfies (*cf.*, Proposition 5.3.4)—has constant coefficients. As such, it is possible to express it in terms of the Lie bracket on G. When G is compact, using the roots of G and H, it is possible to compute the largest radius on which the Riemannian exponential is a submersion. The details can be found in (Rauch 1966)

Remark 4.4.17 (A note on the Lie exponential). The Lie exponential has the advantage over the general Riemannian exponential that it is straightforward to compute for matrix Lie groups, while the Riemannian exponential requires looking for a suitable metric on which the geodesics are easy to compute. On the other hand, once we have a suitable Riemannian exponential, it is possible to give convergence results with respect to the different metrics involved in the Riemannian submersion $\pi\colon G \to M$. Furthermore, the metrics that we work with are often complete, so if M is connected, the Riemannian exponential is surjective, and it is a diffeomorphism onto all the manifold but a set of measure zero—the

conjugate locus (Corollary 4.4.9). On the other hand, for some groups such as $\mathrm{GL}(n,\mathbb{R})$ or $\mathrm{SL}(n,\mathbb{R})$ the Lie exponential is not surjective, which is problematic.

To be able to use maps that are not surjective, and to mitigate the problem of converging to a critical point of the parametrisation, we introduce the framework of dynamic trivialisations.

4.5 Dynamic Trivialisations

In this section, we will exploit vector bundle structure of the tangent bundle to alleviate the problems of lack of surjectivity or of convergence to a critical point of a static trivialisation. We will assume, as it is often the case, that we have access to a retraction $r\colon TM \to M$.

In this setting, to approximate a minimiser of f, we may choose a point $p \in M$ and optimise on a fixed tangent space T_pM, performing gradient descent on the function $f \circ r_p$. We will call p the **basepoint**. If we simply fix the basepoint p, this is exactly the setting of static trivialisations, where we just pulled back the problem by a fixed map. On the other hand, it may be the case that either $r_p\colon T_pM \to M$ is not surjective or that it has critical points. In either case, given that we do not have access to a map but to a whole family of maps indexed by $p \in M$, we could change the basepoint if we detect that we are converging to a critical point of r_p or if we see that we are not exploring all the manifold.

To formalise this, let $p_0 \in M$ be some starting point. Consider the iterates in $T_{p_0}M$

$$v_{0,k+1} = v_{0,k} - \eta_{0,k}\nabla(f \circ r_{p_0})(v_{0,k})$$

with $v_{0,0} = 0$. Then, if after K iterations we observe that we are too close from a critical point of r_{p_0}, or we consider that we are not exploring a part of M due to r_{p_0} not being surjective, then we take $p_1 = r_{p_0}(v_{0,K})$, set $v_{1,0} = 0$ and continue the optimisation process on $T_{p_1}M$.

These ideas can naturally be translated into Algorithm 1 given a boolean stopping rule `stop`. We call it the **dynamic trivialisations framework**. The term "dynamic" comes from the fact that we choose an appropriate basepoint dynamically, according to the stopping rule `stop` into which we can embed some prior knowledge about M and structure of the problem.

Algorithm 1 Dynamic trivialisation framework

Require: A starting point $p_0 \in M$, a boolean statement `stop`, a map $r\colon TM \to M$, a function $f\colon M \to \mathbb{R}$

for $i = 0, \dots$ **do**
  $v_{i,0} = 0$
  for $k = 1, \dots$ **do**
    $v_{i,k} = v_{i,k-1} - \eta_{i,k-1}\nabla(f \circ r_{p_i})(v_{i,k-1})$ ▷ Optimise on $T_{p_i}M$
    if `stop` **then** ▷ Stopping condition
      break
  $p_{i+1} = r_{p_i}(v_{i,k})$ ▷ Update pullback point

The procedure described in Algorithm 1 accounts for changing the metric on M every time the stopping rule fires. An example of a stopping rule to detect that we are converging to a critical point of φ with a stopping rule of the form

$$\texttt{stop} \equiv \frac{\|\nabla(f \circ r_{p_i})(v_{i,k})\|}{\|\nabla f(r_{p_i}(v_{i,k}))\|} < \varepsilon.$$

This algorithm has two interesting limiting cases.

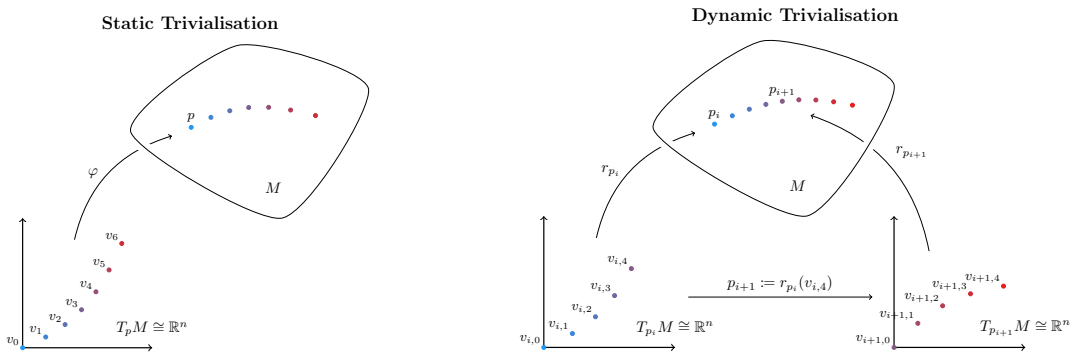

Figure 4.3: Example of the pullback method vs. the dynamic trivialisation procedure changing the basepoint after 4 steps.

Generalisation of pullbacks. If `stop` $\equiv$ `False`, that is, if we never change the basis p_0, it reduces to the static trivialisation framework with the map r_{p_0}.

Generalisation of Riemannian gradient descent. In the case `stop` $\equiv$ `True`, we are changing the basis of the trivialisation on every step. In this case, this method recovers exactly Riemannian gradient descent using r as a retraction. To see this, just note that by the chain rule and the definition of a retraction

$$\mathrm{d}(f \circ r_{p_i})_0 = (\mathrm{d}f)_{r_{p_i}(0)} \circ (\mathrm{d}r_{p_i})_0 = (\mathrm{d}f)_{p_i}.$$

From this it follows that

$$\nabla(f \circ r_{p_i})(0) = \nabla f(p_i)$$

so the update rule simplifies for a learning rate $\eta_{i,0} = \eta_i > 0$ can be rewritten as

$$v_{i,1} = -\eta_i \nabla f(p_i) \qquad p_{i+1} = r_{p_i}(-\eta_i \nabla f(p_i))$$

and p_{i+1} are exactly the iterates given by doing Riemannian gradient descent using the retraction r.

For this reason, we can see dynamic trivialisations as an interpolation between the pullback method using r_{p_0} and Riemannian gradient descent along r.

Remark 4.5.1 (Retractions vs. more general maps). We ask for the map $r\colon TM \to M$ to be a retraction for simplicity, but this is not strictly necessary. If we do not have that $r_p(0) = p$ then, when changing from r_{p_i} to $r_{p_{i+1}}$ we would just have to take the vector on $T_{p_{i+1}}M$ that is mapped through $r_{p_{i+1}}$ to the point at which we are at the moment. Note that it is not necessary for $(\mathrm{d}r_p)_0$ to be the identity either, but simply to be full rank so that it defines a new metric locally. In practice, it would also be necessary for the neighbourhood on which the retraction is non-singular to be large enough, as it is the case with the Riemannian exponential. For other retractions, this is a property that should be studied case by case.

Dynamic trivialisations have several practical advantages over RGD.

Errors do not accumulate Consider usual Riemannian descent on $\mathrm{SO}(n)$ with a bi-invariant metric along the Riemannian exponential map (*cf.*, Section 3.5.1). Unrolling the update rule, we get that after t steps, we will be at the point

$$x_t = x_0 \operatorname{expm}(-\eta \nabla f(x_0)) \cdot \ldots \cdot \operatorname{expm}(-\eta \nabla f(x_{t-1})).$$

This computation is exact in theory, but in practice, one has to compute the exponential of matrices at every step, which will lead to some numerical error in every step. In particular, every matrix $\operatorname{expm}(-\eta \nabla f(x_i))$, when represented in memory, will not be orthogonal but just *very close* to orthogonal. These errors will accumulate, which means that the matrix x_t will deviate from $\mathrm{SO}(n)$ as t increases. This is problematic in practice since, as we will showcase in the experiment section Section 6.4, optimisation on manifolds is often used in practice to regularise very badly conditioned optimisation problems. In these problems, the Lipschitz constant of the function being optimised is rather large, and projecting back to the manifold often affects very negatively the convergence of the algorithm.

Consider now the static trivialisation framework with the Riemannian exponential on a complete and connected Riemannian manifold. After choosing a frame, we are effectively pulling back the problem to $\mathbb{R}^n$—*e.g.*, to $\mathrm{Skew}(n)$ in the case of $\mathrm{SO}(n)$. As this is a vector space, its elements can be represented in memory exactly by the use of a frame $\zeta \colon T_pM \to \mathbb{R}^n$. It is for this reason that the static trivialisation framework does not present this accumulation of errors. On the other hand, by construction, if we have a surjective map from a Euclidean space onto a manifold that is not simply connected, it will necessarily have critical points, as it was the case with the conjugate locus for the Riemannian exponential map.

Dynamic trivialisations allow choosing a sweet-spot between these two situations depending on the problem. For example, consider that we know a priori the conjugate radius of the manifold in hand, which is often the case. With this information, we may run a coarse estimate based on the norm of the gradients so far and the step size of whether the current iterate may be too close to a singular point of the exponential map. If this cheap check passes, we may then run an actual check on the manifold to see if the current point is on the conjugate locus—on $\mathrm{SO}(n)$ this would account for doing an SVD of a matrix and checking the pairwise distance between eigenvalues values. If we are then ε-close to the conjugate locus of p, we may then change the basepoint and so on.

The flexibility of the framework allows for the quantities involved in the estimation to be tuned in an online manner, where one may try to estimate some of the hyperparameters involved in the stopping rule according to the change of the norm of the gradient when one changes the point, or even second order information.

Gradient computations It is very common to find a section in papers on optimisation on manifolds where the equation for the Riemannian gradient on the manifold is derived for the manifold considered in the paper. These computations are always standard, and it often accounts for finding a projection from some matrix space onto a representation of the tangent space of the manifold and computing its adjoint. The computation of the adjoint of this linear map is purely mechanic, but it is often rather cumbersome, even more so in the case of the Riemannian Hessian.

The static and dynamic trivialisation frameworks implement this via composition of functions, after choosing a suitable frame $\zeta\colon T_pM \to \mathbb{R}^n$ on the tangent space. After this choice, one gets the Riemannian gradient without any extra effort, as the adjoint method is automatically computed by the autodiff engine, hence saving a considerable amount of effort. We will show some concrete examples of this in Section 6.1.2.

Modularity by design Another benefit of this framework in this context is that it takes full advantage of the modularity of autodiff frameworks. Consider that we want use the Lie exponential to perform optimisation on a Lie subgroup of a Lie group on which we have already implemented the Lie exponential. If that is the case, we know that the Lie exponential on H will be just the Lie exponential on $\mathfrak{g}$ restricted to $\mathfrak{h}$. As such, we just need to consider the map that embeds $\mathfrak{h}$ into $\mathfrak{g}$ and precompose by this parametrisation to get the exponential at H. Naturally reductive homogeneous spaces provide the same modularity with the Riemannian exponential map. We will explore this further in Section 6.2.2 when looking at the class system in GeoTorch.

Use of different optimisers Arguably the most important thing that this approach brings to the table is that, as one is effectively working on $\mathbb{R}^n$, the update rule in Algorithm 1 need not be that coming from gradient descent but could be that from other optimiser, such as ADAM or ADAGRAD. We will see that these optimisers bring a considerable advantage over traditional gradient descent in the context of neural networks in the experiments (Section 6.4).

Remark 4.5.2 (A note on adaptive optimisers and change of basepoint). When using adaptive optimisers, one has to be careful when performing a change of basepoint. Adaptive optimisers have extra parameters where they keep an approximation of the curvature of the function at the current point. As such, when changing the basepoint, as we are effectively changing the metric, this notion of curvature changes as well. While it is possible to correct for this, it is too expensive to do in practice. Therefore, even though one may leave this estimator hoping that it is a *good* starting point for the estimation of the curvature with respect to this new metric, sometimes destabilises the training. As a rule of thumb, it is preferable to just zero out these estimations and start the estimation from scratch when using adaptive optimisers.

Of course the use of these optimisers is just recommended together with a stopping rule that does not change the basepoint very often like that given by the closeness to the conjugate locus in the case of the exponential. In the general case where one may change the basepoint after every optimisation step, as one does to recover Riemannian gradient descent, there is no hope in that this scheme optimiser would yield a sensible update rule when used with an adaptive optimiser.

Remark 4.5.3 (Vector bundles). This whole construction may be carried for a general vector bundle $\pi\colon E \to M$ with a map $r\colon E \to M$. We chose to carry the presentation in the simpler setting of $E = TM$, since we do not aware of any setting where using a more general bundle is of practical interest.

Chapter 5

First and Second Order Bounds on the Riemannian Exponential

In this chapter we present the main theoretical results of the thesis: Novel second order bounds on the Riemannian exponential map in terms of the curvature of the manifold. With these we then go and prove some conditional convergence results for the static and dynamic trivialisation framework when using the Riemannian exponential map. This kind of results are the first of their kind, to the best of our knowledge. The results presented in this chapter are as general as they can be, and they can be used to give concrete bounds for different manifolds, as many papers in the literature on optimisation on manifolds give their results in terms of second order bounds of retractions.

Outline of the chapter. We start by setting-up the problem and giving a literature review, linking previous theoretical approaches to the one given here. In Section 5.2, we set the notation, and we recall a few results from differential and Riemannian geometry that will be used throughout the chapter. In Section 5.3, we give a self-contained proof of first order bounds on the Riemannian exponential map, which go by the name of Rauch's theorem. We include it as we believe that this proof is not very well-known in the optimisation on manifolds community. We emphasise the maximal domains on which the inequalities hold, which is often neglected in the literature. In Section 5.4, we prove the second-order bounds, following the proof structure given in (Kaul 1976) but heavily simplifying the proof and the final bounds. In Section 5.5, we look at the form that these bounds take for some manifolds used in optimisation. In Section 5.6, we show how to use these bounds to prove conditional convergence of the static and dynamic trivialisation framework for functions of bounded Hessian on manifolds of bounded geometry. To finish the chapter, we include a short section giving a simple algebraic way to tell when is a given retraction the exponential map of a connection.

5.1 Introduction

5.1.1 The problem and overview of assumptions

We are interested in approximating critical points of a—possibly non-convex—function defined on a manifold

$$\min_{x \in M} f(x).$$

In particular, we are interested in giving global convergence bounds for such problems. These problems form a particularly well-behaved subset of the field of constrained optimisation. Examples of the importance of this family of problems are ubiquitous in engineering, statistics, and machine learning. For example, they have found applications in image processing, using Grassmannians for tracking subspaces; analysis of time series in deep learning with the orthogonal group to avoid vanishing and exploding gradient problems; Bayesian statistics and machine learning with the manifold of positive definite matrices in kernel methods and metric learning; the space of low-rank and fixed-rank matrices for low-rank models; and the hyperbolic space for word embeddings, among many others.

The approach that we investigate in this chapter is that of using the exponential map to pullback the problem from the whole manifold to a fixed tangent space, converting the problem into an unconstrained one over $T_pM \cong \mathbb{R}^n$

$$\min_{v \in T_pM} f(\exp_p(v)). \tag{5.1}$$

For this to be a sensible method, the exponential map has to be surjective, so we will assume (M, g) to be **connected and complete** throughout the chapter.

For some manifolds, such as the hyperbolic space or the space of symmetric positive definite matrices (and more generally, Hadamard spaces), one may find (geodesically) convex functions which can be optimised efficiently (Bačák 2014). On the other hand, when the manifold is compact, as is the case for the sphere, or when dealing with orthogonality constraints (Edelman, Arias, and Smith 1998), there exists no convex function besides the constant ones, so we inevitably have to deal with non-convex problems.

To this end, we will consider a function f which is of **α-bounded Hessian** (*i.e.*, with bounded Hessian in operator norm) and we ask ourselves whether $f \circ \exp_p$ is of α'-bounded Hessian for any α'. The answer to this problem will be: Not in general. For example, for the hyperbolic space, the exponential map grows as $\cosh(t)$, which has unbounded first and second derivatives. For this reason, we will have to content ourselves with proving tight bounds on the growth of the Hessian of $f \circ \exp_p$ on a neighbourhood of a given radius.

When it comes to the assumptions on the manifold, we will look at the family of manifolds of **bounded first order geometry**.

Definition 5.1.1 (Bounded first order geometry). We say that a Riemannian manifold (M, g) has $(\delta, \Delta, \Lambda)$**-bounded geometry** if it has uniformly bounded injectivity radius $\mathrm{r_{inj}} > 0$ and we have bounds on the sectional curvature and the derivative of the curvature tensor R

$$\delta \leq \sec \leq \Delta \qquad \|\nabla R\| < \Lambda.$$

These manifolds have found many uses in the area of PDEs on manifolds and geometric analysis. In particular, any compact manifold with any metric is of bounded geometry. On the other hand, this family does not impose topological restrictions on M as any differentiable manifold admits a complete metric of bounded geometry (Greene 1978). We will introduce these manifolds in a bit more depth in Section 5.4.2. For a much more in-depth introduction to this family of manifolds see the thesis (Eldering 2012).

Proof strategy. We will implement the following strategy:

1. Give first and second order bounds for $\exp_p$ that depend on the curvature of the manifold (Sections 5.3 and 5.4)
2. Instantiate these bounds on some manifolds of interest (Section 5.5)
3. Use the first and second order bounds for $\exp_p$ to estimate a bound on the norm of the Hessian of $f \circ \exp_p$ on a bounded domain (Section 5.6)
4. Use these bounds to prove the convergence of the dynamic trivialisation framework (Section 5.6)

5.1.2 Previous work

General pullbacks in constrained optimisation. The idea of pulling back a problem from a space with a complex geometry to a simpler one via a particular family of maps lies at the core of the area of constrained optimisation. Mirror descent (Nemirovsky and Yudin 1983) and Lagrange multipliers are examples of methods that allow simplifying a constrained optimisation problem into one in a simpler space. It is also a recurrent theme in the literature on optimisation on manifolds. We have the Burer–Monteiro method (Burer and Monteiro 2003; Burer and Monteiro 2005) which parametrises the space of low-rank matrices via the multiplication of two tall matrices. Other approaches via different factorisations have also been explored, for example via an SVD (Vandereycken 2013) or other factorisations like the polar factorisation (Mishra et al. 2014). Similar parametrisations have been explored for the algebraic variety of matrices of rank at most k—low-rank optimisation—and manifolds such as positive semidefinite matrices of a fixed rank (Vandereycken, Absil, and Vandewalle 2013; Massart and Absil 2020). The idea behind these algorithms is that of parametrising a space in terms of simpler ones, effectively pulling back the problem to an easier one and then mapping back the solution of the simpler problem to the complex space via the parametrisation map.

Pullbacks along the exponential in numerical analysis. Instances of (5.1) can be found all throughout the area of numerical analysis. For example, in the study of the centre of mass in the manifold of $\mathrm{Sym}^+(n)$ matrices (Arsigny, Commowick, et al. 2006; Arsigny, Fillard, et al. 2006); continuous dynamics on Lie groups, and their discretisations (Magnus 1954; Iserles and Nørsett 1999; Iserles, Munthe-Kaas, et al. 2000); optimisation on compact Lie groups (Lezcano-Casado and Martínez-Rubio 2019); optimisation on non-compact Lie groups (via the Lie exponential) (Mahony and Manton 2002); and probabilistic methods on manifolds for their use in machine learning (Falorsi et al. 2019). More recently, these ideas were unified into the framework of **static trivialisations** (Lezcano-Casado 2019).

The main idea behind pulling back an optimisation problem from a manifold to a flat space is that of heavily simplifying the problem at hand, going from a manifold which might have a complex topology and geometry, to one that is flat and has trivial topology. Of course, that comes at a cost, as this change of topology makes the new function $f \circ \exp_p$ have more critical points. Luckily, for any differentiable manifold and any smooth metric, this set of problematic points, called the **conjugate locus**, has measure zero by a theorem of Sard (Sard 1965). Since the conjugate locus is a subset of the **cut locus**, it is far from the

point p. This is because the cut locus can vaguely be regarded as "the set that is opposite to the point p". For example, on a sphere, the cut locus of a point p is its antipode and on a cylinder, the cut locus of a point is the whole line opposite to that point. As such, the cut locus is, in some sense, *as far from p as possible*, and so is the conjugate locus. At the expense of this technicality, one may heavily simplify the problem of optimising a function on a space with a non-trivial topology to a Euclidean problem. We will expand on this technical point in Section 5.2.

An idea surprisingly similar to this one was outlined in (Manton 2015, Section 7) in the context of second order optimisation and Newton methods on manifolds. Even though the strategy is very similar to that of dynamic trivialisations, the nature of the proofs developed in that paper and the theoretical background are essentially different. It would be of interest to look further into the connections between the approach presented here and that presented in that paper, as they can surely benefit of each other.

Pullbacks along the exponential map: Trivialisations. If we pullback the problem to the tangent space at each iteration of the algorithm along a retraction, we recover Riemannian Gradient Descent (RGD) along that retraction (Boumal, Absil, and Cartis 2019). It was then noted in (Lezcano-Casado 2019) that one may change the point p in (5.1) to avoid converging to a point in the conjugate locus of p. In fact, one may change the point according to an arbitrary stopping rule giving the following meta-algorithm:

1. Compute gradient steps of the function $f \circ \exp_{p_i}$ to obtain iterates $v_{i,k} \in T_{p_i}M$ for $k = 1, 2, \dots$
2. If we are close to a point in the conjugate locus of p_i, we set $p_{i+1} = \exp_{p_i}(v_{i,k})$ and repeat

This is the algorithm presented in Algorithm 1 choosing the Riemannian exponential map rather than a general retraction (Lezcano-Casado 2019)

Algorithm 2 Dynamic trivialisation framework

Require: A starting point $p_0 \in M$, a boolean statement `stop`

for $i = 0, \dots$ **do**
  $v_{i,0} = 0$
  for $k = 1, \dots$ **do**
    $v_{i,k} = v_{i,k-1} - \eta_{i,k-1} \nabla (f \circ \exp_{p_i})(v_{i,k-1})$ ▷ Optimise on $T_{p_i}M$
    if `stop` **then** ▷ Stopping condition
      break
  $p_{i+1} = \exp_{p_i}(v_{i,k})$ ▷ Update pullback point

Two possible stopping rules stand out over the others. If we let `stop` $\equiv$ `False`, then we recover the **static-trivialisation framework** of pulling back the function to a fixed tangent space as in (5.1). On the other hand, if `stop` $\equiv$ `True`, then the algorithm is exactly Riemannian gradient descent, as mentioned before. The stopping rule of being too close to the conjugate locus could be then written as

$$\texttt{stop} \equiv \frac{\|\nabla (f \circ \exp_{p_i})(v_{i,k})\|}{\|\nabla f(x_{i,k})\|} < \varepsilon.$$

The ideas of pulling back the problem at the current point of the optimisation and then taking a few steps on the tangent space have been recently explored in (Criscitiello and Boumal 2019; Sun, Flammarion, and Fazel 2019) in the context of escaping saddle points for optimisation on manifolds.

Retractions with bounded second and third order derivatives. To be able to prove convergence of the dynamic trivialisation framework, we will give novel bounds on the norm of the Hessian of the exponential map in terms of the curvature of the manifold. These bounds are one example of second order bounds for a large family of retractions. These bounds lie at the heart of the convergence results of Riemannian methods via retractions, where it is often referred to as the **L-smoothness of the retraction**. Examples of work under this assumption are adaptive methods on matrix manifolds (Kasai, Sato, and Mishra 2018), quasi-Newton methods on manifolds (W. Huang, Gallivan, and Absil 2015), stochastic methods with variance reduction (Kasai, Sato, and Mishra 2018), and general convergence of Newton methods in Riemannian manifolds (Ferreira and Svaiter 2002), among many others. The bounds developed in this chapter give exact rates of growth for the norm of the Hessian of the retraction in terms of the curvature of the manifold when the retraction is the exponential map, which falls exactly in the framework of these papers and many others.

Third order and higher order bounds are used when trying to converge to second order optima and escaping saddle points (Criscitiello and Boumal 2019) and when adapting Newton methods to manifolds (Agarwal et al. 2020), and other higher order methods. The computations of higher order bounds is analogous to that of second order bounds. In this chapter, we present all the necessary techniques to give n-th order bounds for a manifold of n-bounded geometry (Definition 5.4.3), although we do not perform these computations explicitly. Bounds of this flavour were first developed in (Eichhorn 1991), where the rate of growth of the n-th derivative of the Christoffel symbols in normal coordinates is estimated.

5.2 Differential Geometry: Conventions and Notation

In this section, we establish the notation used to refer to some recurrent objects, such as the distance function, the segment domain, mixed derivatives and pullbacks of connections. We also use it to recall some definitions from differential geometry that are not that commonly seen in the area of optimisation, such as the definition of the Lie derivative and covariant derivative of tensors.

The manifold We will always work on a connected and complete Riemannian manifold (M, g) of dimension at least 2 and at least C^4 regularity. We will denote the sectional curvature on the plane defined by two vectors $u, v \in T_pM$ as $\sec(u, v)$. When we write $\sec \leq \Delta$ for a constant $\Delta \in \mathbb{R}$, we mean that for every $p \in M$, $u, v \in T_pM$, $\sec(u, v) \leq \Delta$. We will implicitly the natural identification $T_{(p,v)}(T_pM) \cong T_pM$ for every $v \in T_pM$.

Distance function, segment domain, conjugate and cut locus We summarise here some well-known results on some objects associated to the geometry of a Riemannian manifold. Most of these were already presented more formally in Definition 4.4.6 and Proposition 4.4.7.

For a point $p \in M$, we define the **segment domain** as

$$\operatorname{seg}(p) \coloneqq \{v \in T_pM \mid \exp_p(tv) \text{ is length minimising for } t \in [0, 1]\}$$

and we denote its interior by $\mathrm{seg}^0(p)$. The set $\mathrm{seg}^0(p)$ is a star-shaped open neighbourhood of 0 in T_pM. We will also write

$$U_p \coloneqq \exp_p(\mathrm{seg}^0(p)).$$

U_p is sometimes referred as a **normal neighbourhood of** p, and its complement is called the **cut locus** $\mathrm{cut}(p) \subseteq M$. We have that the map

$$\exp_p|_{\mathrm{seg}^0(p)} \colon \mathrm{seg}^0(p) \to U_p$$

is a diffeomorphism. In particular, on these sets, we have that $\exp_p^{-1}$ exists and that $\mathrm{d}\exp_p$ is full rank.

A conjugate point $\exp_p(v) = q$ of p is one such at which $(\mathrm{d}\exp_p)_v$ is not full rank. We denote the set of all these points $\mathrm{conj}(p)$. We have that $\mathrm{conj}(p) \subseteq \mathrm{cut}(p)$. By a theorem of Sard, $\mathrm{conj}(p)$ has measure zero in M (Sard 1965). Even more, $\mathrm{cut}(p)$ has Hausdorff dimension at most $n-1$ (Itoh and Tanaka 1998). As such, U_p is an open radially convex neighbourhood of p that covers almost all the manifold.

We will write $r(x) \coloneqq d(x,p)$ for the distance to a fixed point p. This function is differentiable on $U_p \backslash \{p\}$, with gradient the unit radial vector field emanating from p. In particular, we have that for a geodesic γ starting at p, $\nabla r|_\gamma = \dot\gamma$. The cut locus of p is exactly the set of points other than p at which the distance function r is not differentiable.

Einstein convention Whenever we refer to coordinates $\{x^i\}$ on all of U_p, we will always assume that these are the normal coordinates given by $\exp_p^{-1}$ on T_pM. We will denote $\partial_i \coloneqq \frac{\partial}{\partial x^i}$ for short. We will use Einstein's summation convention that if an index appears as a super-index and a sub-index in a formula, it means that we are summing over it. For example, for a vector field X in local coordinates we write

$$X = \sum_{i=1}^n X^i \partial_i = X^i \partial_i.$$

Remark 5.2.1. Almost all the results in this chapter can be developed working just on a neighbourhood of a geodesic rather than on the whole U_p. We will make sure of make this explicit in each result throughout the chapter.

Connections and derivations We will write $D_X f \coloneqq \mathrm{d}f(X)$ for the directional derivative of a function along a vector field X to disambiguate with the gradient of a function, which we will denote by ∇f its gradient. We denote by ∇ the Levi-Civita connection on U_p and by ∇^{flat} the pushforward of the flat connection on T_pM along $\exp_p$. In the same way that we do for the norms, we will abuse the notation and also denote by ∇ the associated connections defined by ∇ on associated bundles.

We recall that connections on tensor bundles are defined so that the Leibnitz rule holds. For example, for the Hessian and three vector fields X, Y_1, Y_2, the Leibnitz rule reads

$$D_X(\mathrm{Hess}\, f(Y_1,Y_2)) = (\nabla_X \mathrm{Hess}\, f)(Y_1,Y_2) + \mathrm{Hess}\, f(\nabla_X Y_1, Y_2) + \mathrm{Hess}\, f(Y_1, \nabla_X Y_2)$$

so $\nabla \mathrm{Hess}\, f$ is given by the $(0,3)$ tensor

$$(\nabla_X \mathrm{Hess}\, f)(Y_1,Y_2) \coloneqq D_X(\mathrm{Hess}\, f(Y_1,Y_2)) - \mathrm{Hess}\, f(\nabla_X Y_1, Y_2) - \mathrm{Hess}\, f(Y_1, \nabla_X Y_2).$$

We can do the same for a tensor T of type $(0,s)$. We define its covariant derivative ∇T as the tensor of type $(0,s+1)$ such that the Leibnitz rule holds

$$(\nabla_X T)(X_1,\dots,X_s) \coloneqq D_X(T(X_1,\dots X_s)) - \sum_{i=1}^{s} T(X_1,\dots,\nabla_X X_i,\dots,X_s).$$

For example, if we have in local coordinates the differential of a function is a $(0,1)$ tensor $\mathrm{d}f = f^i \mathrm{d}x_i$. We may compute its Hessian in local coordinates as the $(0,2)$ tensor given by

$$\nabla \mathrm{d}f = \nabla(f^i \mathrm{d}x_i) = \mathrm{d}f^i \otimes \mathrm{d}x_i + f^i \nabla \mathrm{d}x_i$$

where we have used that the connection on functions is just the differential, by definition.

The Lie derivative of tensors is defined in the same way. For a $(0,s)$ tensor T, the Lie derivative of T, $\mathcal{L}_X T$ in the direction of a tensor X is defined as the $(0,s)$ tensor

$$(\mathcal{L}_X T)(X_1,\dots,X_s) \coloneqq D_X(T(X_1,\dots X_s)) - \sum_{i=1}^{s} T(X_1,\dots,\mathcal{L}_X X_i,\dots,X_s).$$

Pullback connections When dealing with a smooth curve $\gamma\colon [0,r] \to M$, we will sometimes want to derive a vector field X along it. We will write $\nabla_{\partial_t} X$ for the **covariant derivative along** γ, that is, the **pullback connection along** γ.

We will sometimes write the covariant derivative along γ as $\dot{X} \coloneqq \nabla_{\partial_t} X$. This comes from the fact that if we choose a parallel frame $\{e_i\}$ along γ, that is $\nabla_{\partial_t} e_i = 0$, we have that by the Leibnitz rule

$$\dot{X} = \nabla_{\partial_t}(X^i e_i) = \nabla_{\partial_t}(X^i) e_i + X^i \nabla_{\partial_t}(e_i) = \dot{X}^i e_i$$

where $\dot{X}^i$ are just the regular derivatives of the coordinate functions. With this notation, the usual equation for geodesics simply reads $\ddot{\gamma} = 0$.

If instead of a curve we have an embedded surface

$$\begin{aligned} c\colon [0,r]\times[-\varepsilon,\varepsilon] &\to M \\ (t,s) &\mapsto c(t,s) \end{aligned}$$

we will analogously write ∂_s for the vector field in the direction of the second component. Suppose that we have a vector field J along $\gamma(t) \coloneqq c(t,0)$ given by

$$J(t) = \frac{\partial c}{\partial s}(t,0) = \mathrm{d}c(\partial_s)|_\gamma$$

where ∂_s is the coordinate vector field on the second component of $[0,r]\times[-\varepsilon,\varepsilon]$. We will abuse the notation and write for a vector field X along γ

$$\nabla_J X \coloneqq \nabla_{\partial_s} X$$

in the same way that we may write the equation for the geodesics as

$$\nabla_{\dot\gamma}\dot\gamma \coloneqq \nabla_{\partial_t}\dot\gamma = 0.$$

Sometimes, we will also pullback a connection along the exponential map $\exp_p$. For the distance function r to p, we have the radial vector field on $U_p\backslash\{p\} \subseteq M$ given by ∇r. By Gauss's lemma, the

pullback of this vector field to $\operatorname{seg}^0(p)\backslash\{0\} \subseteq T_pM$ via the exponential map is exactly the radial vector field on the tangent space ∂_r, that is

$$\mathrm{d}\exp_p(\partial_r) = \nabla r.$$

With this notation, we can write the equation for the geodesics emanating from p on all of $U_p\backslash\{p\}$ using the pullback connection along $\exp_p$ as

$$\nabla_{\partial_r}\nabla r = 0.$$

Hessian and iterated Hessian If we want to evaluate the $(0,2)$ Hessian twice on the same vector, we may do so by simplifying this problem to just performing a Euclidean derivative. Let γ be the geodesic such that $\dot\gamma(0) = X$

$$\operatorname{Hess} f(X,X) = (f\circ\gamma)''(0) = D_{\dot\gamma(0)}\langle\nabla f,\dot\gamma\rangle = \langle\nabla_{\dot\gamma(0)}\nabla f,\dot\gamma(0)\rangle = \nabla\mathrm{d}f(X,X). \tag{5.2}$$

In other words, $\nabla\mathrm{d}f$ serves as a coordinate-free definition of the $(0,2)$ Hessian of f. In the same way, $\nabla\nabla f$ is the $(1,1)$ Hessian of f.[1]

We also recall the definition of the iterated Hessian of a function as a $(0,2)$ tensor. This is also best introduced in its $(1,1)$ form by using that the $(1,1)$ Hessian is given by $\operatorname{Hess} f(X) = \nabla_X\nabla f$ so that

$$\operatorname{Hess}^2 f(X) \coloneqq (\operatorname{Hess} f\circ\operatorname{Hess} f)(X) = \nabla_{\nabla_X\nabla f}\nabla f.$$

Its $(0,2)$ version is consequently defined as

$$\operatorname{Hess}^2 f(X,Y) \coloneqq \langle\operatorname{Hess} f(\operatorname{Hess} f(X)),Y\rangle = \operatorname{Hess} f(\operatorname{Hess} f(X),Y) = \langle\operatorname{Hess} f(X),\operatorname{Hess} f(Y)\rangle \tag{5.3}$$

where we have used that the Hessian is symmetric. This formula shows that $\operatorname{Hess}^2 f(X,Y)$ is also symmetric.

5.3 First Order Bounds for the Exponential Map

In this section, we give bounds on the norm of the differential of the exponential map of a manifold with bounded sectional curvature. In particular, if the sectional curvature of any plane of M is bounded above and below by $\delta \le \sec \le \Delta$, for a ball $B_p(r)$, the bounds will be of the form

$$f_\Delta(r)\|w\| \le \|(\mathrm{d}\exp_p)_v(w)\| \le f_\delta(r)\|w\| \qquad \forall w\in T_pM, \|v\|\le r$$

for suitable functions $f_\delta, f_\Delta\colon [0,r]\to\mathbb{R}^+$.

First order bounds for the exponential map have been known, at least for surfaces, since the times of Cartan (Cartan 1928), and are well-known in areas such as comparison geometry or PDEs, but they are not so known in the area of optimisation on manifolds. The tight bounds that we will present here are often referred to as **Rauch's theorem** as he was the first to show these bounds in the positive curvature-case (Rauch 1951) and later in the general case (Rauch 1959).

There are quite a few proof techniques of these theorems. There exist proofs via the second variation formula of the energy (Spivak 1999), through bounds on the norm via the Jacobi equation (Jost 2017), by

[1] Note that in the expression $\nabla\nabla f$, the first ∇ represents a connection, while the second represents the gradient of f. This distinction will always by clear, as we represent the directional derivative on functions by the notation D_Xf.

bounding Riccati equation on self-adjoint operators and integrating the result (Eschenburg 1994), or by the study of the distance function to a given point (Gromov 1981). We choose to present here this fourth approach.

The approach presented here was first introduced by Gromov in the context of volume bounds, in what's called now the Bishop–Gromov theorem. This approach has the advantage of yielding upper and lower bounds for positive and negative curvatures at the same time. Other approaches need different techniques for the upper and lower bounds, or they just give some weaker version of the theorem with constraints on the sign of the curvature. This approach also has a more geometric flavour as it accounts for bounding the principal curvatures of geodesic balls on the manifold. The general strategy of this proof is to decompose certain homogeneous second order differential equation into a Riccati equation and a system of first order equations, bound the Riccati equation, and then integrate the result to get the bounds on the differential of the exponential. The view on Jacobi fields and parallel vector fields is from (Cheeger and Ebin 2008), while the approximation to Rauch's theorem using curvature equations is from (Petersen 2016). We provide different proofs of the Riccati comparison lemmas, as those in the book are incorrect. We emphasise throughout the exposition whether the theorems work just on the segment domain or up to the first conjugate point, as this point is often neglected in previous expositions and it is of interest in optimisation.

5.3.1 Jacobi fields

We begin by introducing what will be the main tool that we will just throughout the chapter: Jacobi fields. Jacobi fields are certain vector fields along a given geodesic that describe the behaviour of the differential of the exponential map along this geodesic. We will see that they are the solution of a certain differential equation, and we will use this equation to give the bounds on the differential of the exponential.

Before defining what Jacobi fields are and deducing this differential equation we will need a lemma. This lemma that can loosely be interpreted as *the Lie derivative commutes with the differential of a smooth map.*

Lemma 5.3.1. *For an immersion ψ and two vector fields X, Y on M there exist local extensions $\widetilde{X}, \widetilde{Y}$ along ψ—that is, $\mathrm{d}\psi(X_x) = \widetilde{X}_{\psi(x)}$ for $x \in M$—and*

$$\mathrm{d}\psi([X,Y]) = [\widetilde{X}, \widetilde{Y}].$$

Proof. See for example (O'Neill 1966, 1. Lemma 22 and 1. Lemma 33). □

With this lemma in hand, we are ready to show that the differential of the exponential satisfies certain differential equation.

Proposition 5.3.2 (First order Jacobi equation). *Let (M, g) be a complete Riemannian manifold. Let $\gamma\colon [0, r] \to M$ be a geodesic without conjugate points with initial unit vector $v \coloneqq \dot{\gamma}(0)$ and a vector $w \in T_pM$ such that $w \perp v$. Consider the following variation of γ in the direction of w*

$$\begin{aligned} c\colon [0,r] \times (-\varepsilon, \varepsilon) &\to M \\ (t, s) &\mapsto \exp_p(t(v + sw)) \end{aligned}$$

where ε is small enough so that $c(-,s)$ has no conjugate points for every $s \in (-\varepsilon,\varepsilon)$.

Define the vector field J along γ as

$$J(t) \coloneqq \mathrm{d}c(\partial_s)|_\gamma = \frac{\partial c}{\partial s}(t,0) = (\mathrm{d}\exp_p)_{tv}(tw).$$

This vector field solves the following differential equation on γ

$$\mathcal{L}_{\nabla r} J|_\gamma = 0$$

or equivalently, using the pullback connection

$$\nabla_{\dot\gamma} J = \nabla_J \nabla r. \tag{5.4}$$

Proof. The fact that γ is in U_p is equivalent to saying that $tv \in \mathrm{seg}^0(p)$ for $t \in [0,r]$. From this we get that the ε in the definition exists as $\mathrm{seg}^0(p)$ is open. Since c defines an embedded surface, c is a diffeomorphism onto its image.

Since ∂_t, ∂_s are coordinate vector fields of the surface defined by c, we have that

$$[\partial_t, \partial_s] = 0.$$

Using that c is a diffeomorphism, by Lemma 5.3.1 together with the fact that the Lie derivative restricted to a submanifold is the Lie derivative of the restrictions, we have that

$$[\mathrm{d}c(\partial_t), \mathrm{d}c(\partial_s)] = \mathrm{d}c([\partial_t, \partial_s]) = 0.$$

We get the Lie formulation of the Jacobi equation by noting that $\mathrm{d}c(\partial_t)|_\gamma = \nabla r|_\gamma$. The equation using the pullback connection is just a reformulation of the Lie derivative using that the Levi-Civita connection is torsion-free. □

Remark 5.3.3. Note that, even though the differential equation involves a Lie bracket, we just need to define J along the flow of ∇r, which is just a geodesic γ. This equation can be extended into a PDE on all of $U_p \backslash \{p\}$, looking for vector fields on this domain such that

$$\mathcal{L}_{\nabla r} W = 0.$$

One such a solution is clearly given by the radial vector field $W = \nabla r$.

To find other solutions, we imitate the construction that we did in Proposition 5.3.2 and define W on a sphere around p and extend it radially to all of U_p as

$$W(x) = \mathrm{d}\exp_p\Big(r(x) W\Big(\frac{x}{r(x)}\Big)\Big).$$

It is clear by Proposition 5.3.2 that this vector field solves the first order Jacobi equation on all of $U_p \backslash \{p\}$.

We will now use this equation to derive the linear version of the Jacobi equation.

Proposition 5.3.4 (Jacobi equation). *Let $\gamma\colon [0,r] \to M$ be a geodesic with initial values $\gamma(0) = p$, $\dot\gamma(0) = v$. Then $J(t) = (\mathrm{d}\exp_p)_{tv}(tw)$ is a vector field along γ that solves the following second order homogeneous linear differential equation*

$$\ddot J + R(J,\dot\gamma)\dot\gamma = 0.$$

*We call this equation the **Jacobi equation**.*

Proof. For a vector $w \in T_pM$, we may differentiate (5.4) to get

$$\nabla_{\partial_t} \nabla_{\partial_t} J = \nabla_{\partial_t} \nabla_J \nabla r$$

and using the definition of the curvature tensor and the fact that $[J, \nabla r] = 0$ we get

$$\nabla_{\partial_t} \nabla_{\partial_t} J + R(J, \nabla r)\nabla r = 0.$$

We then get the Jacobi equation by restricting this equation to γ, since $\nabla r \circ \gamma = \dot{\gamma}$.

Using an orthonormal parallel frame $\{e_i\}$ along γ with $e_1 = \dot{\gamma}$, and expressing a solution of this equation as $J^i e_i$, where $J^i \colon [0, r] \to \mathbb{R}$, we see that, in fact, this is a second order linear differential equation on $[0, r]$

$$\ddot{J}^i + R^i_j J^j = 0 \qquad i = 1, \dots, n$$

with coefficients

$$R^i_j = \begin{cases} 0 & \text{if } i = 1 \\ R(e_j, \dot{\gamma}, \dot{\gamma}, e_i) & \text{if } i = 2, \dots, n. \end{cases}$$

□

Definition 5.3.5. We say that a vector field J along a geodesic γ is a **Jacobi field** if it satisfies the Jacobi equation.

Remark 5.3.6 (Basic properties of Jacobi fields). By elementary theory of differential equations, the Jacobi equation has $2n$ independent solutions, defined by the initial values $(J(0), \dot{J}(0)) \in T_pM \times T_pM$. By construction, we have found n independent solutions given by

$$J(t) = (\mathrm{d}\exp_p)_{tv}(tw) \qquad \forall w \in T_pM.$$

These correspond to the initial values $J(0) = 0$, $\dot{J}(0) = w$. Geometrically, they correspond to a family of geodesics c that fixes the initial point, that is, $c(0, s) = p$.

It is also direct to show that $\dot{\gamma}$ is another solution to this equation with initial values $J(0) = v$, $\dot{J}(0) = 0$. The other $n - 1$ independent solutions correspond to variations of the geodesic γ that do not fix the initial point p. These solutions will not be important for our analysis.

It starts being clear now the importance of Jacobi fields. If we want to control the norm of the differential of $\exp_p$ at a point $rv \in \mathrm{seg}^0(p)$, for $r > 0$, $\|v\| = 1$, we may define the Jacobi field along the geodesic $\gamma \colon [0, r] \to M$ with initial conditions (p, v) and we have that

$$(\mathrm{d}\exp_p)_{rv}(w) = \frac{J(r)}{r}.$$

In order to bound the norm of the differential of the exponential, we just need to bound the norm of the solutions of the Jacobi equation.

If the vector w is parallel to v, as we saw in the proof of Proposition 5.3.4, the Jacobi field takes the form $J(t) = t\|w\|\dot{\gamma}(t)$. As such, in this direction we have an exact solution of the Jacobi equation, and we can compute its norm exactly as $\|J(t)\| = t\|w\|$.

If the vector w is perpendicular to $\dot{\gamma}(0)$, then $J(t)$ is perpendicular to $\dot{\gamma}(t)$ for every $t \in [0, r]$, as the equation for J^1 would be given by

$$\ddot{J}^1 = 0 \qquad (J^1(0), \dot{J}^1(0)) = (0, 0).$$

Geometrically, the previous proposition says that Jacobi fields just rotate around the vector field defined by $\dot\gamma$. In symbols, this means that if we split a Jacobi field along γ into its radial and normal part as

$$J^\dagger = \langle J, \dot\gamma\rangle\dot\gamma \qquad J^\perp = J - J^\dagger$$

if $\dot J^\dagger(0) = 0$ then $J^\dagger(t) = 0$ for every $t \in [0, r]$.

The results discussed in this remark are commonly known as the **Gauss's lemma**.

Jacobi fields are also very closely related to conjugate points and the conjugate locus. A point $q = \exp_p(v)$ is conjugate to p if the exponential at v is not full rank. Another way of looking at this definition is by defining a point q conjugate to p if there exists a Jacobi field connecting p and q such that it is zero at p and q. We will see when studying Rauch's theorem that the estimates for Jacobi fields will work on every point on the manifold but the conjugate locus, as these points will be exactly the singularities of the maps that we work with. For a unit vector v, we will denote by $r_{\text{conj}}(v) \in (0, \infty]$ the smallest time t at which the geodesic $\gamma(t) = \exp_p(tv)$ is conjugate to p, that is, the point $\gamma(r_{\text{conj}}(v))$ is conjugate to p.

5.3.2 Parallel vector fields

In the same way that Jacobi fields with initial condition $J(0) = 0$ and $\dot J(0) \perp \dot\gamma(0)$ are the vector fields such that $\mathcal{L}_{\nabla r} J = 0$, we have parallel vector fields.

Definition 5.3.7. We say that E is a **parallel vector field** on $U_p\backslash\{p\}$ if it satisfies

$$\nabla_{\nabla r} E = 0.$$

Parallel vector fields can be easily described along a geodesic as parallel transporting a vector $w \in T_{\gamma(0)}M$ along it. Analogously, we can also define a parallel vector field on all of $U_p\backslash\{p\}$ by specifying a vector field on a sphere and extending it to all of U_p parallel transporting this vector field along geodesics.

5.3.3 Rauch's theorem

We now go back to the study of the norm of the differential of the exponential. The strategy that we will follow will be that of bounding the derivative of the norm of the exponential. For that end, let J be a Jacobi field along a geodesic γ such that $J(0) = 0$, $\dot J(0) \perp \dot\gamma(0)$. As we already saw before, this vector field takes the form

$$J(t) = (\mathrm{d}\exp_p)_{t\dot\gamma(0)}(t\dot J(0)).$$

The derivative of its norm is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\|J\| = \frac{\langle \dot J, J\rangle}{\|J\|}.$$

At first sight it looks like we have not achieved much, as we still have a term involving the norm of J on the right-hand side, but it turns out that it can be conveniently rewritten using (5.4) as

$$\langle \dot J, J\rangle = \langle \nabla_{\nabla r} J, J\rangle = \langle \nabla_J \nabla r, J\rangle = \operatorname{Hess} r(J, J)$$

so we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\log\|J\| = \operatorname{Hess} r\Big(\frac{J}{\|J\|},\frac{J}{\|J\|}\Big).$$

Our plan will be to bound the Hessian of the distance to p on vectors of norm 1 to get bounds on the log-derivative of the norm of the Jacobi fields and then integrate these to get bounds on $\|J\|$. To this end, we start by computing formulas that relate the Hessian of the distance function to the curvature tensor.

A geometric interpretation of this approach comes after noting that the Hessian of the distance function is exactly the second fundamental form (or shape operator) of the distance function. As such, this quantity can be interpreted geometrically as the variation of the curvature of spheres.

Proposition 5.3.8 (Radial Curvature Equations). *Let (M,g) be a Riemannian manifold. For a point $p\in M$, we have on U_p*

$$(\mathcal{L}_{\nabla r}\operatorname{Hess} r)(X,Y) - \operatorname{Hess}^2 r(X,Y) = -R(X,\nabla r,\nabla r,Y) \tag{5.5}$$

$$(\nabla_{\nabla r}\operatorname{Hess} r)(X,Y) + \operatorname{Hess}^2 r(X,Y) = -R(X,\nabla r,\nabla r,Y). \tag{5.6}$$

where Hess^2 denotes the **iterated Hessian** defined in (5.3).

Proof. Since for a geodesic γ we have that $\nabla r|_\gamma = \dot\gamma$, the gradient of r has $\nabla_{\nabla r}\nabla r = 0$. From this we get

$$\begin{aligned}
-R(X,\nabla r,\nabla r,Y) &= \langle\nabla_{\nabla r}\nabla_X\nabla r,Y\rangle + \langle\nabla_{\mathcal{L}_X\nabla r}\nabla r,Y\rangle\\
&= D_{\nabla r}\langle\nabla_X\nabla r,Y\rangle - \langle\nabla_X\nabla r,\nabla_{\nabla r}Y\rangle + \operatorname{Hess} r(\mathcal{L}_X\nabla r,Y)\\
&= D_{\nabla r}(\operatorname{Hess} r(X,Y)) - \operatorname{Hess} r(X,\nabla_{\nabla r}Y) - \operatorname{Hess} r(\mathcal{L}_{\nabla r}X,Y).
\end{aligned}$$

Equation (5.5) follows after expanding the first term as

$$D_{\nabla r}(\operatorname{Hess} r(X,Y)) = (\mathcal{L}_{\nabla r}\operatorname{Hess} r)(X,Y) + \operatorname{Hess} r(\mathcal{L}_{\nabla r}X,Y) + \operatorname{Hess} r(X,\nabla_{\nabla r}Y - \nabla_Y\nabla r)$$

and equation (5.6) follows after expanding it as

$$D_{\nabla r}(\operatorname{Hess} r(X,Y)) = (\nabla_{\nabla r}\operatorname{Hess} r)(X,Y) + \operatorname{Hess} r(\nabla_{\nabla r}X,Y) + \operatorname{Hess} r(X,\nabla_{\nabla r}Y).$$ □

Remark 5.3.9 (A Riccati-type equation). Note that the previous proposition holds for every vector field Y. As such, we may write it in its $(1,1)$ form

$$(\nabla_{\nabla r}\operatorname{Hess} r)(X) + \operatorname{Hess}^2 r(X) + R(X,\nabla r)\nabla r = 0.$$

Denoting by $R_2(X) \coloneqq R(X,\nabla r)\nabla r$ the $(1,1)$ curvature tensor in this equation, and the shape operator for the geodesic balls as $S \coloneqq \operatorname{Hess} r$, this equation can be rewritten in the radial direction as a Riccati equation along gamma on self-adjoint $(1,1)$ tensors

$$\dot S + S^2 + R_2 = 0.$$

After a choice of a parallel frame along the geodesic, it can be regarded as a differential equation on self-adjoint matrices.

As one does in one dimension splitting a second order differential equation into a first order equation and a Riccati equation, here we have split the Jacobi equation into first order symmetric matrix equation—more formally, an equation on self-adjoint endomorphisms along a geodesic—and the first order equation defining Jacobi fields (Equation (5.4))

$$\dot{J} = S(J).$$

A careful analysis of this Riccati equation yields another particularly clean proof of Rauch's theorem, at the expense of the use of more abstract methods, as presented in (Eschenburg and Heintze 1990).

In contrast, we will use parallel and Jacobi vector fields to simplify these matrix equations.

Proposition 5.3.10. *Let γ be a geodesic, and let J be a Jacobi field along it such that $J(0) = 0$, $\dot{J}(0) \perp \dot{\gamma}(0)$. Let E be a parallel vector field along γ that is normal to $\dot{\gamma}$. We have the following differential equations along γ*

$$\frac{\mathrm{d}}{\mathrm{d}t}(\operatorname{Hess} r(J,J)) - \operatorname{Hess}^2 r(J,J) = -\sec(J,\dot{\gamma})\langle J, J\rangle \tag{5.7}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\operatorname{Hess} r(E,E)) + \operatorname{Hess}^2 r(E,E) = -\sec(E,\dot{\gamma}). \tag{5.8}$$

Proof. We just evaluate (5.5) and (5.6) on J, E and use that

$$\mathcal{L}_{\nabla r} J = 0 \qquad \nabla_{\nabla r} E = 0.$$

To go from the Riemannian tensor to the sectional curvature on the right-hand side, we just recall that if a Jacobi field or a parallel vector field is normal to $\dot{\gamma}$ at one point, it is normal to $\dot{\gamma}$ at every point. □

Given the signs of the Hess^2 in this proposition, it is now a bit clearer how Jacobi fields may be used to give lower bounds and parallel vector fields to give upper bounds on the Hessian of the distance function.

We now define some generalised trigonometric functions, which will be useful in the sequel.

Definition 5.3.11 (Generalised trigonometric functions). For a constant $\kappa \in \mathbb{R}$, we define the **generalised sine function** sn_κ as the solution to the differential equation

$$\ddot{x} + \kappa x = 0 \qquad x(0) = 0,\ \dot{x}(0) = 1.$$

In particular, we have

$$\operatorname{sn}_\kappa(t) \coloneqq \begin{cases} \frac{\sin(\sqrt{\kappa}t)}{\sqrt{\kappa}} & \text{if } \kappa > 0 \\ t & \text{if } \kappa = 0 \\ \frac{\sinh(\sqrt{-\kappa}t)}{\sqrt{-\kappa}} & \text{if } \kappa < 0 \end{cases}$$

Define π_κ as the first positive zero of sn_κ if it has one, and ∞ otherwise

$$\pi_\kappa \coloneqq \begin{cases} \frac{\pi}{\sqrt{\kappa}} & \text{if } \kappa > 0 \\ \infty & \text{if } \kappa \leq 0 \end{cases}$$

Finally, define the **generalised cotangent** ct_κ as

$$\operatorname{ct}_\kappa(t) \coloneqq \frac{\operatorname{sn}'_\kappa(t)}{\operatorname{sn}_\kappa(t)} = \begin{cases} \sqrt{\kappa}\cot(\sqrt{\kappa}t) & \text{if } \kappa > 0 \\ \frac{1}{t} & \text{if } \kappa = 0 \\ \sqrt{-\kappa}\coth(\sqrt{-\kappa}t) & \text{if } \kappa < 0 \end{cases}$$

This function is differentiable on $(0, \pi_\kappa)$.

Remark 5.3.12 (Riccati equation). Note that the equation $\ddot{x} + \kappa x = 0$ is the associated second order equation to the Riccati equation $\dot{\rho} + \rho^2 + \kappa = 0$. The relation between these two equations comes from the change of variables $\rho = \frac{\dot{x}}{x}$. This is exactly the Riccati equation that the shape operator for the distance function solves in constant curvature, as pointed out before. Given that sn_κ solves the second order equation, ct_κ solves the Riccati equation.

Remark 5.3.13 (Qualitative behaviour). From a qualitative point of view, it is worth noting that the family of functions $\mathrm{sn}_\kappa(t)$ is strictly increasing in κ. We also have that sn_κ has a zero at 0 for every κ and another one at π_κ for $\kappa > 0$, so

$$\lim_{t \to 0^+} \mathrm{ct}_\kappa(t) = \infty \qquad \kappa \in \mathbb{R}$$
$$\lim_{t \to \pi_\kappa^-} \mathrm{ct}_\kappa(t) = -\infty \qquad \kappa > 0.$$

All we need to do now is to relate the solutions of the Jacobi equation in Riccati as given by the Hessian of the distance function in Proposition 5.3.10 with its solutions in constant curvature. To do so, we need the following elementary comparison lemma for functions of real variable. It can be thought of as a version of Sturm's comparison theorem for Riccati equations, and it is proved in a similar way.

Lemma 5.3.14 (Comparison Lemma for Riccati Equations). *Let $\rho_1, \rho_2 \colon (a, b) \to \mathbb{R}$ be two differentiable functions such that*

$$\dot{\rho}_1 + \rho_1^2 \leq \dot{\rho}_2 + \rho_2^2.$$

If we have that $\rho_1(t_0) \geq \rho_2(t_0)$ for a $t_0 \in (a, b)$, then $\rho_1 \geq \rho_2$ on $(a, t_0]$.

Proof. Let $u = \rho_1 - \rho_2$ and $\zeta = \dot{\rho}_2 - \dot{\rho}_1 + \rho_2^2 - \rho_1^2$. We have by hypothesis that $u(t_0) \geq 0$ and $\zeta \geq 0$ on (a, b), and it is enough to prove that $u \geq 0$ on $(a, t_0]$ provided that $u(t_0) \geq 0$.

We can write the differential inequality as a differential equation in terms of u and ζ as

$$\dot{u} = -(\rho_1 + \rho_2)u - \zeta.$$

and we can solve this differential equation using the method of variation of parameters. We compute the general solution of the homogeneous system

$$\dot{v} = -(\rho_1 + \rho_2)v$$

as $v = Cv_1$ for $v_1 := e^{-\int \rho_1 + \rho_2} > 0$ and a constant $C \in \mathbb{R}$.

To get a particular solution of the inhomogeneous equation, we may use variation of parameters. Let $u = \eta v_1$ and, plugging it into the inhomogeneous equation, we see that the function η satisfies $\dot{\eta} = -\zeta v_1^{-1}$. As ζ and v_1^{-1} are positive, we conclude that η is decreasing.

Finally, we get the general solution by adding the particular solution and the general solution to the homogeneous system $u = v_1(C + \eta)$. Since, by hypothesis, $u(t_0) \geq 0$, we get that $C + \eta(t) \geq 0$ for $t \in (a, t_0]$ so that $u(t) \geq 0$ on $(a, t_0]$. □

Proposition 5.3.15 (Riccati Comparison Estimate). *Let $\rho \colon (0, b) \to \mathbb{R}$ be a differentiable function and fix $\kappa \in \mathbb{R}$.*

1. *If* $\dot\rho + \rho^2 \le -\kappa$ *then* $\rho(t) \le \mathrm{ct}_\kappa(t)$ *on* $(0, b)$. *Furthermore,* $b \le \pi_\kappa$

2. *If* $\dot\rho + \rho^2 \ge -\kappa$ *and* $\lim_{t\to 0^+} \rho(t) = \infty$ *then* $\mathrm{ct}_\kappa(t) \le \rho(t)$ *on* $(0, \min\{b, \pi_\kappa\})$.

Proof. For the first inequality, assume by contradiction that $\rho(t_0) > \mathrm{ct}_\kappa(t_0)$ for some $t_0 \in (0, b)$. By continuity, we can choose $0 < \varepsilon < t_0$ such that

$$\rho(t_0) \ge \mathrm{ct}_\kappa(t_0 - \varepsilon).$$

By Lemma 5.3.14, since $\mathrm{ct}_\kappa(t - \varepsilon)$ solves the differential equation $\dot x + x^2 + \kappa = 0$, we have that

$$\rho(t) \ge \mathrm{ct}_\kappa(t - \varepsilon) \qquad \forall t \in (\varepsilon, t_0).$$

But taking limits as t tends to ε on the right, the right-hand side goes to infinity, which is a contradiction. In this case, we also have that $b \le \pi_\kappa$ for ρ to be differentiable, as $\lim_{t\to\pi_\kappa^-} \mathrm{ct}_\kappa(t) = -\infty$ for $\kappa > 0$.

We get the second inequality after interchanging the roles of ρ and ct_κ in the previous argument. In this case, we need $t \in (0, \min(b, \pi_\kappa))$ for $\mathrm{ct}_\kappa(t)$ to be differentiable as $\lim_{t\to\pi_\kappa^-} \mathrm{ct}_\kappa(t) = -\infty$ for $\kappa > 0$. □

We now have everything we need to prove Rauch's theorem.

Theorem 5.3.16 (Rauch's theorem). *Let* (M, g) *be a Riemannian manifold with bounded sectional curvature* $\delta \le \sec \le \Delta$. *Fix a point* $p \in M$ *and define* r *as the distance to* p. *For any other point in a ball* $x \in B_p(\pi_\Delta)\backslash\{p\}$, *and any* $w \in T_xM$ *we have*

$$\mathrm{ct}_\Delta(r(x))\|w^\perp\| \le (\mathrm{Hess}\, r)_x(w, w) \le \mathrm{ct}_\delta(r(x))\|w^\perp\|.$$

The upper bound also holds for every $x \in M\backslash(\mathrm{cut}(()p) \cup \{p\})$. *These bounds are tight on spaces of constant curvature.*

Proof. Since $x \notin \mathrm{cut}(p)$, there exists a unique distance minimising geodesic between x and p. Let $\gamma\colon [0, r] \to M$ be that geodesic.

If $w \in T_xM$ is a radial vector, assume that $w = \dot\gamma(r)$—the case $w = -\dot\gamma(r)$ follows by linearity. We then have that, by definition of the Hessian, since $r|_\gamma = \dot\gamma$,

$$\mathrm{Hess}\, r(\dot\gamma, \dot\gamma) = \langle \nabla_{\dot\gamma} r, \dot\gamma\rangle = \langle \nabla_{\dot\gamma}\dot\gamma, \dot\gamma\rangle = 0,$$

so the Hessian of the distance function is zero in the radial direction.

For the rest of the bounds, we will take advantage of the simple form that the curvature equations take when evaluated on Jacobi and parallel vector fields, as computed in Proposition 5.3.10.

For the lower bound we consider a Jacobi field along γ such that $J(r) = w$ and define $\rho(t) = \mathrm{Hess}\, r(\frac{J}{\|J\|}, \frac{J}{\|J\|})$. Computing its derivative

$$\dot\rho = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{Hess}\, r(J, J)}{\langle J, J\rangle} = \frac{\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Hess}\, r(J, J)\|J\|^2 - 2\,\mathrm{Hess}\, r(J, J)^2}{\|J\|^4} = \frac{\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Hess}\, r(J, J)}{\|J\|^2} - 2\rho^2.$$

and using the formula (5.7) for the derivative of the Hessian on a Jacobi vector field, we get

$$\dot\rho + 2\rho^2 - \mathrm{Hess}^2\, r\Big(\frac{J}{\|J\|}, \frac{J}{\|J\|}\Big) = -\sec(J, \dot\gamma).$$

Using Cauchy–Schwarz, we can bound the iterated Hessian for any vector field X of norm 1

$$\operatorname{Hess}^2 r(X,X) = \langle \nabla_X \nabla r, \nabla_X \nabla r\rangle \geq \langle \nabla_X \nabla r, X\rangle^2 = \operatorname{Hess} r(X,X)^2$$

so taking $X = \frac{J}{\|J\|}$ together with the upper bound on the sectional curvature we get the differential inequality

$$\dot\rho + \rho^2 \geq -\Delta.$$

Finally, since $\|J(0)\| = 0$ and $\|\dot J(0)\| \neq 0$, we have that $\lim_{t\to 0^+} \rho(t) = \infty$, so we may use the second part of Proposition 5.3.15 to finish the lower bound.

For the upper bound consider a parallel vector field E such that $E(r) = w$ and define $\rho = \operatorname{Hess} r(E,E)$. We may then rewrite Equation (5.8) for the derivative of $\operatorname{Hess} r$ on a parallel vector field as

$$\dot\rho + \rho^2 = -\sec(E,\dot\gamma) \leq -\delta$$

and we finish by applying the first part of Proposition 5.3.15. □

Remark 5.3.17 (Bounds on Jacobi fields). For the upper bound, we never used the fact that $x \notin \operatorname{cut}(p)$. We merely used that the geodesic γ does not have any conjugate points on $[0,r]$, to be able to use Proposition 5.3.15, as if $\|J(t)\| = 0$, then the Hessian of the distance function at that point would be infinite.

As stated, the theorem is as general as it can be, as the distance function r is not differentiable on $\operatorname{cut}(p)$. On the other hand, if it is stated in terms of Jacobi fields, one can further generalise it to Jacobi fields along geodesics without conjugate points, since both the proof of the theorem and that of Proposition 5.3.8 can be done in terms of a geodesic and short variations of geodesics. We will use this more general version in the following theorem.

Theorem 5.3.18 (First order bounds for the exponential map). *Let (M,g) be a Riemannian manifold with bounded sectional curvature $\delta \leq \sec \leq \Delta$. Fix a point $p \in M$, for any unit vector $v \in T_pM$ and $r \in [0, \pi_\Delta]$,*

$$\min\Big\{1, \frac{\operatorname{sn}_\Delta(r)}{r}\Big\}\|w\| \leq \|(\mathrm{d}\exp_p)_{rv}(w)\| \leq \max\Big\{1, \frac{\operatorname{sn}_\delta(r)}{r}\Big\}\|w\| \qquad \forall w \in T_pM.$$

The upper bound also holds for $r < r_{\mathrm{conj}}(v)$.

These bounds are tight on spaces of constant curvature.

Proof. The bound by 1 above and below comes from the Gauss's lemma, as the exponential is a radial isometry. From this, and using linearity, we just have to prove the bound for a vector w in the normal direction with $\|w\| = 1$.

All we have to do now is to transform the bounds on the Riccati equation on bounds on the Jacobi equation. Define γ as the geodesic starting at p with $\dot\gamma(0) = \frac{v}{\|v\|}$, and choose a Jacobi field along γ such that $J(r) = w$, that is, choose $\dot J(0) = \frac{w}{r}$. As we noted at the beginning of this section, we have that

$$\operatorname{Hess} r\Big(\frac{J}{\|J\|}, \frac{J}{\|J\|}\Big) = \frac{\langle \dot J, J\rangle}{\|J\|^2} = \frac{\frac{\mathrm{d}}{\mathrm{d}t}\|J\|}{\|J\|} = \frac{\mathrm{d}}{\mathrm{d}t}\log(\|J\|)$$

so Rauch's theorem may be rewritten on $(0, \pi_\Delta)$ as

$$\frac{\mathrm{d}}{\mathrm{d}t}\log(\mathrm{sn}_\Delta(t)) \leq \frac{\mathrm{d}}{\mathrm{d}t}\log(\|J\|) \leq \frac{\mathrm{d}}{\mathrm{d}t}\log(\mathrm{sn}_\delta(t)).$$

Integrating, using that $\|J(0)\| = \mathrm{sn}_\Delta(0) = \mathrm{sn}_\delta(0) = 0$, and computing the limit using l'Hôpital, we get

$$\mathrm{sn}_\Delta(t) \leq \|J\| \leq \mathrm{sn}_\delta(t).$$

and since $J(r) = (\mathrm{d}\exp_p)_{rv}(rw)$,

$$\|(\mathrm{d}\exp_p)_{rv}(w)\| = \frac{\|J(r)\|}{r}.$$

As it was the case in the proof of Rauch's theorem, we just used that γ does not have conjugate points on $[0, r]$, rather than the stronger $rv \in \mathrm{seg}^0(p)$. As such, just by noting that the proof does not rely on the definition of $\mathrm{Hess}\, r$, and instead, can be carried out in terms of Jacobi fields, we get the finer result for the upper bound. □

Example 5.3.19. To see how the definition of the upper bounds are an improvement over just considering $rv \in \mathrm{seg}^0(p)$, consider the flat torus ($\mathbb{S}^1 \times \mathbb{S}^1$ with the product metric). For this manifold, $\mathrm{seg}^0(p) \subseteq T_pM$ is a square centred at 0. On the other hand, since for the flat torus we have that $\mathrm{conj}(p) = \emptyset$, we get that the upper bound holds on all T_pM.

We make explicit the following strengthening of the Cartan–Hadamard theorem that was implicitly present in the statement of the last theorem.

Theorem 5.3.20. *If (M, g) has $\sec \leq \Delta$, then the exponential map $\exp_p$ is not singular on $B_p(\pi_\Delta)$, so $B_p(\pi_\Delta) \subseteq M \setminus \mathrm{conj}(p)$ and $\pi_\Delta \leq r_{\mathrm{conj}}(v)$ for every point $p \in M$ and every unit vector $v \in T_pM$.*

5.3.4 The law of cosines on manifolds

Rauch's theorem has found countless applications in geometry, as those given by Rauch himself, who proved with it a weak version of the sphere theorem, or as used by Gromov or Karcher to give bounds the volume form of a Riemannian manifold, or to control the size of the fundamental group of a given manifold. Here we deviate slightly from the presentation to remark one that has been found particularly useful in the context of optimisation on manifolds.

Theorem 5.3.21 (Law of cosines). *Let (M, g) be a Riemannian manifold with bounded sectional curvature $\delta \leq \sec \leq \Delta$. Let p be a point on M and let $x, y \in B_p(R)$ for $R \leq \pi_\Delta$ such that the minimising length geodesic γ that connects them also lies in this ball. Define the angle $\alpha = \angle pxy$. We then have*

$$d(y,p)^2 \leq \zeta_{1,\delta}(R)d(x,y)^2 + d(x,p)^2 - 2d(x,y)d(x,p)\cos(\alpha)$$
$$d(y,p)^2 \geq \zeta_{2,\Delta}(R)d(x,y)^2 + d(x,p)^2 - 2d(x,y)d(x,p)\cos(\alpha)$$

where

$$\zeta_{1,\kappa}(r) \coloneqq \max\{1, r\,\mathrm{ct}_\kappa(r)\} = \begin{cases} 1 & \text{if } \kappa \geq 0 \\ \sqrt{-\kappa} r \coth(\sqrt{-\kappa} r) & \text{if } \kappa < 0 \end{cases}$$
$$\zeta_{2,\kappa}(r) \coloneqq \min\{1, r\,\mathrm{ct}_\kappa(r)\} = \begin{cases} 1 & \text{if } \kappa \leq 0 \\ \sqrt{\kappa} r \cot(\sqrt{\kappa} r) & \text{if } \kappa > 0 \end{cases}$$

Furthermore, the first bound holds for any $x, y \in M$ such that there exists a distance minimising geodesic contained in U_p that connects them. These bounds are tight on spaces of constant curvature.

Proof. Define $f_p = \frac{1}{2}r^2$. We start by relating the Hessian of f_p to the Hessian of r using the Leibnitz rule twice

$$\operatorname{Hess} f_p = \nabla(r\mathrm{d}r) = \mathrm{d}r \otimes \mathrm{d}r + r \operatorname{Hess} r.$$

As such, for a radial vector w of norm 1, since $\operatorname{Hess} r(w, w) = 0$, this equation simplifies to

$$\operatorname{Hess} f_p(w, w) = \mathrm{d}r(w)^2 = \langle \nabla r, w\rangle^2 = 1.$$

For normal vectors we may use the bounds on the Hessian of r. It is now evident that the functions $\zeta_{1,\delta}$ and $\zeta_{2,\Delta}$ are defined to be upper and lower bound on the Hessian of f_p acting on vectors of norm 1.

Let $\gamma\colon [0, t] \to M$ be a distance-minimising geodesic such that $\gamma(0) = x$, $\gamma(t) = y$ and $\gamma(s) \in U_p$ for every $s \in [0, t]$. We will prove the first inequality, as the proof of the second one is analogous. By the bounds above, using (5.2), we have that

$$\operatorname{Hess} f_p(\dot\gamma(s), \dot\gamma(s)) = (f_p \circ \gamma)''(s) \leq \zeta_{1,\delta}(R)$$

for every $s \in [0, t]$, since f_p is differentiable on U_p. Integrating this inequality we get

$$\int_0^t (f_p \circ \gamma)''(s)\,\mathrm{d}s = (f_p \circ \gamma)'(t) - \langle \nabla f_p(\gamma(0)), \dot\gamma(0)\rangle \leq \zeta_{1,\delta}(R)t.$$

Integrating once again we have that

$$(f_p \circ \gamma)(t) - (f_p \circ \gamma)(0) - \langle \nabla f_p(\gamma(0)), \dot\gamma(0)\rangle t \leq \zeta_{1,\delta}(R)\frac{t^2}{2}.$$

which, after substituting the values of f_p and γ, gives

$$d(y, p)^2 \leq \zeta_{1,\delta}(R)d(x, y)^2 + d(x, p)^2 + 2d(x, y)\langle \nabla f_p(x), \dot\gamma(0)\rangle.$$

Finally, since $\nabla f_p(x) = r(x)\nabla r(x) = -\exp_x^{-1}(p)$ we get the desired inequality. □

Remark 5.3.22 (Obstructions to a global law of cosines). One could ask whether it is possible to extend this local result on U_p to a global result for geodesic triangles whose sides are distance minimising, as one does in Toponogov's theorem. This is, in general, not possible. Consider the flat torus and a point p in it and an equilateral triangle on a generating circle. The sides of this triangle are distance minimising of length x—one third the circumference of the circle—and the angles of this triangle will be $\alpha = \pi$. Since for the flat torus we have that $\delta = 0$, the first inequality would imply that $3x^2 \leq 2x^2$.

The upper bound for the case $\delta \leq 0$ has found multiples uses in the optimisation literature in the context of Hadamard manifolds (*cf.*, Section 4.4.2.1). Rauch's theorem allows for a particularly clean proof of Cartan–Hadamard theorem, showing that these manifolds are diffeomorphic to $\mathbb{R}^n$ through the Riemannian exponential map at any given point.

Corollary 5.3.23 (Cartan–Hadamard theorem). *A simply connected Hadamard manifold is diffeomorphic to $\mathbb{R}^n$.*

Proof. By Theorem 5.3.20, since $\sec \leq 0$, $\pi_\Delta = \infty$ and the differential exponential map at a point $p \in M$ is always full rank. As such, by the inverse function theorem, it is a covering map, and $\mathbb{R}^n$ is the universal cover of M. Since M is simply connected, M is diffeomorphic to its universal cover. □

It is now clear that simply connected Hadamard manifolds are a class of particularly well-behaved manifolds, in that the $\operatorname{cut}(p) = \emptyset$ for every $p \in M$, so $U_p = M$ and $\operatorname{seg}^0(p) = T_pM$, and all the theorems in Section 5.3 hold globally (*i.e.*, for every $v \in T_pM$).

If one then considers a simply connected Hadamard manifold with sectional curvature bounded below by $\delta < 0$, then one may apply the upper bounds in Theorem 5.3.21 on the whole manifold. This idea was heavily exploited in (Bonnabel 2013) to get convergence rates for stochastic gradient descent, and in (H. Zhang, Reddi, and Sra 2016) to prove convergence both stochastic and non-stochastic setting. In (Ferreira, Louzeiro, and Prudente 2019), the authors used the lower bounds whenever $\delta < 0$ to prove convergence rates for certain step-size schedulers. In (Bento, Ferreira, and Melo 2017), the authors use the law of cosines on manifolds of either non-negative or non-positive curvature to prove convergence rates for subgradient methods and proximal-point methods. In (Agarwal et al. 2020), the lower bounds are used to prove convergence rates for certain second order method.

In the previous presentation, we have showed that the Hadamard restriction, that is, being simply connected and with sectional curvature bounded above, is not really necessary. In particular, if we have a positive bound on the curvature, we can still apply this kind of results, at the expense of working on certain neighbourhood of the initial point. The same happens if we remove the condition of the manifold being simply connected. In this case, the first-order bounds on the exponential still hold globally, but the law of cosines only works on a ball of radius one half the length of the shortest geodesic loop.

5.4 Second Order Bounds for the Exponential Map

In this section, we give bounds on the Hessian of the differential of the exponential map. This kind of bounds were first given in the paper (Kaul 1976). The proof can just be found in German, so we will give a self-contained presentation. Our proof uses a comparison lemma developed by Kaul, and then streamlines all the other technical tools, considerably simplifying the proof and obtaining tighter explicit bounds.

5.4.1 The differential equation

We start by giving a technical remark on the nature of the Hessian of the exponential map. The differential of the exponential for a point $(p, v) \in TM$ is a linear map of the form

$$(\mathrm{d}\exp_p)_v \colon T_pM \to T_{\exp_p(v)}M.$$

For this reason, it can be seen as a section of the bundle $T_p^*M \otimes \exp_p^*(TM)$ over T_pM. We have connections ∇^{flat} and ∇ on T_p^*M and $\exp_p^*(TM)$ respectively. Any two connections induce a connection on a tensor product bundle so that the Leibnitz rule holds. In the concrete example of the exponential map, this Leibnitz rule for vector fields $\overline{W}_1, \overline{W}_2$ on $\operatorname{seg}^0(p)$, takes the form

$$\nabla_{\mathrm{d}\exp_p(\overline{W}_1)}(\mathrm{d}\exp_p(\overline{W}_2)) = (\nabla_{\overline{W}_1}\mathrm{d}\exp_p)(\overline{W}_2) + \mathrm{d}\exp_p(\nabla^{\text{flat}}_{\overline{W}_1}\overline{W}_2).$$

Since $\exp_p$ is a diffeomorphism between $\mathrm{seg}^0(p)$ and U_p, writing $W_i = \mathrm{d}\exp_p(\overline{W}_i)$ for the pushforward of a vector from $\mathrm{seg}^0(p)$ to U_p, we see that the Hessian of the exponential map can be put in terms of the Christoffel symbols in normal coordinates as

$$(\nabla_{\overline{W}_1}\mathrm{d}\exp_p)(\overline{W}_2) = \nabla_{W_1}W_2 - \nabla^{\mathrm{flat}}_{\overline{W}_1}\overline{W}_2 = \Gamma(W_1, W_2) = \Gamma^k_{i,j}W_1^iW_2^j\partial_k. \tag{5.9}$$

In other words, the Hessian of the exponential map is exactly the Christoffel symbols of the connection in normal coordinates at p evaluated at the pushforward of the vector along $\exp_p$. In particular, this shows that the Hessian of the exponential map is a symmetric bilinear map of the form

$$(\nabla\mathrm{d}\exp_p)_v \colon T_pM \times T_pM \to T_{\exp_p(v)}M.$$

As we did for the first order bounds, our strategy to bound this quantity will be to deduce a differential equation for $\nabla\mathrm{d}\exp_p$ and then bound the norm of the solutions of this equation. In this case, we will have to differentiate a second time through a second variation of a geodesic. Given that this second derivative will not be in the direction of the geodesic, it will not be enough to have vector fields along the geodesic—we will have to have them defined also in the direction in which we want to differentiate them.

Proposition 5.4.1. *Let (M, g) be a Riemannian manifold and let γ be a geodesic with initial unit vector $v \in T_pM$. For two vectors $w_1, w_2 \in T_pM$ perpendicular to v, the vector field along γ*

$$K(t) \coloneqq (\nabla\mathrm{d}\exp_p)_{tv}(tw_1, tw_2)$$

satisfies the following second order inhomogeneous linear differential equation along γ

$$\ddot{K} + R(K, \dot\gamma)\dot\gamma + Y = 0 \qquad K(0) = 0, \dot K(0) = 0$$

where Y is the vector field along γ given by

$$Y \coloneqq 2R(J_1, \dot\gamma)\dot J_2 + 2R(J_2, \dot\gamma)\dot J_1 + (\nabla_{\dot\gamma}R)(J_2, \dot\gamma)J_1 + (\nabla_{J_2}R)(J_1, \dot\gamma)\dot\gamma$$

and J_1, J_2 are the Jacobi fields along γ with initial conditions $J_i(0) = 0$, $\dot J_i(0) = w_i$.

Proof. Define the geodesic variation

$$\begin{aligned} c\colon [0, r] \times (-\varepsilon, \varepsilon) \times (-\varepsilon, \varepsilon) &\to M \\ (t, s_1, s_2) &\mapsto \exp_p(t(v + s_1w_1 + s_2w_2)) \end{aligned}$$

Consider the first variation along $\gamma(t) = \exp_p(tv)$ in the direction of s_1 given by the family of Jacobi fields

$$\tilde J_1(t, s) = \left.\frac{\partial c}{\partial s_1}\right|_{s_1=0}(t, s) = (\mathrm{d}\exp_p)_{t(v+sw_2)}(tw_1).$$

We denote by $J_i(t)$ the Jacobi field along γ with initial conditions $(0, w_i)$. In particular, we have that $J_1(t) = \tilde J_1(t, 0)$, so $\tilde J_1$ is an extension of J_1 in the direction of J_2 so that $\nabla_{J_2}\tilde J_1$ is well-defined. Moreover, we have that $\nabla^{\mathrm{flat}}_{J_2}\tilde J_1 = 0$ as $\tilde J_1$ is constant in the direction of J_2, that is, in the direction of stw_2 for $s \in (-\varepsilon, \varepsilon)$ and a fixed t. For this reason, the vector field K is the second variation along γ in the directions w_1, w_2

$$K(t) = (\nabla\mathrm{d}\exp_p)_{tv}(tw_1, tw_2) = \nabla_{J_2}\tilde J_1 = \left.\frac{\partial \tilde J_1}{\partial s}\right|_{s=0}(t).$$

Furthermore, $\tilde{J}_1$ satisfies the Jacobi equation in all its domain with initial conditions $(0, \frac{1}{\|v+sw_1\|})$ so that

$$\nabla_{\nabla r}\nabla_{\nabla r}\tilde{J}_1 + R(\tilde{J}_1, \nabla r)\nabla r = 0.$$

We may then derive a differential equation for K along γ by differentiating this equation in the direction of J_2

$$\nabla_{J_2}\nabla_{\nabla r}\nabla_{\nabla r}\tilde{J}_1 + \nabla_{J_2}(R(\tilde{J}_1, \nabla r)\nabla r) = 0. \tag{5.10}$$

Note that, as we just are interested in deriving a differential equation for K along γ, we will use that $\nabla r|_\gamma = \dot{\gamma}$ and $\tilde{J}_1 = J_1$ whenever we have that a quantity just depends on the values of the vector fields involved along γ, rather than in a neighbourhood around γ in the direction of J_2.

We may put the first term of (5.10) in terms of K by repeatedly using the definition of the curvature tensor and the fact that since $[J_2, \nabla r]|_\gamma = 0$, we have that $\nabla_{J_2}\nabla r = \nabla_{\dot{\gamma}} J_2 = \dot{J}_2$

$$\begin{aligned}
\nabla_{J_2}\nabla_{\nabla r}\nabla_{\nabla r}\tilde{J}_1 &= \nabla_{\dot{\gamma}}\nabla_{J_2}\nabla_{\nabla r}\tilde{J}_1 + R(J_2, \dot{\gamma})\nabla_{\dot{\gamma}} J_1 \\
&= \nabla_{\dot{\gamma}}\nabla_{\dot{\gamma}} K + \nabla_{\dot{\gamma}}(R(J_2, \dot{\gamma})J_1) + R(J_2, \dot{\gamma})\dot{J}_1 \\
&= \nabla_{\dot{\gamma}}\nabla_{\dot{\gamma}} K + (\nabla_{\dot{\gamma}} R)(J_2, \dot{\gamma})J_1 + R(\dot{J}_2, \dot{\gamma})J_1 + 2R(J_2, \dot{\gamma})\dot{J}_1.
\end{aligned}$$

We can expand the second term as

$$\nabla_{J_2}(R(\tilde{J}_1, \nabla r)\nabla r) = (\nabla_{J_2} R)(J_1, \dot{\gamma})\dot{\gamma} + R(K, \dot{\gamma})\dot{\gamma} + R(J_1, \dot{J}_2)\dot{\gamma} + R(J_1, \dot{\gamma})\dot{J}_2.$$

Putting everything together and using the symmetries of the curvature tensor and the first Bianchi identity we get the differential equation.

For the initial conditions, we have that

$$K(0) = (\nabla \mathrm{d}\exp_p)_0(0, 0) = 0.$$

Differentiating the Hessian in the direction of $\dot{\gamma} = \frac{\mathrm{d}}{\mathrm{d}t}$ we have that

$$\dot{K}(0) = (\nabla\nabla \mathrm{d}\exp_p)_0(v, 0, 0) + (\nabla \mathrm{d}\exp_p)_0(w_1, 0) + (\nabla \mathrm{d}\exp_p)_0(0, w_2) = 0. \qquad \square$$

5.4.2 Manifolds of bounded geometry

By Proposition 5.4.1, we have that the Hessian of the exponential map solves a Jacobi-like differential equation with two main differences: It is inhomogeneous and depends on the covariant derivative of the curvature tensor.

The inhomogeneity of the differential equation will stop us from simplifying the differential equation into two first order differential equations, as we did for the Jacobi equation. In contrast, this time we will have to deal directly with the second order equation. Luckily, we have already developed most of the tools to do so.

The fact that involves the covariant derivative of the curvature tensor is a more intrinsic difference. Since the sectional curvature completely defines the curvature tensor, bounds on the sectional curvature can be translated into bounds on the norm of the curvature tensor. One question that naturally arises is whether these 0-th order bounds also give first order bounds. As one may expect, this is not the case.

Example 5.4.2 (Manifold of bounded 0-th order and unbounded 1-st order). Consider a rotationally symmetric surface of the form $\mathrm{d}r^2 + \rho(r)^2\mathrm{d}\theta^2$ on the cylinder $(0,1)\times\mathbb{S}^1$ for a function $\rho > 0$. These metrics have a particularly simple formula for the sectional curvature

$$\sec(\partial_r, \partial_\theta) = \frac{R(\partial_r, \partial_\theta, \partial_\theta, \partial_r)}{\rho^2} = -\frac{\ddot{\rho}}{\rho}.$$

All we have to do now is to choose a positive function ρ with bounded second derivative and arbitrarily large third derivative, for example $\rho(r) = 2 + r^5\sin(1/r)$. This metric has $|\sec| < 12$, but

$$(\nabla_{\partial_r} R)(\partial_r, \partial_\theta, \partial_\theta, \partial_r) = D_{\partial_r}(R(\partial_r, \partial_\theta, \partial_\theta, \partial_r)) - 2R(\partial_r, \nabla_{\partial_r}\partial_\theta, \partial_\theta, \partial_r) = \dot{\rho}\ddot{\rho} - \rho\dddot{\rho}$$

which is unbounded as r goes down to 0.

This example motivates the following definition.

Definition 5.4.3 (Manifold of bounded geometry). Let (M, g) be a Riemannian manifold. We say that a manifold has k**-bounded geometry** if for every $i = 0, \dots, k$ there exist constants $C_i \geq 0$ such that

$$\|\nabla^i R\| \leq C_i$$

and the injectivity radius is uniformly bounded below by a positive constant

$$\mathrm{r}_{\mathrm{inj}} \coloneqq \inf_{p\in M} \mathrm{r}_{\mathrm{inj}}(p) > 0.$$

Remark 5.4.4. A manifold has 0-bounded geometry if it has bounded sectional curvature above and below and positive injectivity radius. As one may expect, given that the 1-bounded geometry is given by the Christoffel symbols, which are given by directional derivatives of the metric g in normal coordinates, the condition on the boundedness of the derivatives of the curvature tensor is equivalent to asking for the entries of the metric tensor to be bounded in normal coordinates. This was first proved in (Eichhorn 1991).

It is clear that any compact manifold with any given metric will be of k-bounded geometry for every $k \in \mathbb{N}$. A natural question would be whether this condition puts any constraint in the topology of the manifold. This is not the case.

Theorem 5.4.5 (Greene 1978). *Every differentiable manifold admits a complete metric of bounded geometry.*

Given that we usually have access to upper and lower bounds on the sectional curvature, and given the form that takes the covariant derivatives in Proposition 5.4.1, we give the following definition, which is just a generalisation of that given in the introduction.

Definition 5.4.6. We say that a Riemannian manifold (M, g) has $(\delta, \Delta, \Lambda)$**-bounded geometry** if $\mathrm{r}_{\mathrm{inj}} > 0$ and

$$\delta \leq \sec \leq \Delta$$
$$\|(\nabla_x R)(y,x)y + (\nabla_y R)(y,x)x\| \leq 2\Lambda\|x\|^2\|y\|^2 \qquad \forall p \in M \quad \forall x, y \in T_pM.$$

Remark 5.4.7. If a manifold is of $(\delta, \Delta, \Lambda)$-bounded geometry according to Definition 5.1.1 it is of $(\delta, \Delta, \Lambda)$-bounded geometry according to Definition 5.4.6, but not necessarily the other way around. In the proofs, we will just use Definition 5.4.6, as it is all we need. This definition also simplifies the computations when one wants to estimate the actual constant Λ for a concrete manifold.

Remark 5.4.8 (Injectivity radius). We will not use the fact that these manifolds have uniformly bounded injectivity radius in this chapter, but this fact turns out to be crucial when proving other flavour of theorems in optimisation of manifolds that involve $\exp_p^{-1}$ or parallel transport along geodesics or a retraction.

5.4.3 Curvature bounds

We start by recalling the following lemma that gives a closed formula for the curvature tensor of manifolds of constant sectional curvature. In its more general form, it says that the Riemannian curvature tensor is completely specified by the sectional curvature.

Lemma 5.4.9 (Riemann, 1854). *Let (M, g) be a Riemannian manifold of constant sectional curvature $\kappa \in \mathbb{R}^n$. The curvature tensor takes the form*

$$R(x, y)z \coloneqq R_\kappa(x, y)z = \kappa(\langle z, y\rangle x - \langle z, x\rangle y) \qquad \forall x, y, z \in T_pM.$$

To be able to bound the terms in the differential equation, we will need some estimates on the norm of the curvature tensor in terms of the sectional curvature. We will use 4 inequalities that can be regarded as a generalisation of Berger's lemma (Berger 1960). The first three inequalities are announced—the last one with a worse constant—in (Kaul 1976) citing for their proof a monograph by Karcher that was never published. We provide original proofs for these inequalities. The third inequality appears in (Karcher 1970), but with a different proof. The last inequality is entirely from (Karcher 1970).

Proposition 5.4.10. *Let (M, g) be a Riemannian manifold with bounded sectional curvature $\delta \leq \sec \leq \Delta$ and define*

$$\varepsilon = \frac{\Delta - \delta}{2} \qquad \mu = \frac{\Delta + \delta}{2} \qquad K = \max\{|\Delta|, |\delta|\}.$$

Given a point $p \in M$, we have the following inequalities

$$|R(x, y, y, w) - R_\mu(x, y, y, w)| \leq \varepsilon \|x\| \|y\|^2 \|w\| \qquad \forall x, y, w \in T_pM \tag{5.11}$$

$$|R(x, y, z, w) - R_\mu(x, y, z, w)| \leq \frac{4}{3}\varepsilon \|x\| \|y\| \|z\| \|w\| \quad \forall x, y, z, w \in T_pM \tag{5.12}$$

The constants 1 and $\frac{4}{3}$ are tight.

Furthermore, if $e \in T_pM$ is a unitary vector, defining $R^\perp(x, y)z$ as the component of the curvature tensor perpendicular to e, we have that for every y, z perpendicular to e

$$\|R^\perp(e, y)z\| \leq \frac{4}{3}\varepsilon \|y\| \|z\| \tag{5.13}$$

$$\|R(e, y)z\| \leq 2K. \tag{5.14}$$

Proof. We may assume that the vectors x, y, z, w are of norm 1 by linearity. For (5.11), since the curvature tensor is skew-symmetric in the first two components and the last two components, we may assume that x, w are perpendicular to y. Fix a vector y and consider the bilinear form

$$T_y(x, w) = R(x, y, y, w) - R_\mu(x, y, y, w).$$

This application is symmetric, and as such, it attains its maximum at an eigenvector x_1 of norm one orthogonal to y so that for every x

$$|T_y(x, w)| \leq |R(x_1, y, y, x_1) - R_\mu(x_1, y, y, x_1)| = |\sec(x_1, y) - \mu| \leq \varepsilon.$$

For the second inequality, by linearity we can assume that all the vectors have the same norm. By polarisation, we have that

$$\begin{aligned} 6R(x, y, z, w) = R(x, y + z, y + z, w) - R(x, y - z, y - z, w) \\ - R(y, x + z, x + z, w) + R(y, x - z, x - z, w). \end{aligned}$$

We then apply the triangle inequality to the expression for $(R - R_\mu)(x, y, z, w)$ together with (5.11) to get

$$6|(R - R_\mu)(x, y, z, w)| \leq \varepsilon\Big(\|x\|\|w\|(\|y + z\|^2 + \|y - z\|^2) + \|y\|\|w\|(\|x + z\|^2 + \|x - z\|^2)\Big)$$

and using the parallelogram law together with the fact that all the vectors have the same norm, we get that

$$\|x + y\|^2 + \|x - y\|^2 = 2(\|x\|^2 + \|y\|^2) = 4\|x\|\|y\|$$

and the second inequality follows.

These two inequalities are tight on $\mathbb{C}P^n$ seen as a real manifold (Karcher 1970).

The bound on the normal part of the curvature tensor follows directly from (5.12), since $R_\mu(e, y, z, w) = 0$ whenever y, z, w are perpendicular to e, so choosing w as the unitary vector in the direction of $R^\perp(e, y)z$ we get that

$$\|R^\perp(e, y)z\| = |R(e, y, z, w)| \leq \frac{4}{3}\varepsilon\|y\|\|z\|.$$

The last inequality is proved in (Karcher 1970). □

5.4.4 A comparison lemma

We shall proceed in a very similar way to how we did in the case of the first order equation. We will simplify the differential equation to one in one dimension, and there, we will use a comparison theorem for functions of real variable. The only difference is that, in this case, we will not be able to simplify the computations to a Riccati equation. We start by proving the second order version of the Riccati comparison lemma but for the Jacobi equation. This lemma is stated in (Kaul 1976), but the reference given is in German and does not prove this inequality. We provide here a proof for completeness.

Lemma 5.4.11 (Jacobi comparison lemma). *Fix a real number $\kappa \in \mathbb{R}$, and let $f, g\colon [0, r] \to \mathbb{R}^+$ be two functions such that*

$$\ddot{f} + \kappa f \leq \ddot{g} + \kappa g \qquad f(0) = g(0), \dot{f}(0) \leq \dot{g}(0).$$

Then $f \leq g$ on $[0, \min\{r, \pi_\kappa\}]$.

Proof. Let $h = g - f$ and $\zeta = \ddot{h} + \kappa h \geq 0$. Let us show that the solution to the linear inhomogeneous initial value problem

$$\ddot{h} + \kappa h = \zeta \qquad h(0) = 0,\ \dot{h}(0) \geq 0$$

is indeed positive on the given interval. Solving the equation, we find that the solution is given by

$$\begin{aligned} h(x) &= \operatorname{sn}_\kappa(x) \int_0^x \operatorname{sn}'_\kappa(t)\zeta(t)\,\mathrm{d}t - \operatorname{sn}'_\kappa(x) \int_0^x \operatorname{sn}_\kappa(t)\zeta(t)\,\mathrm{d}t + \dot{h}(0)\operatorname{sn}_\kappa(x) \\ &= \int_0^x \operatorname{sn}_\kappa(x-t)\zeta(t)\,\mathrm{d}t + \dot{h}(0)\operatorname{sn}_\kappa(x). \end{aligned}$$

where we have used the trigonometric identity

$$\operatorname{sn}_\kappa(x-t) = \operatorname{sn}_\kappa(x)\operatorname{sn}'_\kappa(t) - \operatorname{sn}'_\kappa(x)\operatorname{sn}_\kappa(t).$$

From this we see that $h(x) \geq 0$ on $[0, \min\{r, \pi_\kappa\}]$ as $\operatorname{sn}_\kappa(x-t)$, $\zeta(t)$, $\dot{h}(0)$ and $\operatorname{sn}_\kappa(x)$ are positive in this interval. □

We can now show how to estimate the inhomogeneous differential equation for the Hessian of the exponential, taking advantage of the fact that the initial conditions are both zero.

Proposition 5.4.12 (Kaul, 1976). *Let (M, g) be a Riemannian manifold with bounded sectional curvature $\delta \leq \sec \leq \Delta$. Let $\gamma\colon [0, r] \to M$ be a geodesic, and let X, Y be vector fields along γ with $X, Y \perp \dot{\gamma}$ such that*

$$\ddot{X} + R(X, \dot{\gamma})\dot{\gamma} = Y \qquad X(0) = 0, \dot{X}(0) = 0.$$

Assume that there exists a continuous η function such that $\|Y\| \leq \eta$ on $[0, r]$. Then, we have that $\|X\| \leq \rho$ on $[0, \min\{r, \pi_{\frac{\Delta+\delta}{2}}\}]$, where ρ is the solution of

$$\ddot{\rho} + \delta\rho = \eta \qquad \rho(0) = 0, \dot{\rho}(0) = 0.$$

Proof. We define again the quantities

$$\varepsilon = \frac{\Delta - \delta}{2} \qquad \mu = \frac{\Delta + \delta}{2}.$$

Fix a $t_0 \in [0, r]$ and let E be the parallel vector field along γ such that $E(t_0) = \frac{X(t_0)}{\|X(t_0)\|}$. The function $f = \langle X, E\rangle$ satisfies the differential equation

$$\begin{aligned} \ddot{f} + \mu f &= \langle \ddot{X} + \mu X, E\rangle \\ &= \langle Y - R(X, \dot{\gamma})\dot{\gamma} + R_\mu(X, \dot{\gamma})\dot{\gamma}, E\rangle \\ &\leq \|Y\| + \varepsilon\|X\| \end{aligned}$$

where we have used Lemma 5.4.9 in the first equality and (5.11) for the bound.

Define g as the solution to the differential equation

$$\ddot{g} + \mu g = \|Y\| + \varepsilon\|X\| \qquad g(0) = \dot{g}(0) = 0.$$

Using that $f(t_0) = \|X(t_0)\|$ and that $f(0) = \dot{f}(0) = 0$, by Lemma 5.4.11, we get that $\|X(t_0)\| \leq g(t_0)$, and since the definition of g does not depend on the chosen t_0, we may do this for any $t_0 \in [0, r]$ getting that $\|X\| \leq g$ on $[0, \min\{\pi_\mu, r\}]$. Now, Using that $\|X\| \leq g$, we get that g satisfies the inequality

$$\ddot{g} + \delta g \leq \|Y\| \leq \eta.$$

So, for the solution of

$$\ddot{\rho} + \delta\rho = \eta \qquad \rho(0) = 0, \dot{\rho}(0) = 0.$$

we have that $g \leq \eta$ on $[0, \min\{\pi_\delta, r\}] \supseteq [0, \min\{\pi_\mu, r\}]$, getting the result. □

5.4.5 A second order version of Rauch's theorem

We are now ready to give bounds on the Hessian of the exponential map. We first note that, since our goal is to bound the norm of the Hessian of $f \circ \exp_p$, as this Hessian is symmetric, it attains its maximum at an eigenvector. As such, we just need to bound the quantity

$$(\nabla \mathrm{d}\exp_p)_v(w, w) \qquad v \in \mathrm{seg}^0(p), w \in T_pM.$$

For this reason, we will start by giving bounds on the diagonal of the Hessian of the exponential map. We will then see that, since the Hessian is a symmetric bilinear map, we can leverage these bounds to give bounds on the full Hessian for any pair of vectors $w_1, w_2 \in T_pM$.

We give the second order bounds on the exponential map for a manifold of $(\delta, \Delta, \Lambda)$-bounded geometry (*cf.* Definition 5.4.6).

Theorem 5.4.13 (Second order bounds for the exponential map). *Let (M, g) be a Riemannian manifold with $(\delta, \Delta, \Lambda)$-bounded geometry. For a geodesic $\gamma\colon [0, r] \to M$ with initial unit vector v, and a vector $w \in T_pM$, we have that*

- *If w is radial to $\dot{\gamma}(0)$,*

$$(\nabla \mathrm{d}\exp_p)_{rv}(w, w) = 0.$$

- *If w is normal to $\dot{\gamma}(0)$, the radial part of the Hessian is bounded as*

$$\Big(\frac{1}{r} - \frac{\mathrm{sn}_{4\delta}(r)}{r^2}\Big)\|w\|^2 \leq \langle(\nabla \mathrm{d}\exp_p)_{rv}(w, w), \dot{\gamma}(r)\rangle \leq \Big(\frac{1}{r} - \frac{\mathrm{sn}_{4\Delta}(r)}{r^2}\Big)\|w\|^2$$

  for $r < \pi_\Delta$ for the upper bound and $r < r_{\mathrm{conj}}(v)$ for the lower bound.

  The normal part of the Hessian is bounded for $r < \pi_{\frac{\Delta+\delta}{2}}$ as

$$\|(\nabla \mathrm{d}\exp_p)_{rv}(w, w)^\perp\| \leq \rho(r)\|w\|^2$$

  where

$$\rho(t) = \frac{8}{9r^2}\,\mathrm{sn}_\delta\Big(\frac{t}{2}\Big)^2\Big(3\Lambda\,\mathrm{sn}_\delta\Big(\frac{t}{2}\Big)^2 + 2(\Delta - \delta)\,\mathrm{sn}_\delta(t)\Big).$$

Both bounds and their radii are tight in spaces of constant curvature.

Furthermore, the radius of convergence for the normal part is tight for $\mathrm{SO}(n)$.

Proof. As in Proposition 5.4.1, we write

$$\begin{aligned}\tilde{J}(t,s) &= (\mathrm{d}\exp_p)_{t(v+sw)}\Big(\frac{t}{r}w\Big)\\ J(t) &= \tilde{J}(t,0)\\ K(t) &= (\nabla \mathrm{d}\exp_p)_{tv}\Big(\frac{t}{r}w, \frac{t}{r}w\Big).\end{aligned}$$

By linearity of the Hessian, it is enough to prove the result for $\|w\| = 1$.

If w is radial, in Proposition 5.4.1 we have that $Y = 0$, so K is a solution of the equation

$$\ddot{K} + R(K, \dot{\gamma})\dot{\gamma} = 0 \qquad K(0) = 0, \dot{K}(0) = 0$$

and since it is a homogeneous second order linear equation with zero as the initial condition, its solution is $K(t) = 0$ for $t \in [0, r]$.

If w is normal, we need to bound the quantity $\nabla_J \tilde{J}$. Note that this is a vector field along γ, but to take the derivative in the direction of J we need to have $\tilde{J}$ defined in that direction as well. We start by bounding its radial part

$$\langle \nabla_J \tilde{J}, \nabla r\rangle = D_J\langle \tilde{J}, \nabla r\rangle - \langle \tilde{J}, \nabla_J \nabla r\rangle = D_J\langle \tilde{J}, \nabla r\rangle - \operatorname{Hess} r(J, J). \tag{5.15}$$

We can compute the first term directly. By Gauss's lemma we can simplify this derivative to one on T_pM

$$D_J\langle \tilde{J}, \nabla r\rangle(\gamma(t)) = \frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}s}\Big|_{s=0}\Big\langle \frac{t}{r}w, \frac{t(v+sw)}{\|t(v+sw)\|_{T_pM}}\Big\rangle_{T_pM} = \frac{t}{r^2}.$$

Using the bounds on the Hessian of the distance function given in Theorem 5.3.16 we can bound the second term in (5.15). Evaluating these two quantities at r and using the bounds on the size of the Jacobi fields together with the trigonometric identity

$$\operatorname{sn}_\kappa(t)\operatorname{sn}'_\kappa(t) = \frac{\operatorname{sn}_\kappa(2t)}{2} = \operatorname{sn}_{4\kappa}(t) \tag{5.16}$$

we get the bounds on the tangential part of the Hessian of the exponential.

Finally, for its normal part, consider the differential equation given by Proposition 5.4.1. Since $\langle K, \dot{\gamma}\rangle' = \langle \dot{K}, \dot{\gamma}\rangle$, the radial (resp. normal) part of the derivative is the derivative of the radial (resp. normal) part. For this reason, we have that

$$(K^\perp)'' + R(K^\perp, \dot{\gamma})\dot{\gamma} = -Y^\perp.$$

Therefore, we just have to bound $\|Y^\perp\|$ to be able to use Proposition 5.4.12 and finish.

We start by giving a bound on the norm of $\dot{J}$. Using that $\dot{J} = \nabla_J \nabla r = \operatorname{Hess} r(J)$,

$$\|\dot{J}\| \le \|\operatorname{Hess} r\|\|J\|. \tag{5.17}$$

We can then bound the norm of $Y^\perp$ as

$$\begin{aligned}\|Y^\perp\| &\le \|(\nabla_{\dot{\gamma}}R)(J,\dot{\gamma})J + (\nabla_J R)(J,\dot{\gamma})\dot{\gamma}\| + 4\|R^\perp(J,\dot{\gamma})\dot{J}\|\\ &\le 2\Lambda\|J\|^2 + \frac{8(\Delta-\delta)}{3}\|J\|\|\dot{J}\|\\ &\le \Big(2\Lambda + \frac{8(\Delta-\delta)}{3}\operatorname{ct}_\delta(t)\Big)\|J\|^2\\ &\le \Big(2\Lambda + \frac{8(\Delta-\delta)}{3}\operatorname{ct}_\delta(t)\Big)\frac{\operatorname{sn}_\delta(t)^2}{r^2}\\ &= \frac{2}{r^2}\Big(\Lambda\operatorname{sn}_\delta(t)^2 + \frac{2(\Delta-\delta)}{3}\operatorname{sn}_\delta(2t)\Big).\end{aligned}$$

where we have used (5.13)—given that since J is perpendicular to $\dot\gamma$, so is $\dot J$—to bound the normal part of the curvature tensor. We have also used the first order bound on the differential of the exponential for perpendicular initial conditions to bound the norm of the Jacobi fields and (5.17) to bound their derivative. In the last equality we have used (5.16) again.

The result follows by noting that ρ is the solution to the differential equation

$$\ddot\rho + \delta\rho = \eta \qquad \rho(0) = \dot\rho(0) = 0$$

where η is the bound on $\|Y^\perp\|$ and applying Proposition 5.4.12.

We will see that the radius for the bound of the normal part is tight in the case of $\mathrm{SO}(n)$ in Example 5.5.4. □

The first thing to note is that these bounds go to zero as r tends to zero. This is exactly what we expect, as the Christoffel symbols in normal coordinates vanish at the origin.

The bounds in this theorem provide a notable improvement compared to the best bounds previously known (*cf.* (Kaul 1976)). For one, these bounds are tighter. We also have that these bounds are explicit, compared to the previous bounds, which were given in terms of a solution of a differential equation that did not have an explicit integral. We also simplified the technical tools necessary to get to these bounds.

We finish this section by giving a bound on the full Hessian of the exponential.

Theorem 5.4.14 (Bounds on the Full Hessian). *Let (M, g) be a Riemannian manifold with $(\delta, \Delta, \Lambda)$-bounded geometry. For a geodesic $\gamma\colon [0, r] \to M$ with initial unit vector v, $r < \pi_{\frac{\Delta+\delta}{2}}$, and any two vectors $w_1, w_2 \in T_pM$, we have that*

$$\|(\nabla \mathrm{d}\exp_p)_{rv}(w_1, w_2)\| \leq \frac{8}{3r^2}\,\mathrm{sn}_\delta\Big(\frac{r}{2}\Big)^2\big(\Lambda\,\mathrm{sn}_\delta\Big(\frac{r}{2}\Big)^2 + 2\max\{|\Delta|, |\delta|\}\,\mathrm{sn}_\delta(r)\big)\|w_1\|\|w_2\|.$$

Furthermore, the radius $\pi_{\frac{\Delta+\delta}{2}}$ is tight for $\mathrm{SO}(n)$.

Proof. To bound the full Hessian we first see that it is just enough to bound its diagonal part. For any symmetric bilinear form and any two vectors we have the polarisation formula

$$4\Phi(u, v) = \Phi(u + v, u + v) - \Phi(u - v, u - v).$$

By linearity, we may assume that $\|u\| = \|v\| = 1$. Taking absolute values and applying the triangle inequality and Cauchy–Schwarz, we get that

$$4\|\Phi(u, v)\| \leq \|\Phi|_{\mathrm{diag}}\|(\|u + v\|^2 + \|u - v\|^2) = 4\|\Phi|_{\mathrm{diag}}\|$$

where $\|\Phi|_{\mathrm{diag}}\|$ is the operator norm of the application $u \mapsto \Phi(u, u)$. For this reason, it is enough to bound the map $u \mapsto (\nabla \mathrm{d}\exp_p)_{tv}(u, u)$.

Let w be a vector normal to v. As we did in Theorem 5.4.13, we consider the differential equation for

$$K(t) = (\nabla \mathrm{d}\exp_p)_{tv}\Big(\frac{t}{r}w, \frac{t}{r}w\Big).$$

This is exactly the same differential equation that we had for $K^\perp$, only that rather than having to bound the normal part of the curvature tensor, we bound the full curvature tensor. For that we use (5.14), and

following with the bounds as we did in Theorem 5.4.13 and solving the resulting equation we get that for w perpendicular to v, the norm of K is bounded by the solution of the differential equation

$$\ddot{\rho} + \delta\rho = \frac{2}{r^2}(\Lambda \operatorname{sn}_\delta(t)^2 + 2\max\{|\Delta|, |\delta|\}\operatorname{sn}_\delta(2r)) \qquad \rho(0) = \dot{\rho}(0) = 0$$

which is solved by

$$\rho(t) = \frac{8}{3r^2}\operatorname{sn}_\delta\left(\frac{t}{2}\right)^2(\Lambda \operatorname{sn}_\delta\left(\frac{t}{2}\right)^2 + 2\max\{|\Delta|, |\delta|\}\operatorname{sn}_\delta(t)).$$ □

Remark 5.4.15 (Tighter bounds). Another way to obtain bounds on the full Hessian of the exponential would be to take the bounds from Theorem 5.4.13 for the normal and parallel part of the Hessian and using the Cauchy–Schwarz inequality to get bounds of the form

$$\|(\nabla \mathrm{d}\exp_p)_{rv}(w_1, w_2)\| \leq \sqrt{\sigma^2 + \rho^2}\|w_1\|\|w_2\|.$$

Where ρ is the bound of the normal part defined in Theorem 5.4.13 and

$$\sigma = \frac{1}{r^2}\max\{|r - \operatorname{sn}_{4\delta}(r)|, |r - \operatorname{sn}_{4\Delta}(r)|\}$$

that is, σ is the bound on the norm of the parallel part of the Hessian.

This bound, although tighter in most specific examples, might be more difficult to manipulate and lacks the simplicity of that presented in the theorem above.

5.5 Concrete Second Order Bounds

5.5.1 Constant curvature

The simplest family to evaluate these bounds in is that of the spaces of constant curvature such as the sphere, the hyperbolic plane, the Euclidean space or the flat torus.

For a manifold of constant curvature κ, since the curvature is constant, the derivative of the curvature tensor is zero. If we further assume that their injectivity radius is positive[2], as is the case of the hyperbolic space and the sphere, we have that they have $(\kappa, \kappa, 0)$-bounded geometry. Instantiating the bounds for these spaces we see that

$$\langle(\nabla \mathrm{d}\exp_p)_{rv}(w, w), \dot{\gamma}(r)\rangle = \left(\frac{1}{r} - \frac{\operatorname{sn}_{4\kappa}(r)}{r^2}\right)\|w\|^2$$
$$\|(\nabla \mathrm{d}\exp_p)_{rv}(w, w)^\perp\| = 0$$

for $r < \pi_\kappa$. As a first sanity check, we see that for the flat case $\kappa = 0$, the radial part is exactly equal to zero, as the whole Hessian of the exponential map is everywhere zero in this case—the second derivative of an affine function is zero.

To check that this solution is actually correct, we can solve the differential equation in Proposition 5.4.1 for these spaces for

$$K(t) \coloneqq (\nabla \mathrm{d}\exp_p)_{tv}(tw_1, tw_2).$$

[2]This is not really used to prove the bounds, but it is part of the definition of bounded geometry.

Using Lemma 5.4.9, together with the fact that, since the sectional curvature is constant, $\nabla R = 0$, and the trigonometric identity

$$\mathrm{sn}_\kappa(t)\,\mathrm{sn}'_\kappa(t) = \frac{\mathrm{sn}_\kappa(2t)}{2} = \mathrm{sn}_{4\kappa}(t)$$

we have that the differential equation for the constant curvature case is given by

$$\ddot{K}(t) + \kappa K^\perp(t) = 4\kappa\,\mathrm{sn}_{4\kappa}(t)\dot{\gamma}(t) \qquad K(0) = 0,\ \dot{K}(0) = 0$$

We can split this equation into its normal and radial part. The normal part is clearly zero, since the right-hand side is zero, and the remaining equation is linear with zero as the initial condition. For the radial part, setting $x = \langle K, \dot{\gamma}\rangle$ we get

$$\ddot{x}(t) = 4\kappa\,\mathrm{sn}_{4\kappa}(t) \qquad x(0) = 0,\ \dot{x}(0) = 0.$$

Finally, $\frac{x(r)}{r^2} = \frac{r - \mathrm{sn}_{4\kappa}(r)}{r^2}$ is exactly the value announced before.

5.5.2 Locally symmetric spaces

Locally symmetric spaces define a large family of particularly well-behaved Riemannian manifolds. Examples of these spaces are the flat torus, the orthogonal group (or any compact Lie group with a bi-invariant metric), the space of symmetric positive definite matrices, the Grassmannian, the oriented Grassmannian and the hyperbolic Grassmannian.[3]

We recall the algebraic definition of a locally symmetric space.

Definition 5.5.1. A Riemannian manifold (M, g) is **locally symmetric** if the curvature tensor is covariantly constant, that is, $\nabla R = 0$.

Locally symmetric spaces were introduced and studied by Cartan in 1926, who also gave a complete classification of these in 1932.

The most notable examples of locally symmetric spaces are symmetric spaces which are one of the most important families of real manifolds in Riemannian geometry. These were intensively studied by Sigurður Helgason (Helgason 1978).

Definition 5.5.2. A Riemannian manifold (M, g) is a **symmetric space** if, for every point $p \in M$ there exists an involutive isometry σ_p that fixes p, that is

$$\sigma_p(p) = p \qquad (\mathrm{d}\sigma_p)_0 = -\mathrm{Id}.$$

As the name implies, one may prove that symmetric spaces are indeed locally symmetric spaces.

For these manifolds we have the following result.

Proposition 5.5.3. *A symmetric space with bounded sectional curvature $\delta \le \sec \le \Delta$ is of $(\delta, \Delta, 0)$-bounded geometry.*

Proof. Since it is a locally symmetric space, we have that $\nabla R = 0$, so the first order bound is clear. Since symmetric spaces are Riemannian homogeneous spaces, for every two points there exists an isometry taking one to the other. As such, the injectivity radius is constant throughout the manifold and thus, positive. □

[3] All these manifolds are to be regarded as Riemannian manifolds with the metric inherited from their quotient structure

From this, we get that we just need to compute bounds on the sectional curvature for these manifolds in order to give second order bounds for the exponential map. Now, bounds on the sectional curvature of these manifolds are well-known. We give here some examples that are particularly useful in the context of optimisation.

Example 5.5.4 (The special orthogonal group). The sectional curvature for a Lie group with a bi-invariant metric, after identifying any pair of tangent vector with vectors in the Lie algebra, is given for a pair of orthonormal vectors $X, Y \in \mathfrak{g}$

$$\sec(X, Y) = \frac{1}{4}\|[X, Y]\|^2.$$

In the special case of $\mathrm{SO}(n)$ for $n > 2$, we have that $\mathfrak{g} \cong \mathrm{Skew}(n)$. It is clear that the sectional curvature is non-negative for any bi-invariant metric. In the case of $\mathrm{SO}(n)$ this bound is tight.

Consider the bi-invariant metric given by the Frobenius norm—the metric inherited from $\mathbb{R}^n$. For the upper bounds, if we work with a matrix Lie group, it is enough to bound the norm of $XY - YX$ for matrices X, Y of Frobenius norm 1. In the general case, this inequality is called the Böttcher–Wenzel inequality and it reads

$$\|[X, Y]\| \leq 2\|X\|\|Y\| \qquad \forall X, Y \in \mathbb{R}^{n\times n}.$$

For a review of this inequality and a particularly clean proof see (Lu 2012).

For the case of $\mathrm{SO}(n)$, that is, when X, Y are skew-symmetric, this inequality can be improved (Bloch and Iserles 2005)

$$\|[X, Y]\| \leq \|X\|\|Y\| \qquad \forall X, Y \in \mathrm{Skew}(n).$$

For $n = 3$ the constant can be further improved to $\frac{1}{2}$. These constants are tight.

Wrapping all this together, we get that the bounds for $\mathrm{SO}(n)$ with $n > 2$ are given by

$$\begin{aligned} 0 \leq \langle(\nabla \mathrm{d}\exp_p)_{rv}(w, w), \dot\gamma(r)\rangle &\leq \Big(\frac{1}{r} - \frac{\sin(r)}{r^2}\Big)\|w\|^2 && r < 2\pi \\ \|(\nabla \mathrm{d}\exp_p)_{rv}(w, w)^\perp\| &\leq \frac{r}{9}\|w\|^2 && r < 2\sqrt{2}\pi. \end{aligned}$$

Note that the radius of definition of the normal part of the Hessian are tight, as the conjugate radius of $\mathrm{SO}(n)$ is exactly $2\sqrt{2}\pi$. This can be seeing by noting that the exponential of matrices is not full rank at matrices with two eigenvalues that are $2\pi i$ apart.

In particular, we can give a bound on the full Hessian by bounding the derivative of $\sqrt{\sigma^2 + \rho^2}$ where σ and ρ are the bounds on the tangential and normal part of the Hessian. This gives

$$\|(\nabla \mathrm{d}\exp_p)_{rv}(w_1, w_2)\| \leq \frac{2r}{9}\|w_1\|\|w_2\| \qquad r < 2\pi.$$

We can see that Theorem 5.4.14 gives us a coarser bound, but on the other hand, the bound is defined on a larger radius. In particular, we get

$$\|(\nabla \mathrm{d}\exp_p)_{rv}(w_1, w_2)\| \leq \frac{r}{3}\|w_1\|\|w_2\| \qquad r < 2\sqrt{2}\pi.$$

This radius is tight, as there are points that are $2\sqrt{2}\pi$ apart from any given point which are conjugate to it. This can be seen by computing the eigenvalues of the differential of the exponential map (see for example Lezcano-Casado and Martínez-Rubio 2019, Theorem D.2).

Better bounds are possible by using a tighter version of the bounds on the curvature tensor at the expense of having an uglier numeric constant. For example, the $\frac{1}{3}$ constant can be improved this way to $\frac{\sqrt{61}}{36}$.

These same ideas can be generalised to symmetric spaces. We will use this to compute bounds for the Hessian of the exponential map in the Grassmannian.

Example 5.5.5 (Sectional Curvature of a Symmetric space). Let G/H be a symmetric space. Since a symmetric space is a Riemannian homogeneous space, we just need to bound the sectional point at one point, as the space has the same sectional curvature at every point. Denote by $\mathfrak{m} = \mathfrak{h}^{\perp} \subseteq \mathfrak{g}$ the orthogonal complement of the Lie algebra of H with respect to the metric at the identity in G. This complement to $\mathfrak{h}$ is not a Lie algebra itself and in fact $[\mathfrak{m}, \mathfrak{m}] \subseteq \mathfrak{h}$, as G/H is a symmetric space. This set $\mathfrak{m}$ may be identified isometrically with the tangent space at $\pi(e)$, the projection of the identity element $e \in G$, in other words, the map

$$(\mathrm{d}\pi)_e|_{\mathfrak{m}} \colon \mathfrak{m} \to T_{\pi(e)}G/H$$

is a linear isometry. Via this identification we may treat vectors $X, Y \in T_{\pi(e)}G/H$ as vectors $\overline{X}, \overline{Y} \in \mathfrak{m} \subseteq \mathfrak{g}$. Now, by O'Neill's formula (O'Neill 1966, Ch. 7, Thm. 47), the sectional curvature for these manifolds has a particularly simple formula for orthonormal vectors $X, Y \in T_{\pi(e)}G/H$

$$\sec_{G/H}(X, Y) = \sec_G(\overline{X}, \overline{Y}) + \frac{3}{4}\|[\overline{X}, \overline{Y}]\|^2$$

where we have used that $[\mathfrak{m}, \mathfrak{m}] \subseteq \mathfrak{h}$.

If G is a Lie group with a bi-invariant metric we say that G/H is a **normal symmetric space**. For a normal metric, the sectional curvature simplifies to

$$\sec_{G/H}(X, Y) = \|[\overline{X}, \overline{Y}]\|^2.$$

Trough the study of symmetric spaces of non-compact type and their duality, it is not difficult to prove that the sectional curvature on any symmetric space is either the norm or minus the norm of the Lie bracket of a pair of vectors, although we will not prove that as we will not need it.

Example 5.5.6 (The real Grassmannian). The real Grassmannian as a symmetric space is given by the quotient $\mathrm{Gr}(n, k) = \mathrm{SO}(n)/(\mathrm{SO}(k) \times \mathrm{SO}(n-k))$, where the metric on $\mathrm{SO}(n)$ is the bi-invariant metric generated by the scalar product $\langle X, Y \rangle = \frac{1}{2}\operatorname{tr}(X^{\intercal}Y)$ on the Lie algebra. For this symmetric space we have that

$$\mathfrak{h} = \mathfrak{so}(k) \otimes \mathfrak{so}(n-k) = \left\{ \begin{pmatrix} B & 0 \\ 0 & C \end{pmatrix} \;\middle|\; B \in \mathrm{Skew}(k), C \in \mathrm{Skew}(n-k) \right\}$$
$$\mathfrak{m} = \left\{ \begin{pmatrix} 0 & A \\ -A^{\intercal} & 0 \end{pmatrix} \;\middle|\; A \in \mathbb{R}^{n-k \times k} \right\}$$

Note that the metric is chosen so that for $\overline{X} \in \mathfrak{m}$ we have that $\|\overline{X}\| = \|X\|$, where $X \in \mathbb{R}^{n-k \times k}$. This is so that this norm agrees with the usual norm in the projective plane as $\mathrm{Gr}(n, 1) \cong \mathbb{R}P^n$.

Since the metric on $G = \mathrm{SO}(n)$ is bi-invariant, the Grassmannian is a normal symmetric space, so the sectional curvature is given by the norm of the commutator of elements in $\mathfrak{m}$. Using the bound on the

norm of the Lie bracket for skew-symmetric matrices, we would get

$$0 \leq \sec(X, Y) \leq 4.$$

As it can be seen by (Ge 2014, Lemma 2.5), these bounds are not tight. They can be refined as announced in (Wong 1968, Theorem 3a) and proved in (Hildebrandt, Jost, and Widman 1980, p.292) via an application of Cauchy–Schwarz as

$$0 \leq \sec(X, Y) \leq 2.$$

Using these bounds, we get analogous bounds for the Hessian of the exponential map for the Grassmannian,

$$\begin{aligned} 0 \leq \langle (\nabla \mathrm{d}\exp_p)_{rv}(w,w), \dot\gamma(r)\rangle &\leq \Big(\frac{1}{r} - \frac{\sin(2\sqrt{2}r)}{2\sqrt{2}r^2}\Big)\|w\|^2 && r < \frac{\pi}{\sqrt{2}} \\ \|(\nabla \mathrm{d}\exp_p)_{rv}(w,w)^\perp\| &\leq \frac{8r}{9}\|w\|^2 && r < \pi. \end{aligned}$$

Note that for the real Grassmannian $\mathrm{r_{inj}} = \frac{\pi}{2}$ (Kozlov 2000), so the radii of these equations should be enough for any practical purposes. As in the case of $\mathrm{SO}(n)$, it is also direct to get linear bounds on the full Hessian.

5.6 Convergence Rates for Dynamic Trivialisations

We now have all the necessary tools to be able to complete the program stated in the introduction. In particular, we can prove the conditional convergence of the dynamic trivialisation framework for any stopping rule. We first recap the results of the previous two sections in the following proposition.

Proposition 5.6.1. *Let (M, g) be a connected and complete Riemannian manifold of $(\delta, \Delta, \Lambda)$-bounded geometry and let f be a function of α-bounded Hessian on it. Fix a point $p \in M$, and let $\mathcal{X} \subseteq U_p$ be a convex subset of M with $\operatorname{diam}(\mathcal{X}) \leq 2r \leq 2\pi\frac{\Delta+\delta}{2}$, and at least one critical point of f in it. Denote the pullback of $\mathcal{X}$ under the exponential as $\overline{\mathcal{X}} \coloneqq \exp_p^{-1}(\mathcal{X}) \subseteq T_pM$. Then, the map $f \circ \exp_p \colon \overline{\mathcal{X}} \to \mathbb{R}$ is of $\widehat{\alpha}_r$-bounded Hessian with constant*

$$\widehat{\alpha}_r = \alpha(C_{1,r} + C_{2,r})$$

$$C_{1,r} = \max\Big\{1, \frac{\operatorname{sn}_\delta(r)^2}{r^2}\Big\} \qquad C_{2,r} = \frac{8}{3}\operatorname{sn}_\delta\big(\frac{r}{2}\big)^2(\Lambda \operatorname{sn}_\delta\big(\frac{r}{2}\big)^2 + 2\max\{|\Delta|, |\delta|\}\operatorname{sn}_\delta(r)).$$

Proof. We will bound the Hessian of the map $f \circ \exp_p$ for an arbitrary $p \in M$. By the Leibnitz rule, we have that

$$\nabla \mathrm{d}(f \circ \exp_p) = \nabla(\mathrm{d}f \circ \mathrm{d}\exp_p) = \nabla \mathrm{d}f \circ \mathrm{d}\exp_p + \mathrm{d}f \circ \nabla \mathrm{d}\exp_p .$$

Taking norms, and since the left-hand side is a symmetric tensor, its maximum value—*i.e.*, its norm—is reached at a singular vector. As such, writing the bound explicitly

$$\|\nabla \mathrm{d}(f \circ \exp_p)\|_{\mathcal{X}} = \max_{\substack{v \in \exp^{-1}(\mathcal{X}) \\ w \in T_pM,\ \|w\|=1}} \Big[(\nabla \mathrm{d}f)_{\exp_p(v)}((\mathrm{d}\exp_p)_v(w), (\mathrm{d}\exp_p)_v(w)) + (\mathrm{d}f)_{\exp_p(v)}((\nabla \mathrm{d}\exp_p)_v(w,w))\Big].$$

Since there exists a critical point of f in U_p, and by the α-bound on the Hessian, we have that f is αr^2-Lipschitz on $\mathcal{X}$. Using this, the triangle inequality and Cauchy–Schwarz, we can bound this quantity as

$$\|\nabla \mathrm{d}(f \circ \exp_p)\|_{\mathcal{X}} \leq \alpha \max_{\substack{v \in \exp_p^{-1}(\mathcal{X}) \\ w \in T_pM,\ \|w\|=1}} \|(\mathrm{d}\exp_p)_v(w)\|^2 + \alpha r^2 \max_{\substack{v \in \exp_p^{-1}(\mathcal{X}) \\ w \in T_pM,\ \|w\|=1}} \|(\nabla \mathrm{d}\exp_p)_v(w,w)\|.$$

From Theorems 5.3.18 and 5.4.14, we have that $C_{1,r}, C_{2,r}$ come from the bounds for the square of the norm of the differential and the norm of the Hessian of the exponential respectively. □

After the heavy work of giving bounds on the Hessian of the pullback of a function along the exponential map, we are in a position to prove convergence rates for different instances of the dynamic trivialisation framework in terms of $\widehat{\alpha}_r$. We start with one of the simplest ones, namely **static trivialisations**. This is the algorithm that comes from choosing $\texttt{stop} \equiv \texttt{False}$ in Algorithm 2. Equivalently, this is the algorithm coming from solving

$$\min_{v \in T_pM} f(\exp_p(v))$$

using gradient descent on T_pM.

The critical assumption in this result is that iterates remain bounded inside a compact set $\mathcal{X} \subseteq T_pM$. This assumption is restrictive, but it is standard in previous work (Bonnabel 2013; Sato, Kasai, and Mishra 2019; Tripuraneni et al. 2018; Ahn and Sra 2020).

Theorem 5.6.2 (Convergence of static trivialisations). *Let (M, g) be a connected and complete Riemannian manifold of $(\delta, \Delta, \Lambda)$-bounded geometry and let f be a function of α-bounded Hessian on it. Fix a point $p \in M$, and let $\mathcal{X} \subseteq U_p$ be a convex subset of M with $\operatorname{diam}(\mathcal{X}) \leq R \leq 2\pi_{\frac{\Delta+\delta}{2}}$, and at least one critical point of f in it. Consider Algorithm 2 with the stopping rule $\texttt{stop} \equiv \texttt{False}$ and fixed step-size $\eta_{i,k} = \frac{1}{\widehat{\alpha}_{R/2}}$ where $\widehat{\alpha}_{R/2}$ is as in Proposition 5.6.1. If all the iterates of the method stay in $\mathcal{X}$, the method will find a point $v_{0,t} \in T_pM$ such that $\|\nabla(f \circ \exp_p)(v_{0,t})\| < \varepsilon$ in at most*

$$\left\lceil \frac{2\widehat{\alpha}_{R/2}}{\varepsilon^2}(f(\exp_p(v_{0,0})) - f^*) \right\rceil$$

steps, where f^* is a lower-bound of f on $\mathcal{X}$.

Proof. We proved in Proposition 5.6.1 that the map $f \circ \exp_p$ is of $\widehat{\alpha}_{R/2}$-bounded Hessian on $\mathcal{X}$. This can be regarded as a function on a Euclidean space, for which the convergence rate is well-known (*cf.*, Theorem 2.2.6). □

Remark 5.6.3 (The hypotheses of this convergence theorem). The condition $\mathcal{X} \subseteq U_p$ might seem scary at first, but it should not be since its complement—the cut locus $\operatorname{cut}(p) = M \backslash U_p$—has (Borel) measure zero (Corollary 4.4.9). Also, as explained in Section 4.4.2, the cut locus is, in some sense, *as far as possible from p*, so it should not pose any problems in practice.

As we can see, the point $p \in M$ does not play any role in the proof of Theorem 5.6.2, besides for the technical condition of the iterates being bounded in U_p. Generalising this assumption, we can prove at once the convergence of the scheme of dynamic trivialisations, for an arbitrary stopping rule.

Theorem 5.6.4 (Convergence of dynamic trivialisations). *Let (M, g) be a connected and complete Riemannian manifold of $(\delta, \Delta, \Lambda)$-bounded geometry and let f be a function of α-bounded Hessian on it. Assume that in the algorithm Algorithm 2 with an arbitrary stopping rule* `stop`*, all the iterates $v_{i,k}$ are contained in convex sets $\mathcal{X}_i \subseteq U_{p_i}$ with $\operatorname{diam}(\mathcal{X}_i) \leq R \leq 2\pi_{\frac{\Delta+\delta}{2}}$, with at least one critical point of f in each of them. Then, for the choice $\eta_{i,k} = \dfrac{1}{\widehat{\alpha}_{R/2}}$, the algorithm will find a point $v_{i,k} \in T_{p_i}M$ such that $\|\nabla(f \circ \exp_{p_i})(v_{i,k})\| < \varepsilon$ in at most*

$$\left\lceil \frac{2\widehat{\alpha}_{R/2}}{\varepsilon^2}(f(\exp_p(v_{0,0})) - f^*) \right\rceil$$

steps, where f^* is a lower-bound of f on $\mathcal{X}$.

Proof. Analogous to Theorem 5.6.2, as the proof does not depend on the pullback point p. □

This theorem can be considered as a building block to then be used in specific examples to get actual convergence rates under weaker assumptions. In plain words, this result asserts that if the algorithm is converging to a critical point, it is fine to change the trivialisation point, as the exponential map will not distort the metric too much.

Remark 5.6.5 (Practical considerations). Of course, these bounds are worst-case bounds, but one can do better in practice. As we have bounds for the distortion of the exponential map at every point, we can choose a dynamic step-length that accounts for this. For example, at a point with $\|v_{i,k}\| = s$, we could consider

$$\eta_{i,k} = \frac{1}{\widehat{\alpha}_s}.$$

This is not likely to give any improvement in a real-world problem, as the constant α is itself an upper-bound on the norm of the Hessian of the function itself, and the exponential map may incur in large variations of this second derivative on points far from 0.

This is particularly true when using a dynamic stopping rule that controls how much the norm of the gradient of the pullback deviates from the actual norm of the gradient of the function. In this case, one could consider having a rule similar to the one that we outlined in the introduction, but that does not only account for the case in which the algorithm is converging to the cut locus, which will mostly happen if the sectional curvature is positive, but also accounts for the case when the gradient of the pullback is much larger than that of the function, which might happen in cases of negative curvature

$$\texttt{stop} \equiv \left(\frac{\|\nabla(f \circ \exp_{p_i})(v_{i,k})\|}{\|\nabla f(x_{i,k})\|} < \varepsilon\right) \texttt{ or } \left(\frac{\|\nabla(f \circ \exp_{p_i})(v_{i,k})\|}{\|\nabla f(x_{i,k})\|} > \frac{1}{\varepsilon}\right).$$

This rule can be read as "change the trivialisation point when we detect that $\exp_p$ has deviated too much from being an isometry in the direction of the gradient". Of course, this rule can be adapted using the information that one has a priori about the geometry of the manifold and, in particular, its cut locus, conjugate locus or injectivity radius.

5.7 When Is a Retraction an Exponential Map?

As we have seen throughout this chapter and the previous one, both the exponential and retractions may be used to pullback problems to a tangent space, although it is the exponential map for which we can have more a-priori information in the general setting.

It is clear that the exponential map associated to an affine connection—not necessarily the Levi-Civita connection of a metric—is a retraction (Definition 4.1.1). A natural question is the converse: When is a retraction the exponential map associated to some connection? We give an algebraic way to check this in practice.

Theorem 5.7.1. *Let $r\colon TM \to M$ be a retraction on a differentiable manifold. This retraction is the exponential map associated to an affine connection on M if and only if r induces a flow. In symbols, writing $\gamma_{p,v}(t) \coloneqq r_p(tv)$ for $(p,v) \in TM$, then r comes from a connection if and only if*

$$\dot\gamma_{p,v}(t+s) = \dot\gamma_{\gamma_{p,v}(t),\dot\gamma_{p,v}(t)}(s) \qquad \forall t, s > 0.$$

Proof. By definition, the exponential map is the projection of the geodesic spray flow on TTM, so the exponential map defines a flow in the sense specified above.

For the reverse implication, assume that r defines a flow. Define the curves on TM

$$\sigma_{p,v} = (\gamma_{p,v}, \dot\gamma_{p,v})$$

and define the vector field on TM for $(p,v) \in TM$

$$S_{p,v} \coloneqq \dot\sigma_{p,v}(0) = (\dot\gamma_{p,v}(0), \frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \dot\gamma_{p,v}(t)) = (v, \frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \dot\gamma_{p,v}(t)) \in TTM$$

where we have used that r is a retraction on the last equality. Note that for $s > 0$

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \dot\gamma_{p,sv}(t) = s \frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \dot\gamma_{p,v}(st) = s^2 \frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=0} \dot\gamma_{p,v}(t)$$

so the second term is 2-homogeneous. This makes S into a spray, and by the Ambrose–Palais–Singer theorem (Ambrose, Palais, and I. M. Singer 1960), there exists an affine connection that has S as its spray with its exponential map being the projection of the flow of the spray evaluated at time 1.

To finish, we are just missing checking that the curves $\sigma_{p,v}$ are the integral curves of S, but this is direct by using that r comes from a flow

$$\begin{aligned}
\dot\sigma_{p,v}(t) &= \frac{\mathrm{d}}{\mathrm{d}s}\Big|_{s=0} \sigma_{p,v}(t+s) \\
&= (\dot\gamma_{p,v}(t), \frac{\mathrm{d}}{\mathrm{d}s}\Big|_{s=0} \dot\gamma_{p,v}(t+s)) \\
&= (\dot\gamma_{p,v}(t), \frac{\mathrm{d}}{\mathrm{d}s}\Big|_{s=0} \dot\gamma_{\sigma_{p,v}(t)}(s)) \\
&= S_{\sigma_{p,v}(t)}.
\end{aligned}$$

□

The result for retractions that are just defined on an open neighbourhood of the zero section of TM is analogous.

Remark 5.7.2. The connection on the previous proof is not unique as connections such as their difference is a skew-symmetric tensor induce the same spray. In particular, the connection in the proposition above can be chosen to be torsion-free.

Chapter 6

GeoTorch: A Library for Optimisation on Manifolds at Scale

In this chapter, we describe some practical applications of the dynamic trivialisation framework developed in Chapter 4. We show how to implement it with GPU efficiency and flexibility in mind.

These ideas have led to the development of the free-software library **GeoTorch**. GeoTorch allows for coupling manifold constraints with the GPU/TPU-friendly PyTorch library to allow performing optimisation on manifolds at scale. The library can be found under the MIT license at

`https://github.com/Lezcano/geotorch`

Some of these ideas were published in the papers (Lezcano-Casado and Martínez-Rubio 2019; Lezcano-Casado 2019) at ICML 2019 and NeurIPS 2019.

The main idea presented in Section 6.2.2 has been submitted to be added to core PyTorch, and it is currently being discussed at

`https://github.com/pytorch/pytorch/pull/33344`

It is hoped to be accepted and merged into core PyTorch in the next release (1.9.0).

The work in Section 6.3.2 led to the implementation of the matrix exponential in core PyTorch, which was implemented by Nikitaved in

`https://github.com/pytorch/pytorch/pull/40161`

This implementation yields a $\times 12$ speed-up over its Tensorflow counterpart.

Outline of the chapter. We start by giving a detailed overview of how to implement some first-order optimisation methods on $\mathrm{SO}(n)$ and an introduction to orthogonal constraints in RNNs in Section 6.1. In Section 6.2, we show how GeoTorch compares against the other open libraries to perform optimisation on manifolds, and we go over how to implement the core features that allow for its flexibility and simplicity. In Section 6.3, we show how to use these ideas to stabilise the training of RNNs. We also provide a literature review on the topic. Finally, in Section 6.4, we test experimentally on RNNs the dynamic trivialisation framework, comparing the efficiency of the methods presented in this thesis to previous methods from the literature.

6.1 Orthogonal Constraints within Neural Networks

We start by looking at the practical ideas that first motivated this whole thesis. In particular, we look at orthogonality constraints in the context of recurrent neural networks. These constraints help to alleviate the problems of vanishing and exploding gradients.

6.1.1 Vanishing and exploding gradient problems

Training deep neural networks presents many difficulties. Arguably the most important hindrances are the exploding and vanishing gradient problems, as first observed in (Bengio, Simard, and Frasconi 1994). This problem arises from the ill-conditioning of the function defined by a neural network as the number of layers increases. This issue is particularly problematic in Recurrent Neural Networks (RNNs).

Recall the definition of an RNN (*cf.*, Definition 2.3.4), for $t = 1, \dots, T$

$$z_t = Bh_{t-1} + Cx_t$$
$$h_t = \sigma(z_t),$$

where $B \in \mathbb{R}^{d\times d}$, $C \in \mathbb{R}^{d\times k}$ and σ is a non-linearity applied coordinate-wise. Suppose that we choose the last output vector h_T as the encoding of the sequence, and that we have a loss function ℓ that maps the encoding h_T and a label in the set $\mathcal{Y}$ onto the real numbers. Starting on $h_0 = 0$, we have a function

$$\ell \circ \mathtt{rnn}_{B,C} \colon (\mathbb{R}^k)^* \times \mathcal{Y} \to \mathbb{R}$$
$$(x_1, \dots, x_T), y \mapsto \ell(h_T, y)$$

We will denote the differential of this function with respect to the hidden states as $\dfrac{\partial \ell}{\partial h_t}$ and the gradients of the hidden states with respect to the canonical metric on the Euclidean space as $\left(\dfrac{\partial \ell}{\partial h_t}\right)^*$. Note that the adjoint with respect to this metric is simply the transpose, so this notation should be clear.

We can compute the differential iteratively using the chain rule,

$$\frac{\partial \ell}{\partial h_i} = \frac{\partial \ell}{\partial h_T}\frac{\partial h_T}{\partial h_{T-1}} \cdot \ldots \cdot \frac{\partial h_{i+1}}{\partial h_i} = \frac{\partial \ell}{\partial h_T}\prod_{t=i+1}^{T} D_t B \tag{6.1}$$

where $D_t = \operatorname{diag}(\sigma'(z_t))$. This formula hints at the general problem that happens in RNNs. When the matrix B has eigenvalues of norm different from 1, computing the gradient is similar to computing powers of B, which becomes numerically unstable. For example, if $\sigma'(x)$ is uniformly bounded by $\gamma > 0$ and the largest eigenvalue of B is less than $\frac{1}{\gamma}$, the norm of $\dfrac{\partial \ell}{\partial h_i}$ goes to zero exponentially fast in T.

This is problematic when computing the gradient of the recurrent kernel B which is given by

$$\left(\frac{\partial \ell}{\partial B}\right)^* = \sum_{t=1}^{T} h_{t-1} \otimes \left(\frac{\partial \ell}{\partial z_t}\right)^* = \sum_{t=1}^{T} h_{t-1} \otimes \left(D_t B^\intercal D_{t+1} B^\intercal \dots B^\intercal D_T \left(\frac{\partial \ell}{\partial h_T}\right)^*\right). \tag{6.2}$$

Note that, for $\sigma = \mathtt{relu}$—or, in general, for a simple piecewise linear function—the term D_t barely encodes any information about x_t. For the `relu` case D_t is simply a matrix with ones in some elements of the diagonal and zeros everywhere else. As such, most of the information about x_t in the gradients comes from the term h_t.

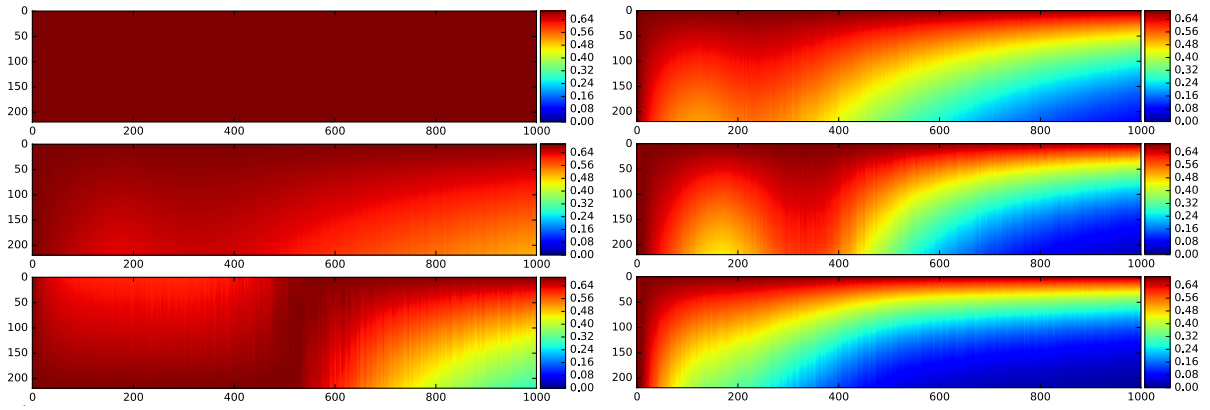

Figure 6.1: Vanishing gradient problem in an RNN. Each plot shows the evolution of the norm of the gradient $\left(\frac{\partial \ell}{\partial h_{T-i}}\right)^*$ for $T = 220$, throughout 1000 update iterations of an RNN without a non-linearity. In each plot, the recurrent kernel B of the RNN has singular values restricted to the interval $[1-m, 1+m]$ for $m = 0, 10^{-3}, 10^{-2}$, and $m = 10^{-1}, 1, \infty$ respectively. We see that, as the eigenvalues deviate from 1, the gradient flow degrades exponentially fast. Gradient norms are normalised across the time dimension (Vorontsov et al. 2017).

Since we often work with the `relu` activation for which $\frac{\mathrm{d}}{\mathrm{d}t}\texttt{relu}(t) = \mathbb{1}_{\geq 0}(t)$, **let's assume for the rest of the section that** $D_t = \mathrm{I}_n$ to simplify the discussion.

Under this assumption, the $i+1$-th term of the sum in (6.2) is given by

$$h_i \otimes ((B^\intercal)^{T-i} x), \tag{6.3}$$

for the vector $x = \left(\frac{\partial \ell}{\partial h_T}\right)^*$. In other words, if the norms of some of the eigenvalues of B are larger than 1, the norm of the gradient will explode—**exploding gradient problem**—while if the norms of all the eigenvalues of B are too small, the gradient will effectively vanish—**vanishing gradient problem**.

In practice, since the entries of the matrix B are often initialised according to a normal distribution of mean zero and small variance (Glorot and Bengio 2010), the eigenvalues of $B^\intercal$ tend to be small, meaning that we often encounter the vanishing gradient problem in practice. This translates to a very slow training, as the RNN learns the signal coming from the last elements of the sequence $(\ldots, x_{T-1}, x_T)$, for which $\|B\|^{T-i}$ is not too small and the norm of (6.3) is not too small, but it often struggles to recall the information coming from the first element $(x_1, x_2, \ldots)$ of the sequence.

The exploding gradient problem is often encountered when tuning the learning rate. If one uses a learning rate that is too large, the gradients may explode at the beginning of the training, which leads to overflows and the appearance of `NaN`s, which renders the rest of the training useless. In contrast, if the learning rate is too small, then the training will either converge very slowly or would just get stuck in a local minimum. To alleviate these problems, practitioners often resort to **learning rate schedulers** where the learning rate is changed throughout the optimisation process to try to avoid these scenarios. Vanishing gradient problems (*cf.*, Figure 6.1) and exploding gradient problems are often encountered in practice (Pascanu, Mikolov, and Bengio 2013).

Another thing that can be deduced from Equation (6.2) is that we should be looking to choose a non-linearity such that $\lim_{x \to \pm\infty} |\sigma'(x)| = 1$.[1] This condition allows the gradients to *flow back* when

[1] This is a relaxation of the stronger condition of choosing σ to have derivative of absolute value 1.

backpropagated[2].

One way of mitigating the stability problems of Equation (6.1) is to decree B to be orthogonal. This was first noted in practice in (Arjovsky, Shah, and Bengio 2016). After this, optimisation methods over the unitary and orthogonal group have found rather fruitful applications in RNNs. We will do a literature review of these methods in Section 6.3.1. In parallel to this work, there has been an increasing interest in optimisation over the orthogonal group and the Stiefel manifold in neural networks (Ozay and Okatani 2016; L. Huang et al. 2018). As shown in these papers, orthogonal constraints in linear and CNN layers can be rather beneficial for the generalisation of the network as they act as a form of regularisation of the Lipschitz constant of the neural network.

6.1.2 Optimisation with orthogonal constraints

In this section, we will make explicit every detail about how to implement some first-order optimisation algorithms on $\mathrm{SO}(n)$. All the ideas in this section were already presented in an abstract way in Chapter 4, and specialised to the Stiefel manifold in Section 3.5.1, so the computations should already be familiar to the reader. We will flesh out every detail of these computations for $\mathrm{SO}(n)$, as it is the main building block used to implement optimisation in many other different manifolds.

Let $\overline{f}\colon \mathbb{R}^{n\times n} \to \mathbb{R}$ be a function and define $f \coloneqq \overline{f}|_{\mathrm{SO}(n)}$. We are interested in approximating a solution to the problem

$$\operatorname*{arg\,min}_{B\in\mathrm{SO}(n)} f(B).$$

The classical way to approach this problem would be to apply Riemannian gradient descent (RGD). Let A be a skew-symmetric matrix and let $B = \operatorname{expm}(A) \in \mathrm{SO}(n)$. We will fix canonical metric on $\mathbb{R}^{n\times n}$ and the induced bi-invariant metric on $\mathrm{SO}(n)$. For these metrics, we have the gradient on the total space $\nabla\overline{f}(B) \in \mathbb{R}^{n\times n}$ and the gradient on the manifold $\nabla f(B) \in T_B\,\mathrm{SO}(n)$.

RGD works by following the geodesic defined by the direction $-\nabla f(B)$ at the point B for a time $\eta > 0$, that is,

$$B_{t+1} = \exp_{B_t}(-\eta\nabla f(B_t)).$$

The tangent space to $\mathrm{SO}(n)$ at a matrix B is

$$T_B\,\mathrm{SO}(n) = \{X \in \mathbb{R}^{n\times n} \mid B^\intercal X + X^\intercal B = 0\}$$

and, as we showed in Section 3.5.1, the projection onto the tangent space takes the form

$$\begin{aligned}\pi_B\colon \mathbb{R}^{n\times n} &\to T_B\,\mathrm{SO}(n)\\ X &\mapsto \frac{1}{2}(X - BX^\intercal B)\end{aligned}$$

Since $\mathrm{SO}(n)$ is an embedded submanifold of $\mathbb{R}^{n\times n}$, the gradient of f is just the tangent component of the gradient on the ambient space

$$\nabla f(B) = \pi_B(\nabla\overline{f}(B)) = \frac{1}{2}(\nabla\overline{f}(B) - B\nabla\overline{f}(B)^\intercal B).$$

[2]The idea of the gradients "flowing back" during the backpropagation process—while computing the gradients using the chain rule—is often used in the deep learning context to describe that the gradients are computed from the end of the neural network all the way to the beginning of the neural network.

Writing for short $B_t = \operatorname{expm}(A_t)$, we have that the RGD update rule for $\mathrm{SO}(n)$ is given by

$$B_{t+1} = \operatorname{expm}(A_t)\operatorname{expm}(-\eta B_t^\intercal \nabla f(B_t)) = \operatorname{expm}(A_t)\operatorname{expm}(-\eta \pi_{\mathrm{I}_n}(B_t^\intercal \nabla \overline{f}(B_t))).$$

or more succinctly

$$B_{t+1} = \operatorname{expm}(A_t)\operatorname{expm}(-\eta \pi_{\mathrm{I}_n}(\nabla(\overline{f} \circ L_{B_t})(\mathrm{I}_n))).$$

It is now easy to see how this equation is implemented by the autodiff engine when using the dynamic trivialisations. Consider the Riemannian exponential on $\mathrm{SO}(n)$

$$\exp_B = L_B \circ \operatorname{expm} \circ \pi_{\mathrm{I}_n} : \mathbb{R}^{n\times n} \to \mathrm{SO}(n)$$

where we have used left-invariant vector fields to identify $T_B \mathrm{SO}(n)$ with $\mathfrak{so}(n)$ and we have used π_{I_n} to parametrise the skew-symmetric matrices. We then have that

$$\nabla f(B) = \nabla(f \circ \exp_B)(0_n) = \nabla(\overline{f} \circ L_B \circ \operatorname{expm} \circ \pi_{\mathrm{I}_n})(0_n).$$

Remark 6.1.1 (Frames). It is perhaps now easier to see the idea that we mentioned in Section 4.5 about how, after a choice of parametrisation of the tangent space—*i.e.*, a choice of frame $\zeta \colon T_pM \to \mathbb{R}^n$—the dynamic trivialisation framework automates the gradient computations. It should be now clear that different frames change the representation of the tangent space, but they do not change the optimisation algorithm. Here we are using the projection from the total space—that is, a linear Riemannian submersion—given by $\pi_{\mathrm{I}_n} : \mathbb{R}^{n\times n} \to \mathfrak{so}(n)$.

Alternatively, we could have used the frame `frame_skew`: $\mathbb{R}^{n\times n} \to \mathrm{Skew}(n)$ which takes the lower triangular part of a matrix and uses it to form a skew-symmetric matrix $A - A^\intercal$. This frame has the advantage that, even though we are working with matrices in $\mathbb{R}^{n\times n}$ to parametrise skew-symmetric matrices, we are effectively just modifying the lower $\frac{n(n-1)}{2}$ entries. This leaves the upper and diagonal $\frac{n(n+1)}{2}$ entries untouched, and they can be used to make the model more expressive without requiring any extra memory. We use this extensively in some parts of the GeoTorch library, such as when we parametrise low-rank matrices via their SVD using two matrices on Stiefel manifolds. In this case, one of them is parametrised using the upper triangular part of the matrix and the other one uses the lower-triangular part.

Another way to attack the problem of optimisation over $\mathrm{SO}(n)$ is through static trivialisations as described in Section 4.4. As the matrix exponential is surjective onto $\mathrm{SO}(n)$, we can use it as a static trivialisation, pulling back the problem as

$$\min_{B\in\mathrm{SO}(n)} f(B) = \min_{A\in\mathrm{Skew}(n)} f(\operatorname{expm}(A)). \tag{6.4}$$

We call this the **exponential parametrisation of** $\mathrm{SO}(n)$. The update rule, applying gradient descent to this problem, is then given by

$$A_{t+1} = A_t - \eta \nabla(f \circ \operatorname{expm})(A_t)$$
$$B_{t+1} = \operatorname{expm}(A_{t+1}).$$

To initialise this method, we sample a matrix $B_0 \in \mathrm{SO}(n)$ according to some probability measure and let $A_0 \coloneqq \mathrm{logm}(B_0)$.

It is also natural to consider the dynamic trivialisation framework discussed in Section 4.5, having, for a fixed $B_0 \in \mathrm{SO}(n)$, the problem

$$\min_{B \in \mathrm{SO}(n)} f(B) = \min_{A \in \mathrm{Skew}(n)} f(B_0 \exp(A)). \tag{6.5}$$

which has an update rule

$$\begin{aligned} A_{t+1} &= A_t - \eta \nabla (f \circ L_{B_0} \circ \exp)(A_t) \\ B_{t+1} &= B_0 \exp(A_{t+1}). \end{aligned}$$

As mentioned in Section 4.5, we sample B_0 according to a probability measure on $\mathrm{SO}(n)$ and start at the point $A_0 = 0$. Then, we may update B_0 according to some stopping rule. In practice, we compute the gradient of $f \circ L_{B_0} \circ \exp \circ$`frame_skew` which is computed automatically by the autodiff engine.

Remark 6.1.2 (RGD and trivialisations agree on commutative Lie groups). It is quite revealing to put side by side the update rule for RGD and the static trivialisation framework[3]

$$\begin{aligned} \text{RGD:} \qquad & B_{t+1} = \exp(A_t) \exp(-\eta \nabla f(\exp(A_t))) \\ \text{Static Triv. at } I_n\text{:} \qquad & B_{t+1} = \exp(A_t - \eta \nabla (f \circ \exp)(A_t)) \\ \text{Static Triv. at } B_0\text{:} \qquad & B_{t+1} = B_0 \exp(A_t - \eta \nabla (f \circ L_{B_0} \circ \exp)(A_t)). \end{aligned}$$

We can see here that the difference between RGD and the static trivialisation method at the identity stems from the fact that the Lie exponential is not a Lie group homomorphism, that is, in general we have that $\exp(A_1 + A_2) \neq \exp(A_1)\exp(A_2)$. In fact, it is not difficult to show[4] that the Lie exponential is a homomorphism if and only if the Lie group is commutative. Examples of commutative Lie groups are the torus or $\mathbb{R}^n$—in fact, every commutative Lie group is a direct product of these two.

The geometric reason for this equality stems from the fact that a bi-invariant metric on a Lie group is flat if and only if the group is commutative. Therefore, pulling back by the Lie exponential accounts for lifting the problem to its universal cover being the Riemannian exponential the cover map, which is a Riemannian submersion in this case. This behaviour happens more generally on any Hadamard manifold as per the Cartan–Hadamard theorem (*cf.*, Section 4.4.2.1).

Remark 6.1.3 (Different optimisation algorithms). Both the static trivialisation and the dynamic trivialisation pullback the problem to a Euclidean space. As such, we could use any Euclidean optimisation method to solve (6.4) and (6.5). This is a big advantage in practice, given that virtually all the methods developed to improve the stability and convergence of optimisation algorithms for neural networks have been devised having the Euclidean space in mind. We will go over these and other advantages of this method over RGD in the context of neural networks in this next section.

[3] Note that the gradient in RGD is the gradient taken with respect to the metric on the manifold, and the gradient in the trivialisation framework is taken with respect to the flat metric on T_pM.

[4] For example, using the Baker–Campbell–Hausdorff formula.

6.2 GeoTorch

In this section, we go over some implementation details of the library **GeoTorch**. This library allows for performing optimisation on manifolds within PyTorch. This library is built upon a few core ideas:

1. GPU efficiency and parallelism by design
2. Automatic computation of Riemannian gradients and Hessians
3. Functorial design benefiting from code reuse
4. Seamless interoperability with PyTorch

We will show how these ideas are implemented in practice in Section 6.2.2.

6.2.1 Libraries for optimisation on manifolds

We start by reviewing the current available frameworks for performing optimisation on manifolds.

At the time of writing, the most important library to perform constrained optimisation and optimisation on manifolds is **Manopt** (Boumal, Mishra, et al. 2014). This is a MATLAB library maintained by Nicolas Boumal and Bamdev Mishra. This tool is widely used in papers within the field of optimisation on manifolds, being the reference when it comes to theoretical optimisation on manifolds. The main differences between Manopt and GeoTorch stem from the audience the library is targeting. Manopt targets a specialised audience that works on optimisation and may want to compare a new research line with pre-existing optimisation algorithms. In contrast, GeoTorch targets a more mainstream audience that wants to integrate constraints into a pre-existing model to regularise it, without having to change much of their initial model.

As mentioned, Manopt is designed to provide the researcher with a framework to showcase the performance of their algorithms in simple settings. While this approach has been invaluable to the researchers doing optimisation on manifolds for the last 10 years, it is not designed to be used in large models. Manopt has been ported to several other languages in packages like **Manopt.jl** and **PyManopt**, but these ports suffer of similar drawbacks.

GeoTorch comes to fill in this gap. GeoTorch is built on top of PyTorch, the most widely used deep learning library. As such, it takes advantage of the efficiency improvements and GPU, TPU, AMP, distributed training support that PyTorch offers natively, among others. It also supports autodiff natively. All these are systems that GeoTorch is designed to interact with seamlessly.

This integration with PyTorch also heavily simplifies the maintenance required for the library. Not only does one not need to implement the gradients or Hessians, but it also benefits from all the numerical analysis algorithms (QR, SVD, Cholesky...) together with their derivatives implemented natively on GPU by the PyTorch community. We will showcase this idea in Section 6.3.2, where we will note the advantage of a GPU implementation of the matrix exponential, obtaining a $\times 2.5$ wall-clock time improvement over the non-native GPU implementation and a $\times 12$ wall-clock time improvement over the implementation in Tensorflow. This implementation is now in core PyTorch as of PyTorch 1.7.0, and it can be used by any other library or researcher that uses PyTorch.

The first attempt to integrate constrained optimisation together with a deep learning library was **McTorch** (Meghwanshi et al. 2018), lead by Bamdev Mishra. The project was discontinued, as their approach required forking the whole PyTorch project and adding their software on top. We will sidestep this problem by using the dynamic trivialisation framework and a number of metaprogramming tricks.

Geoopt is another package that aims to be compatible with PyTorch. Its implementation relies on the user using their `Tensor` and `Parameter` classes, rather than that of PyTorch. This reason make this library difficult to use for the general audience, as they would have to commit to the use of these custom classes in their models, which is already a very big drawback. For example, if the user already uses a different tensor class in their model, it would be impossible for her to use this library. Even more, the user needs to wrap every parameter of a deep model using these special classes, which can be somewhat tedious and difficult to maintain.

6.2.2 The design of GeoTorch

The main idea driving GeoTorch is to use the static and dynamic trivialisation framework to pullback an optimisation problem from a manifold onto a Euclidean space. As described in Chapter 4, if we have a map $\varphi\colon \overline{M} \to M$, we may use it to pullback the problem from M to $\overline{M}$. In practice, we will often choose a map with domain $\overline{M} \cong \mathbb{R}^n$, such as the Riemannian exponential map or a retraction, to be able to apply Euclidean optimisation methods to the pullback problem. Even then, in this section we will not delve into the different maps used for the different manifolds, as most of these are standard in the literature. In turn, we will look at the ideas necessary for the modular implementation behind GeoTorch.

Before delving into the implementation details, which might at times be too technical, we would like to start by showing the simplicity of GeoTorch in action.

```python
import torch
import torch.nn as nn
import geotorch

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.linear = nn.Linear(64, 128)
        self.cnn = nn.Conv2d(16, 32, 3)
        # Make the linear layer into a low rank layer with rank at most 10
        geotorch.low_rank(self.linear, "weight", rank=10)
        # Also works on tensors. Makes every kernel orthogonal
        geotorch.orthogonal(self.cnn, "weight")


    def forward(self, x):
        # self.linear has rank at most 10 and every 3x3 kernel in the CNN is orthogonal

# Nothing fancy from here on. Use the model as you would normally do.
model = Model()

# Any optimiser works out of the box with any parametrisation
optim = torch.optim.Adam(model.parameters(), lr=0.01)
```

Listing 6.1: Example of the integration of GeoTorch with PyTorch

GeoTorch implements optimisation on 16 spaces in a way that is compatible with any pre-existing PyTorch model and allows for constraining any PyTorch module with just one line of code. These spaces are listed in Table 6.1.

Real numbers	General linear group	Positive definite
Symmetric	Special orthogonal group	Positive semidefinite
Skew-symmetric	Stiefel manifold	Positive semidefinite low rank
Low rank	Grassmannian	Positive semidefinite fixed rank
Fixed rank	Sphere	Product manifolds
	Matrices with singular values in (a, b)	

Table 6.1: List of spaces that GeoTorch currently implements.

We will illustrate how to implement such a simple API using the example of orthogonal optimisation. We will implement a linear layer with orthogonal weights (*cf.*, Definition 2.3.3)

$$f_B(x) = Bx \qquad B \in \mathrm{SO}(n).$$

We will do so by pulling back the problem along the matrix exponential to a Euclidean space as we did for exprnn. We will then point out the flaws that a naïve implementation has and we will motivate the implementation in GeoTorch by correcting these.

Example 6.2.1 (A naïve implementation). Assume that we have the following simplified implementation of a square linear layer

```
class SquareLinear(nn.Module):
    def __init__(self, n_features):
        super().__init__()
        self.weight = nn.Parameter(torch.Tensor(n_features, n_features))

    def forward(self, x):
        return x @ self.weight  # Right-multiply as x is of size [batch, n_features]
```

Listing 6.2: Basic `SquareLinear` layer

We can modify this implementation to make it into an orthogonal linear layer by pulling back the problem by the matrix exponential:

```
1  class Orthogonal(nn.Module):
2      def __init__(self, n_features):
3          super().__init__()
4          self.weight = nn.Parameter(torch.Tensor(n_features, n_features))
5
6      def forward(self, x):
7          A = self.weight
8          A = A - A.T         # A is skew-symmetric
9          B = A.matrix_exp() # B is special orthogonal
10         return x @ B        # Right-multiply as x is of size [batch, n_features]
```

Listing 6.3: `Orthogonal` v.1. Basic implemenation.

Note that here we are using the frame $\pi_{\mathrm{I}_n}(A) = A - A^\intercal$ (*cf.*, Section 6.1.2). If we replaced Line 7 with `A = self.weight.tril()` we would have an implementation of `frame_skew` as described in Remark 6.1.1.

This *ad hoc* implementation is good enough in most scenarios, but it has three main drawbacks:

1. To implement a parametrisation on a layer we effectively need to reimplement the whole layer
2. The parametrisation is implemented in the layer itself. We would have to repeat the implementation of the parametrisation for every layer that we want
3. If we use the layer several times, like one does in an RNN, the matrix exponential would be computed multiple times on the same matrix

We will solve the first and second problem in two steps. First, we will show how solve the first one by using inheritance and **Python properties**. Then, we will show how to use some advanced **metaprogramming** techniques to decouple the implementation of the class from the implementation of the constraints.

The third one calls for the implementation of a caching mechanism, which is somewhat tricky in the context of autodiff systems. We will show a working solution, but we should warn the reader that, even though the final solution looks simple, similar-looking solutions yield problems related to the propagation of gradients. We will not mention these here to avoid making the section even more technical.

6.2.2.1 Reusing pre-existing classes

Let us continue with the example of the `Orthogonal` layer. We will show now how to use inheritance and Python properties to reuse the code from `SquareLinear`.

Python properties are often called **computed attributed**. They allow for using functions to define an attribute of a class. As an example, consider that we have an attribute in a class in centimetres, but we also want to have it in inches. We could implement this as follows:

```
class Length:
    def __init__(centimetres):
        self.centimetres = centimetres

    @property
    def inches(self):
        return self.centimetres / 2.54
l = Length(2.)
print(l.centimetres) # Will print 2.0
print(l.inches)      # Will print 0.787402
```

Properties allow having computed attributes, which is exactly what trivialisations are. It is now easy to modify the code of `Orthogonal` to reuse the code from `SquareLinear`.

```
class Orthogonal(SquareLinear):
    def __init__(n_features):
        super().__init__(n_features)
        # Rename weight attribute to _weight
        self._weight = self.weight
        delattr(self, "weight")

    @property
    def weight(self):
        A = self._weight
        A = A - A.T
        return A.matrix_exp()
```

Listing 6.4: `Orthogonal` v.2. Reuse code through properties and inheritance.

Even better, we can get rid of `SquareLinear` altogether and use the PyTorch implementation `nn.Linear` without having to reimplement it. As such, we can simply implement `Orthogonal` as

```
class Orthogonal(nn.Linear):
    def __init__(n_features):
        super().__init__(n_features, n_features, bias=False)
        # Rename weight attribute to _weight
        self._weight = self.weight
        delattr(self, "weight")

    @property
    def weight(self):
        A = self._weight
        A = A - A.T
```

```
12            return A.matrix_exp()
```

Listing 6.5: `Orthogonal` v.2.1. Use the native `nn.Linear` from PyTorch.

By doing this, whenever we call `Orthogonal.forward`, it will call the `nn.Linear.forward`, which will call `self.weight`, which will call the property and return an orthogonal matrix. Furthermore, all these operations are differentiable, so the gradients are computed automatically by the autodiff engine with no extra work from our side.

Remark 6.2.2 (On the above implementation). The attentive reader will have spotted that the implementation in Listing 6.5 does not work in practice. This is because the parameter `weight` is renamed in `__init__` after the property is defined during the parsing of the class body. A correct implementation of this idea would involve the use of metaclasses. We omit this as metaclasses are quite technical and would not add much to the exposition.

6.2.2.2 Caching tensors

To solve the problem of having to compute the matrix exponential several times, we have to implement a **caching system**. A caching system simply accounts for saving an output of a deterministic function $\varphi(x)$ for certain value x, so that if $\varphi(x)$ has to be recomputed for the same x, we simply return the result. Note that in our case $x = \texttt{self.weight} \in \mathbb{R}^{n\times n}$ and $\varphi = \exp\mathrm{m} \circ \pi_{I_n}$ is the parametrisation of $\mathrm{SO}(n)$ using of skew-symmetric matrices so that $\varphi(x) \in \mathrm{SO}(n)$.

Remark 6.2.3 (A word on automatising the caching system). The caching mechanism has to be aware of two particular moments in the optimisation process: The moment when it has to compute and cache the value, and the moment when it has to discard the result. The first is simple to detect: If we look at the cache and it does not have the value computed, we compute it and store it in the cache. Detecting when the cache is *dirty*—*i.e.*, when does it have to recompute the value in the cache—is much tricky. In particular, it accounts for detecting when the parameter x has been modified.

In optimisation problems, we know that the tensor that we are parametrising is often just going to be updated by the optimiser after a full optimisation step. A model very rarely modifies its own weights. As such, it would be natural to ask for a system that automatically detects this and it automatically discards the cache value after the optimiser has modified the parameters.

On the other hand, if we want to implement this in a generic way, we have to be able to provide a mechanism that works for every possible model. In particular, it should be able to detect the notably rare case when the model modifies its own parameters before the optimisation step. To do this, we would have to be able to detect when `self.weight` has been modified. This poses two problems. First, it would have to check for equality of the full tensor every time the tensor is accessed, which is not only very expensive, but also equality on floating-point numbers is unstable at best. Second, even though in Python it is possible to detect in some situations when a member of a class has been modified, it is not so easy to do so in PyTorch. PyTorch is mostly implemented in C++, so it is close to impossible to differentiate when a member of a class has been modified versus when it has just been accessed. To close this discussion, we quote again the *Zen of Python*:

Explicit is better than implicit.

For all these reasons, we opted for a simpler solution along the lines of similar solutions adopted in core PyTorch. We implement a caching system that can be activated and deactivated by the user in their model via a global flag managed through a context manager. For example, assume that we have a parametrisation in place on an RNN that makes the recurrent kernel orthogonal. The user would then activate the caching system as

```
with geotorch.cached():
    for x in inputs:
        hidden_state = self.rnn(x, hidden_state)
```

This can also be done at a level of the whole model in the training loop (*cf.*, Listing 2.1), by guarding the forward pass

```
for idx, (batch_x, batch_y) in enumerate(loader):
    batch_x, batch_y = batch_x.to(device), batch_y.to(device)

    with geotorch.cached():
        out = model(batch_x)
    loss = F.nll_loss(out, batch_y)

    optim.zero_grad()
    loss.backward()
    optim.step()
```

This guard effectively signals the whole model when should it start caching values and when are the cached values not valid any more.

In order to take advantage of this mechanism, we would have to simply check whether one should use the cache value or not. For example, for the class `Orthogonal` we could write

```
_cache_enabled = False
_cache = {}

class Orthogonal(nn.Linear):
    def __init__(n_features):
        super().__init__(n_features, n_features)
        # Rename weight attribute to _weight
        self._weight = self.weight
        delattr(self, "weight")

    def compute_weight(self):
        A = self._weight
        A = A - A.T
        return A.matrix_exp()

    @property
    def weight(self):
        global _cache
        key = id(self)
        if _cache_enabled:
            if key not in _cache:
                # If we have not cached the value, we compute it and store it
                _cache[key] = self.compute_weight()
            return _cache[key]
        else:
            # If the cache is not enabled, simply return the matrix
            return self.compute_weight()

class cached:
    def __enter__(self):
        global _cache_enabled
        _cache_enabled += 1

    def __exit__(self, exc_type, exc_value, traceback):
```

```
        global _cache_enabled
        _cache_enabled -= 1
        if not _cache_enabled:
            _cache = {}
```

Listing 6.6: `Orthogonal` v.3. Integrating the caching system.

Note that the implementation does not consider `_cache_enabled` to be a boolean but an integer, to be able to handle correctly nested calls to the `cached` context manager.

6.2.2.3 Injecting properties into classes

To approach the second and third problem of reusing code, we resort to metaprogramming. Metaprogramming is the ability of a language of treating code as data. That is, a metafunction could take some parameters and return not an instance of a class, but a whole new class.

In our case, we will use to create the code class `Orthogonal` given a class that just implements the orthogonality code, and the class that we want to put the parametrisation on. We would like to have an `Orthogonal` class that looks exactly like a PyTorch module, that is

```
class Orthogonal(nn.Module):
    def forward(self, A):
        A = A - A.T
        return A.matrix_exp()
```

Listing 6.7: A class that implements the orthogonal parametrisation

The desideratum is to have a *metafunction* that takes an instance of this class and an instance of `nn.Linear` and returns an instance of a class that looks like the `Orthogonal` layer from Listing 6.6.

Even though this can be implemented in about 50 lines of code, the details are very technical. We will summarise them here for the interested reader, but they may be regarded as a convenient implementation detail for the general reader.

In order to implement the caching system and the parametrisation system automatically, we would have to inject a property into a given object. Sadly, this is not possible in Python, as properties are stored at a class level, not at an instance level. One way of overcoming this issue is to dynamically create a new class that inherits from `nn.Linear`, inject the property in this class and then replace the class from our initial object with this custom class. This trick was suggested to us by Adam Paszke, and it is implemented in the following piece of code:

```
def _inject_new_class(module):
    r"""Sets up the parametrization mechanism used by parametrizations.
    This works by substituting the class of the module by a class
    that extends it to be able to inject a property

    Args:
        module (nn.Module): module on which to inject the property
    """

    # We create a new class so that we can inject properties in it
    cls_name = "Parametrized" + module.__class__.__name__

    param_cls = type(
        cls_name,
        (module.__class__,),
        {
            "__qualname__": cls_name + str(id(module)),
        },
    )
```

```
20
21      # Declare the class globally to be able to pickle it
22      globals()[param_cls.__qualname__] = param_cls
23      module.__class__ = param_cls
```

6.2.2.4 Initialising parametrisations

Once we have a way to change the class to one that is virtually identical but different, we may simply inject the property. While doing so, we take the chance of defining not only a getter for the property, but also a setter.

```
1  def _inject_property(module, _name):
2      r"""Injects a property into the class of a given module.
3      It assumes that the tensor under ``module[tensor_name]``
4      has already been moved
5
6      Args:
7          module (nn.Module): module on which to inject the property
8          tensor_name (string): name of the property
9      """
10
11     # Define the getter
12     def get_parametrized(module):
13         global _cache_enabled
14         global _cache
15
16         key = _key(module, tensor_name)
17         # If the _cache is not enabled or the caching was not enabled for this
18         # tensor, this function just evaluates the parametrization
19         if _cache_enabled and key in _cache:
20             if _cache[key] is None:
21                 _cache[key] = module.parametrizations[tensor_name].evaluate()
22             return _cache[key]
23         else:
24             return module.parametrizations[tensor_name].evaluate()
25
26     # Define the setter
27     def set_value(module, value):
28         module.parametrizations[tensor_name].initialize_(value)
29
30     setattr(module.__class__, tensor_name, property(get_parametrized, set_value))
```

This allows to assign to the properties provided that we implement an `initialize_` method.

```
1  class Skew(nn.Module):
2      def forward(self, X):
3          X = X.triu(1)
4          return X - X.T
5
6      def is_skew(self, X):
7          # Skew modulo rounding errors
8          return torch.allclose(X, -X.T)
9
10     def initialize_(self, X):
11         if not self.is_skew(X):
12             raise ValueError("The input matrix is not skew-symmetric")
13         return X.triu(1)
14
15 model = nn.Linear(5, 5)
16 geotorch.register_parametrization(model, "weight", Skew())
17 # Just computes `model.weight` and checks that the weight is and checks that the weight
       is skew-symmetric
18 with geotorch.cached():
19     assert(torch.allclose(model.weight, -model.weight.T))
20 # Sample a skew matrix X and initialise the parametrised model.weight
21 X = torch.rand(5, 5)
22 model.weight = X - X.T
23 assert(torch.allclose(model.weight, X))
```

In most cases, the method `initialize_` is simply a right-inverse of the `forward` method, which exists locally whenever `forward` is a submersion. It is at this point where the theory and practice meet to give a particularly modular design for the static and dynamic trivialisation framework.

We are missing to show how to put all these ideas together to allow composing parametrisations—*i.e.*, calling several times the method `register_parametrization` on the same tensor—and also finish the implementation of `register_parametrization` in terms of the auxiliary methods presented here. We spare the reader all those details, as they are a matter of a few lines of software engineering, and they are not particularly interesting. A fully working implementation can be found in

https://github.com/Lezcano/geotorch/blob/master/geotorch/parametrize.py

All these ideas have been submitted to be added to core PyTorch, and it is currently being discussed in

https://github.com/pytorch/pytorch/pull/33344

It is hoped to be accepted and merged into core PyTorch in the next release (1.9.0).

6.2.2.5 Class system

In this section, we will touch on the class system designed for GeoTorch. The idea here is to abstract most of the patterns used in the context of optimisation on manifolds into Python constructions.

Before starting with the explanation, it is worth quickly summarising what we have accomplished so far: We have a way of injecting a function to parametrise a tensor in our model.

This may not seem like too much on its own, but it is truly powerful tool due two main reasons and a third one that allows to put them together.

1. The whole PyTorch package is designed around being able to define differentiable functions as classes inheriting from `nn.Module`.

2. We have detailed in Chapters 3 and 4 how fibred manifolds provide a compositional way to pullback problems via pre-composing by frames and post-composing by submersions.

3. Python's inheritance system is designed around pre and post-composition of functions (*decoration of methods*).

We shall now show how to put these three ideas together to implement a class system on manifolds that uses optimisation simple manifolds to perform optimisation on more complex ones. We will do it again continuing our example of orthogonal optimisation, extending it to non-square orthogonal optimisation.

Example 6.2.4 (Stiefel manifold). At the moment, we have the following implementation of a map from $\mathbb{R}^{n\times n}$ to $\mathrm{SO}(n)$ via the Riemannian exponential map, which can be injected into any parameter of a model.

```
class SO(nn.Module):
    def __init__(self, n):
        super().__init__()
        # Initialise self.base to a matrix sampled according to the Haar measure on SO(n)
        self.register_buffer("base", nn.init.orthogonal_(torch.empty(n, n)))

```

```
    def frame(self, X):
        X = X.tril(-1)
        return X - X.T

    def exponential(self, B, A):
        return B @ torch.matrix_exp(A)

    def forward(self, X):
        A = self.frame(X)
        return self.exponential(self.base, A)
```

Listing 6.8: Final version of the parametrisation of $\mathrm{SO}(n)$

Note that the map has two separate maps: A linear frame, and the computation of the Riemannian exponential. Denote the frame that we have used to map the lower-triangular part of a matrix onto a skew-symmetric matrix as

$$\texttt{frame_skew}\colon \mathbb{R}^{n\times n} \to \mathrm{Skew}(n)$$

The `SO` class defined in Listing 6.8 can then be defined mathematically denoting $\overline{Q} = \texttt{self.base}$ as

$$L_{\overline{Q}} \circ \mathrm{expm} \circ \texttt{frame_skew}\colon \mathbb{R}^{n\times n} \to \mathrm{SO}(n)$$

Let us now implement optimisation on $\mathrm{St}(n,k)$ based on this. Recall that the tangent space of the Stiefel manifold at a point $Q \in \mathrm{St}(n,k)$ may be identified with $\mathfrak{m} \subseteq \mathfrak{so}(n) \cong \mathrm{Skew}(n)$ (Section 3.5.1), where $\mathfrak{m}$ is given by

$$\mathfrak{m} = \left\{ \begin{pmatrix} S & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} \;\middle|\; S \in \mathfrak{so}(k), A \in \mathbb{R}^{(n-k)\times k} \right\}.$$

To implement optimisation on $\mathrm{St}(n,k)$, we just have to embed a matrix $\mathbb{R}^{n\times k}$ into $\mathfrak{m}$. We may do this by simply considering the map $\iota\colon \mathbb{R}^{n\times k} \to \mathbb{R}^{n\times n}$ that given a matrix appends an $n \times (n-k)$ matrix of zeros to it. If we then denote by $\pi\colon \mathrm{SO}(n) \to \mathrm{St}(n,k)$ the projection of a matrix onto its first k columns, we have that we may implement optimisation on $\mathrm{St}(n,k)$ as

$$\exp_U^{\mathrm{St}(n,k)} = \pi \circ \exp_{\overline{U}}^{\mathrm{SO}(n)} \circ \iota = \pi \circ L_{\overline{U}} \circ \mathrm{expm} \circ \texttt{frame_skew} \circ \iota.$$

In particular, we may reuse all the implementation of $\mathrm{SO}(n)$ to implement the exponential on $\mathrm{St}(n,k)$ and simply implement ι and π.

```
class Stiefel(SO):
    def __init__(self, n, k):
        super().__init__(n)
        self.n = n
        self.k = k

    def frame(self, X):
        # X \in R^{n x k}
        # We embed X into R^{n x n} by forming Y = [X, 0] \in R^{n x n}
        Y = torch.cat([X, X.new_zeros(self.n, self.n - self.k)], dim=1)
        return super().frame(Y)

    def exponential(self, B, A):
        U_total = super().exponential(B, A)
        # Project onto the first k components
        return U_total[:, :self.k]
```

Listing 6.9: Stiefel manifold from $\mathrm{SO}(n)$

Furthermore, the Riemannian gradients at $U = \pi(\overline{U})$ may be obtained by letting the autodiff engine differentiate the mapping at zero, as we have previously mentioned.

Example 6.2.5 (Grassmannian). It should be clear that there is nothing special about the Stiefel manifold in this construction. For example, we could go one step further and implement the exponential on the Grassmannian manifold in terms of that of the Stiefel manifold in just two lines. We start by recalling that, for the Grassmannian

$$\mathfrak{m} = \left\{ \begin{pmatrix} 0_{k,k} & -A^\intercal \\ A & 0_{n-k,n-k} \end{pmatrix} \;\middle|\; A \in \mathbb{R}^{(n-k)\times k} \right\},$$

that is, it is the same set as that of the Stiefel manifold but with the top square zeroed-out. As such, to parametrise the Grassmannian in terms of the Stiefel manifold, we simply have to implement the embedding ι_2 that, given a matrix in $\mathbb{R}^{n\times k}$, zeroes out its first k rows, so that $\iota \circ \iota_2$ is a map from $\mathbb{R}^{n\times k}$ onto the set $\mathfrak{m}$ from the Grassmannian. This translates to the code:

```
class Grassmannian(Stiefel):
    def frame(self, X):
        # X \in R^{n x k}, we zero-out its first k rows
        Y = torch.cat([X.new_zeros(self.k, self.k) , X[self.k:, :]], dim=0)
        return super().frame(Y)
```

Listing 6.10: Grassmannian manifold from $\mathrm{St}(n, k)$

Note that these three lines of code provide a complete implementation of the Riemannian exponential map on the Grassmannian.

More generally, this idea of composing on the right by a linear immersion and composing on the left by a map—preferably a submersion—is a very flexible one. Consider now that we have implemented the Stiefel manifold and optimisation on $\mathbb{R}_{>0}$ (which can be done by pulling back the problem to $\mathbb{R}$). We can then form the submersion

$$\begin{aligned} \pi\colon \mathrm{SO}(n) \times \mathbb{R}^n_{>0} \times \mathrm{SO}(n) &\to \mathrm{GL}^+(n) \\ U, \Sigma, V &\mapsto U\Sigma V^\intercal \end{aligned}$$

The immersion for this map would simply be the map that splits a matrix into its strictly lower-triangular, strictly upper triangular and diagonal parts, while the submersion is the multiplication map.

This pattern of using fibred spaces together with product spaces allows for a rather compact implementation of optimisation on virtually any manifold. We implement a class that, given a list of manifolds, creates a product manifold. This is as simple as writing the following functor

```
class ProductManifold(nn.ModuleList):
    def forward(self, Xs):
        return tuple(mani(X) for mani, X in zip(self, Xs))
```

Listing 6.11: Implementation of a product manifold

Through the abstraction of a `ProductManifold`, Python's inheritance, and the mathematical description of the Riemannian exponential and submersions as a concatenation of linear immersions and submersions, we were able to keep the implementation of every manifold to less than 20 lines on average with no code duplication. All the manifolds support the dynamic trivialisation framework so, in particular, they implement static trivialisations and RGD with no extra work.

6.3 Exponential Recurrent Neural Networks (EXPRNN / DTRIV)

Given a sequence of inputs $(x_1, \dots, x_T)$ with $x_i \in \mathbb{R}^k$, we define an orthogonal exponential RNN (EXPRNN) with hidden size $d > 0$ as

$$h_t = \sigma(\exp(A)h_{t-1} + Cx_t), \qquad t = 1, \dots, T$$

where $A \in \mathrm{Skew}(d)$, $C \in \mathbb{R}^{d \times k}$ are parameters of the model and σ is a non-linearity. This equation is exactly that of a regular RNN (Definition 2.3.4) using the matrix exponential to impose orthogonality constraints on the recurrent kernel.

We also define the dynamic trivialisation architecture (DTRIV), which rather than using the Lie exponential uses the Riemannian exponential to pullback the problem

$$h_t = \sigma(B\exp(A)h_{t-1} + Cx_t), \qquad t = 1, \dots, T$$

where $B \in \mathrm{SO}(n)$ is a fixed matrix—we do not optimise B. In general, we write DTRIVK for $K \in \{1, 2, \dots\} \cup \{\infty\}$ for the case where we update base B after K steps. This is an instance of the dynamic trivialisation framework (Algorithm 1) with stopping rule $\texttt{stop} \equiv k = K$. For the case DTRIV$\infty$, we recover the rule $\texttt{stop} \equiv \texttt{False}$, that is, DTRIV$\infty$ is effectively a static-trivialisation along $\exp_{B_0}$ for a fixed matrix $B_0 \in \mathrm{SO}(n)$. In particular, if $B_0 = \mathrm{I}_n$, DTRIV∞ recovers EXPRNN.

Note that generalising this architecture to the complex unitary case simply accounts for considering A to be skew-Hermitian rather than skew-symmetric and letting $B \in \mathrm{U}(n)$. Note that this same pattern can be used on any naturally reductive homogeneous space, lifting the problem to the linear space $\mathfrak{m}$ of the Lie algebra of the total space.

This split the study of these models in three parts which will be treated in three different sections:

1. The approximation of expm
2. The choice of non-linearity σ
3. The initialisation of the model

But before delving into these, we start by reviewing the previous approaches used in literature to enforce exact orthogonal constraints in this kind of models.

6.3.1 Comparison with previous approaches

A number of approaches have been presented to perform optimisation with orthogonal and unitary constraints in the context of RNNs previous to the introduction of the static and dynamic trivialisation framework. These approaches can be separated into two main categories: Those that use RGD and those that parametrise the special orthogonal group using unconstrained parameters. The latter ones are pullback problems along maps from $\mathbb{R}^k$ for some $k > 0$. As there are no submersions between $\mathbb{R}^{\frac{n(n-1)}{2}}$ and $\mathrm{SO}(n)$—since this map would be a covering map, and the universal cover of $\mathrm{SO}(n)$ is not trivial—we know that such maps will have singularities or will not be surjective. We will comment on these properties for some of these algorithms.

6.3.1.1 Riemannian Gradient Descent (RGD)

This approach was used in the papers (Wisdom et al. 2016; Vorontsov et al. 2017). Recall that the update step for RGD in $\mathrm{SO}(n)$ with step-size $\eta > 0$ and a map $\varphi\colon \mathfrak{so}(n) \to \mathrm{SO}(n)$ is given by

$$B_{t+1} = B_t \varphi(-\eta B_t^\intercal \nabla f(B_t)),$$

that is, we use the Lie group structure to construct the step map $r_B \coloneqq L_B \circ \varphi \circ (\mathrm{d}L_B)^*$. If $(\mathrm{d}\varphi)_0 = \mathrm{Id}$, then r_B is a retraction

In these two papers, they perform this update step using a variation of the Cayley map as the retraction

$$\begin{aligned} \varphi\colon \mathfrak{so}(n) &\to \mathrm{SO}(n) \\ A &\mapsto \frac{\mathrm{I}_n + \frac{1}{2}A}{\mathrm{I}_n - \frac{1}{2}A}. \end{aligned}$$

RGD presents the issue of needing a step map that lies exactly on the manifold. In the case of $\mathrm{SO}(n)$, we need a function φ whose output lies exactly on the manifold. Our static-trivialisation approach does not require this, allowing for the use of approximations to the matrix exponential on $\mathrm{Skew}(n)$ whose result is not orthogonal, like truncating the Taylor series of the matrix exponential at a low degree. These approaches could be useful when the efficiency of the parametrisation is more important in comparison to the orthogonality of the resulting matrix.

A related problem of RGD arises from the numerical approximations inherently present in numerical algorithms. The Cayley map—and sometimes the matrix exponential—is often computed as a solution of a system of the form $BX = C$. For example, in the case of the Cayley map, we set $C = \mathrm{I}_n + \frac{1}{2}A$, $B = \mathrm{I}_n - \frac{1}{2}A$. These systems are solved via an LU decomposition and the solution of two triangular systems, which gives an approximate solution. As such, the resulting matrix is not exactly orthogonal due to numerical and rounding errors. To account for this loss of orthogonality, implementations of RGD perform a projection onto $\mathrm{SO}(n)$ after certain number of iterations. The orthogonal projection onto $\mathrm{SO}(n)$ for a matrix $B \in \mathbb{R}^{n\times n}$ is given by a projection onto the orthogonal component of its polar decomposition. This is sometimes implemented through the SVD decomposition $B = U\Sigma V^\intercal$ as

$$\pi_{\mathrm{SO}(n)}(B) = UV^\intercal.$$

Leaving aside the fact that this is a fairly expensive operation on a GPU, this projected gradient descent is problematic when doing non-convex optimisation. Recall that RNNs are particularly bad-behaved functions, as discussed in Section 6.1.1. As such, the projection step often results on a great increase on the value of the objective function, often precluding the convergence of the algorithm.

6.3.1.2 Parametrisations

There are four main parametrisations of the orthogonal and unitary group presented in the literature that have been used to maintain the orthogonality of layers within a neural network.

Unitary RNN (URNN) (Arjovsky, Shah, and Bengio 2016) This was the first paper to introduce unitary constraints to tackle the vanishing and exploding gradient problem in RNNs. It parametrises $\mathrm{U}(n)$ using the following product of matrices

$$B = D_3 T_2 \mathcal{F}^{-1} D_2 \Pi T_1 \mathcal{F} D_1.$$

$D_k = \operatorname{diag}(e^{i\omega_k})$ where $\omega_k \in \mathbb{R}^n$ are parameters. $T_k = \mathrm{I}_n - 2\frac{v_k \otimes v_k}{\|v_k\|^2}$ where $v_k \in \mathbb{R}^n$ are parameters. $\mathcal{F}$ and $\mathcal{F}^{-1}$ are Fourier transform and inverse Fourier transform matrices. Π is a fixed permutation matrix. The final matrix B is unitary since it is a product of unitary matrices. One of the problems of this parametrisation is that it cannot represent every unitary matrix as $\dim_{\mathbb{R}}(\mathrm{U}(n)) = n^2$, while the model has just $5n$ parameters.

Efficient Unitary NN (EUNN) (Jing et al. 2017) This paper tried to address this surjectivity problem that URNN presents by proposing a different parametrisation. Their parametrisation is based on the decomposition of a unitary matrix via Givens rotations

$$B = D \prod_{i=2}^{N} \prod_{j=1}^{i-1} R_{i,j},$$

where $R_{i,j}$ are Givens rotation matrices and $D = \operatorname{diag}(e^{i\omega})$ for a parameter $\omega \in \mathbb{R}^n$. They propose an efficient computation of this decomposition using ideas related to the fast Fourier transform (FFT), having B as a product of $\log(n)$ matrices. They do not provide any guarantee for the critical points of this parametrisation.

Householder reflections (Mhammedi et al. 2017) In this case, the authors propose to parametrise $\mathrm{O}(n)$ by using a product of n Householder reflections

$$B = \prod_{i=1}^{n} \Big[\mathrm{I}_n - 2\frac{v_i \otimes v_i}{\|v_i\|^2} \Big].$$

This is a surjective parametrisation of the matrices in $\mathrm{O}(n)$ with determinant $(-1)^n$. On the other hand, it presents a number of problems.

Given that this parametrisation is not locally unique, it will inherently have critical points on those points. For example, this happens at points where $v_i = v_j$ for $i \neq j$, or more generally, at points at which the matrix $V = (v_1/\|v_1\|, \dots, v_n/\|v_n\|)$ is singular.

Another problem that this parametrisation has is that it is not defined at $v_i = 0$. This may induce unstable gradients around those points. We believe that this is the reason why different papers have reported that they were not able to reproduce the results of the experiments presented in this paper. We also observed that this parametrisation does not work well in practice. As such, we will not include it in the comparison in the experiments.

Remark 6.3.1 (Parametrisations as long products of matrices). These three parametrisations have a problem in common: They are all given by a product of matrices. For this reason, whenever the parameters are updated, one has to compute the product of matrices sequentially, which is rather costly in a GPU. This is a problem that our parametrisation does not present.

Remark 6.3.2 (Complex parametrisations). Some of these parametrisations only work in the complex case, like those involving the FFT. This created the artificial burden of having to implement complex-valued layers. At the moment of this writing, the support for complex valued networks in PyTorch is rather limited, so putting this extraneous constraints in the model may heavily hinder the adoption of these parametrisations.

Scaled Orthogonal / Unitary RNN (SCORNN / SCURNN) (Helfrich, Willmott, and Ye 2018; Maduranga, Helfrich, and Ye 2019) These parametrisations are the closest in spirit to ours. They use the Cayley transform as a map between $\mathfrak{so}(n)$ and $\mathrm{SO}(n)$ and pullback the problem to $\mathfrak{so}(n)$ (resp. $\mathfrak{u}(n)$ and $\mathrm{U}(n)$). The problem with this map is that orthogonal matrices which have -1 as an eigenvalue are not covered by the map. On the other hand, this does not seem to be particularly problematic in practice.

Even then, they implement certain overparametrisation trick in the papers to deal with this case, although it is not clear whether the performance gain that they get comes from the soundness of their trick or the fact that, because of this trick, their model has some extra structure and is overparametrised.

Matrix Exponential (Hyland and Rätsch 2017) This paper presented a basic version of the idea of using the matrix exponential to map $\mathfrak{u}(n)$ surjectively onto $\mathrm{U}(n)$. On the other hand, the authors write:

> The matrix exponential appearing in Equation 7 poses an issue for gradient calculations. In general, the derivative of the matrix exponential does not have a closed-form expression, so computing gradients is intractable.

It turns out that this claim is not accurate, as we will show in the following section.

6.3.2 The exponential map on a GPU

Most of both the theoretical and practical work in this thesis uses the matrix exponential as the main building block. In this section, we will look at how to efficiently implement this map on a GPU.

There are a myriad of methods to approximate the exponential of a matrix (Moler and Van Loan 1978; Moler and Van Loan 2003). Some of the most important ones are the following:

Diagonal Padé approximants. Diagonal Padé approximants, or simply Padé approximants, are rational approximations of the form $\exp(A) \approx p_n(A)q_n(A)^{-1}$ for polynomials p_n, q_n of degree n. A Padé approximant of degree n agrees with the Taylor expansion of the exponential to degree $2n$. It is easy to show that the diagonal Padé approximant of the exponential map applied to a skew-symmetric matrix is always orthogonal. In particular, the Padé approximant of degree 1 is the Cayley map. These methods and their implementations are described in detail in (Higham 2009).

Taylor approximants. A Taylor approximant simply accounts for truncating the series defining the matrix exponential at a given term. These methods had been historically disregarded in the literature in favour of the Padé approximants as on CPU, they are slower than their Padé counterparts.

Scale-squaring trick. The error of the Padé approximant scales as $\mathcal{O}(\|A\|^{2n+1})$. If $\|A\| > 1$ and we have an approximant ψ, the scale-squaring trick accounts for computing $\psi(\frac{A}{2^k})^{2^k}$ for the first $k = 0, 1, \dots$ such that $\frac{\|A\|}{2^k} < \frac{1}{2}$ to lower the norm of the matrix fed into the approximant. Most types of approximants, like Padé's or Taylor's, can be coupled with the scale-squaring trick to reduce the error (Higham 2009).

Machine-precision approximant. Combining one of the approximants together with the scale-squaring trick, and bounding the error, it is possible to get an efficient approximation of the exponential to machine-precision. The one implemented in most standard numerical libraries is based on the paper (Al-Mohy and Higham 2009). It accounts for an efficient use of the scale-squaring trick and a Padé approximant, in the sense of trying to compute the lowest degree Padé approximant such that it gives an error less than the machine-precision.

On the other hand, we note that the fact that the Padé approximant is faster than the Taylor approximant due to its lower degree is no longer true in the context of GPUs. On a GPU, solving a system of the form $BX = C$ with $B = q_n(A)$ and $C = p_n(A)$ is much slower than performing a few more matrix multiplications. This idea is leveraged in the paper (Bader, Blanes, and Casas 2019). The authors factor the Taylor approximants to compute them using the smallest possible number of multiplications. By doing so, together with a careful backwards analysis of the algorithm, they are able to provide an algorithm that needs of at most 5 matrix multiplications to compute any approximant.

We implemented this algorithm in PyTorch, and we were able to get a $\times 6$ speed-up in matrices of size 1024×1024 over the Padé approximant. We also helped Nikitaved, a PyTorch core developer, to implement this algorithm natively on CUDA, obtaining a further $\times 2.5$ speed improvement, totalling a $\times 14.5$ speed-up against the Padé approximant.

We compare in Tables 6.2 and 6.3 the efficiency of the Padé approximant on GPU compared with an optimised implementation of the algorithm from (Bader, Blanes, and Casas 2019). We can see that, on sizes of 1024×1024 we get a speed-up of $\times 14.5$ with respect to the Padé implementation.

This implementation is even faster than the usual retraction used on $\mathrm{SO}(n)$, the Cayley map, as we can see in Table 6.4. We see that for larger sizes, the matrix exponential is still faster than the Cayley map. We see this even though the LU decomposition needed for the Cayley map is implemented entirely in Magma which handles batching in parallel, while the exponential map simply handles the batch sequentially.

For reference, all the tests were implemented in the PyTorch benchmark framework to ensure the accuracy of the results.[5]

Machine-precision gradients. These algorithms give an approximation to the matrix exponential, but we still need to approximate its adjoint. We showed that this accounts for computing the differential of expm: $\mathfrak{gl}(n) \to \mathrm{GL}^+(n)$ at the transpose (Theorem 4.4.12). To finish the implementation, we use the following theorem to approximate the differential of an analytic matrix function.

[5] https://pytorch.org/docs/master/benchmark_utils.html

Batch	64×64	256×256	1024×1024
16	0.7	0.6	11.1
32	0.7	1.0	22.2
64	0.7	1.7	44.3
128	0.7	3.1	88.9

Table 6.2: Machine-precision matrix exponential on GPU. Used the current implementation in PyTorch 1.7.0. Times in milliseconds.

Batch	64×64	256×256	1024×1024
16	1.4	4.0	195.0
32	1.4	6.0	347.0
64	1.5	10.6	557.0
128	1.5	24.0	1288.0

Table 6.3: Padé approximant on GPU. Used the current implementation in Tensorflow 2.3.0. Times in milliseconds.

Batch	64×64	256×256	1024×1024
16	0.5	2.5	24.1
32	0.5	2.8	34.6
64	0.5	3.6	55.6
128	0.6	5.7	100.0

Table 6.4: Cayley map on GPU implemented in PyTorch 1.7.0. Times in milliseconds.

Theorem 6.3.3 (Differential of a Matrix Function (Mathias 1996)). *Let $U \subseteq \mathbb{C}$ be open and let $\psi\colon U \to \mathbb{C}$ be an analytic function. Let $X \in \mathbb{C}$ be a matrix with spectrum contained in U then, for any $A \in \mathbb{C}^{n\times n}$, denoting by the same letter the associated matrix function, we have that*

$$\psi \begin{pmatrix} X & A \\ 0 & X \end{pmatrix} = \begin{pmatrix} \psi(X) & (\mathrm{d}\psi)_X(A) \\ 0 & \psi(X) \end{pmatrix}.$$

From Theorems 4.4.12 and 6.3.3 we have that, if we know how to approximate an analytic matrix function, such as the matrix exponential, we can approximate its adjoint on matrices of size $n \times n$ by computing the value of the function on a $2n \times 2n$ matrix.

This way, we can use the fast matrix exponential implementation to implement its adjoint. We helped to implement the matrix exponential in PyTorch. It is now part of PyTorch 1.7.0, so all these ideas are open to be used by the machine learning community.

6.3.3 Non-linearities

As we already pointed out in Section 6.1.1, the non-linearity used in the recurrent neural network has a big influence on the vanishing-gradient problems. We see the interaction of the non-linearity with the gradient when we compute the differential of the loss function with respect to the hidden states

$$\frac{\partial \ell}{\partial h_i} = \frac{\partial \ell}{\partial h_T} \prod_{t=i}^{T-1} D_{t+1} B.$$

where $D_t = \operatorname{diag}(\sigma'(z_t))$ and $z_{t+1} = Bh_t + Cx_{t+1}$. Given that in our case B is orthogonal, there is a trade-off between the deviation of σ from a function with $|\sigma'(x)| = 1$ and some possible exploding or vanishing gradient problems suffered by the RNN.

In the simplest case, we could choose $\sigma(x) = x$. The resulting RNN works surprisingly well for many examples, but falls short when used in more complex experiments, like the TIMIT experiment presented in Section 6.4.

In the paper that introduced the unitary RNN (Arjovsky, Shah, and Bengio 2016), the authors introduced the `modrelu` non-linearity

$$\begin{aligned} \texttt{modrelu}_b \colon \mathbb{C} &\to \mathbb{C} \\ z &\mapsto \frac{z}{|z|}\texttt{relu}(|z| + b) \end{aligned}$$

where $\texttt{relu}(x) := \max(x, 0)$ is the rectified linear unit and $b \in \mathbb{R}^n$ is a parameter. This is the non-linearity that was used for this model in all the other papers mentioned before. As such, it will be the one that we will use in our experiments.

Even then, we also propose a novel non-linearity that explicitly models the trade-off between being close to the identity and expressiveness. We define the **dynamic soft shrink** function as

$$\texttt{dynsoftshrink}_{a,b}(x) = \begin{cases} x + b(a-1) & \text{for } x > b \\ ax & \text{for } x \in [-b, b] \\ x - b(a-1) & \text{for } x < -b. \end{cases}$$

for two parameters $a \in \mathbb{R}$, $b \in \mathbb{R}_{\geq 0}$. In other words, this non-linearity is continuous, piecewise linear, and has constant derivative 1 outside of the interval $[-b, b]$ and derivative a inside of this interval. We initialise it to $b = 0.5$, $a = 1$, so that initially $\texttt{dynsoftshrink}_{1,0.5}(x) = x$.

Even though we will not show experiments with this non-linearity in Section 6.4, as the purpose of that section is to compare the orthogonality optimisation process, we report here that have seen large improvements when using this non-linearity instead of the `modrelu`. That being said, we leave the design of better recurrent models for further research.

6.3.4 Initialisation

For the initialisation of the layer with a matrix $A_0 \in \text{Skew}(p)$, we drew ideas from (Henaff, Szlam, and LeCun 2016). The initialisations considered sample blocks of the form

$$\begin{pmatrix} 0 & -s_i \\ s_i & 0 \end{pmatrix}.$$

for s_i i.i.d. with respect to some probability measure on $[-\pi, \pi]$ and then form A_0 as a block-diagonal skew-symmetric matrix with these blocks.[6]

The uniform initialisation accounts for sampling $s_i \sim \mathcal{U}[-\pi, \pi]$. This defines a block-diagonal orthogonal matrix $\text{expm}(A)$ with uniformly distributed blocks on half of the torus of block-diagonal 2×2 rotations.

Initialising the matrix on the main torus seemed to help convergence when compared with other less spare measures such as the Haar measure and others. This was already noted in (Henaff, Szlam, and LeCun 2016). While we do not have a theoretical explanation for this phenomenon, we hypothesise that such initialisation, that acts locally in two coordinates at a time, couples well with the element-wise non-linearities that are present in the RNN.

We chose $h_0 = 0$ as the initial hidden vector of the RNN for simplicity, as we did not observe any empirical improvement when using the initialisation given in (Arjovsky, Shah, and Bengio 2016).

[6] Recall that $\exp(2\pi A) = \exp(A)$ for $A \in \text{Skew}(n)$ since the eigenvalues of a skew-symmetric matrix are purely imaginary.

6.4 Experiments

In this section, we assess the effectiveness of the exponential RNN (EXPRNN) and dynamic trivialisations using the matrix exponential (DTRIVK) in the context of orthogonal optimisation in RNNs. We test the dynamic trivialisations with the basis changed every $K = 1, 100, \infty$ steps.

We compare the performance of our parametrisation for orthogonal RNNs with the following approaches:

- Long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997).
- Unitary Recurrent Neural Networks (URNN) (Arjovsky, Shah, and Bengio 2016).
- Efficient Unitary Recurrent Neural Network (EUNN) (Jing et al. 2017).
- Real Cayley Parametrisation (SCORNN) (Helfrich, Willmott, and Ye 2018).
- Complex Cayley Parametrisation (SCURNN) (Maduranga, Helfrich, and Ye 2019).
- Riemannian Gradient Descent (RGD) (Wisdom et al. 2016).

Remark 6.4.1. Note that (stochastic) RGD is equivalent to DTRIV1 together with the optimiser SGD. Furthermore, EXPRNN is equivalent DTRIV∞ with $B = \mathrm{I}_n$ and $\varphi = \exp$m, while DTRIV∞ has the base at a randomly sampled matrix. At the same time, SCORNN and SCURNN fall into the context of dynamic trivialisations, but using $B = \mathrm{I}_n$ and the Cayley map as a retraction.

We use three tasks that have become standard to measure the performance of RNNs and their ability to deal with long-term dependencies. These are the copying memory task, the pixel-permuted MNIST task, and the speech prediction on the TIMIT dataset (Arjovsky, Shah, and Bengio 2016; Wisdom et al. 2016; Henaff, Szlam, and LeCun 2016; Mhammedi et al. 2017; Helfrich, Willmott, and Ye 2018).

Remark 6.4.2. We found empirically that having a learning rate for the orthogonal parameters that is between 5–10 times larger than that of the non-orthogonal parameters yields a good performance in practice.

For the other experiments, we executed the code that the other authors provided with the best hyperparameters that they reported and a batch of 128. The results for EUNN are those reported in (Jing et al. 2017), and for RGD and URNN are those reported in (Helfrich, Willmott, and Ye 2018).

The code with the exact configuration and seeds to replicate these results can be found here:

https://github.com/Lezcano/expRNN

6.4.1 Copying memory task

The copying memory task was first proposed in (Hochreiter and Schmidhuber 1997). The task can be defined as follows. Let $\mathcal{A} = \{a_i\}_{i=1}^{A}$ be an alphabet and let `<blank>`, `<start>` be two symbols not contained in $\mathcal{A}$. For a sequence length of S and a spacing of length L, the input sequence would be S ordered characters $(b_1, b_2, \dots, b_S)$ sampled i.i.d. uniformly at random from $\mathcal{A}$, followed by L repetitions of the character `<blank>`, the character `<start>` and finally $S - 1$ repetitions of the character `<blank>` again. The output for this sequence would be $S + L$ times the `<blank>` character and then the sequence of

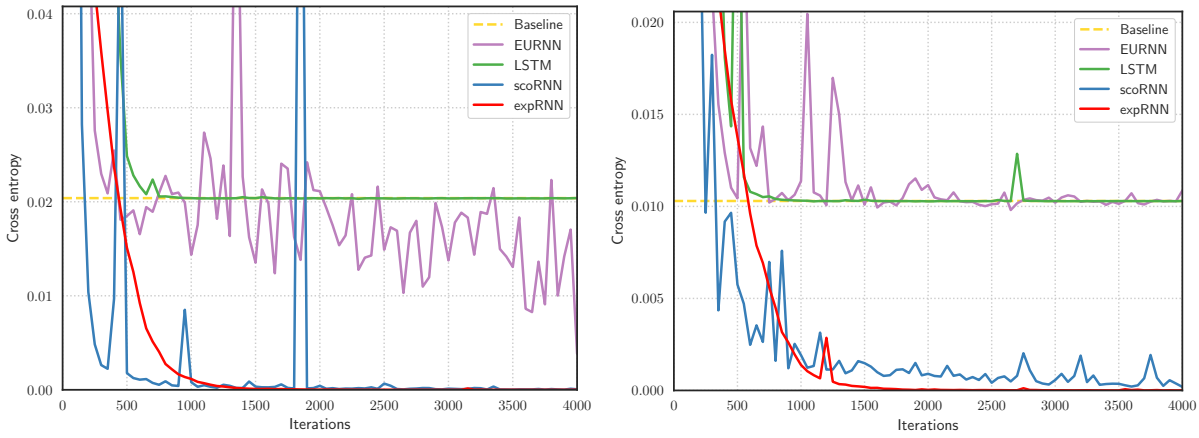

Figure 6.2: Cross entropy of the different algorithms in the copying problem for $L = 1000$ (left) and $L = 2000$ (right).

characters $(b_1, b_2, \ldots, b_S)$. In other words, the system has to recall the initial S characters and reproduce them after detecting the input of the character `<start>`, which appears L time-steps after the end of the input characters. For example, for $A = 4$, $S = 5$, $L = 8$, if we represent `<blank>` with a dash and `<start>` with a colon, and the alphabet $\mathcal{A} = \{1, 2, 3, 4\}$, the following sequences could be an element of the (synthetic) dataset:

```
Input:   14221--------:----
Output:  -------------14221
```

The loss function for this task is the cross entropy. The standard baseline for this task is the output of $S + L$ `<blank>` symbols, followed by the remaining S symbols being outputted at random. This strategy yields a cross entropy of $S \log(A)/(L + 2S)$.

In the experiments we set $A = 9$, $S = 10$, as it was done in the previous papers.

We observe that the training of SCORNN is unstable, which is probably due to the degeneracies explained in Section 6.3.1. In the follow-up paper (Maduranga, Helfrich, and Ye 2019), SCURNN presents the same instabilities as its predecessor. As explained in Section 6.3.1, EXPRNN does not suffer from this, and can be observed in our experiments as a smoother convergence. In the more difficult problem, $L = 2000$, EXPRNN is the only architecture that is able to fully converge to the correct answer.

6.4.2 Pixel-by-pixel MNIST

Table 6.5 is structured so that architectures with the same number of parameters are compared together. The number n represents that we are working with a matrix of hidden size $n \times n$.

In this experiment we observed that EXPRNN and DTRIV models were able to saturate the capacity of the orthogonal RNN model for this task much faster than any other parametrisation, as per Table 6.5. We conjecture that coupling the exponential parametrisation with an LSTM cell or a GRU cell would yield a superior architecture. We leave this for future research.

As we can see, the addition of any dynamic trivialisation to the Lie parametrisation improves the results on this experiment by 0.4% out of the 1.3% possible in the largest size. Moreover, it always improves the previous results, suggesting that it is always a better option to use dynamic trivialisations rather

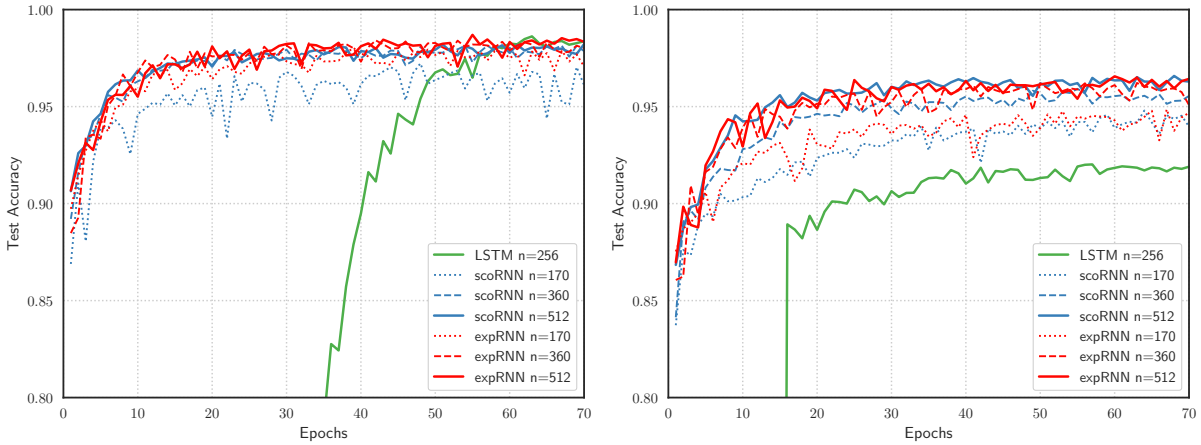

Figure 6.3: Test losses for several models on pixel-by-pixel MNIST (left) and permuted MNIST (right).

than just plain trivialisations. In general, we saw that DTRIV100 and DTRIV∞ gave the highest stability and the best results across the experiments. We conjecture that this is due to the fact that changing too often the base is detrimental for the accumulated momentum and adaptive term that the algorithm stores internally.

6.4.3 TIMIT speech dataset

We performed speech prediction on audio data with our model. We used the TIMIT speech dataset (Garofolo et al. 1993) which is a collection of real-world speech recordings. The task accounts for predicting the log-magnitude of incoming frames of a short-time Fourier transform as it was first proposed in (Wisdom et al. 2016).

We use the separation in train / test proposed in (Garofolo et al. 1993), having 3640 utterances for the training set, a validation set of size 192, and a test set of size 400. The validation / test division and the whole preprocessing of the dataset was done according to (Wisdom et al. 2016). The preprocessing goes as follows: The data is sampled at 8kHz and then cut into time frames of the same size. These frames are then transformed into the log-magnitude Fourier space and finally, they are normalised according to a per-training set, test set, and validation set basis. The result of this process gives sequences of 129 complex numbers per step, and a variable length between 61 and 490.

In this experiment we see a similar behaviour of the dynamic trivialisations as the one already seen in the MNIST and P-MNIST experiments. We also see in this experiment that DTRIV100 and DTRIV∞ always improve the performance of their static counterparts with base at the identity (EXPRNN). They also greatly improve over vanilla RGD.

Remark 6.4.3 (Computing the correct loss on TIMIT). As a side note, we must say that the results in this experiment should be interpreted under the following fact: We had access to two of the implementations for the tests for the other architectures regarding this experiment, and neither of them correctly handled sequences with different lengths present in this experiment. We suspect that the other implementations followed a similar approach, given that the results that they get are of the same order. In particular, the implementation released by Wisdom, which is the only publicly available implementation

Table 6.5: Best test accuracy at MNIST and P-MNIST.

Model	n	MNIST	P-MNIST
dtriv1	170	**98.3**	**95.2**
dtriv100	170	98.2	95.1
dtriv∞	170	98.1	95.0
exprnn	170	98.0	94.9
scornn	170	97.2	94.8
scurnn	116	97.6	94.9
lstm	128	81.9	79.5
rgd	116	94.7	92.5
eunn	512	–	93.7
urnn	512	97.6	94.5
dtriv1	360	98.4	96.3
dtriv100	360	98.8	96.4
dtriv∞	360	**98.9**	**96.5**
exprnn	360	98.4	96.2
scornn	360	98.1	95.9
scurnn	250	98.3	96.2
lstm	256	88.8	88.8
rgd	256	96.1	93.9
urnn	2170	98.4	95.3
dtriv1	512	98.7	96.7
dtriv100	512	**99.1**	96.7
dtriv∞	512	99.0	**96.8**
exprnn	512	98.7	96.6
scornn	512	98.2	96.5
lstm	512	91.9	91.8
rgd	512	97.3	94.7

Table 6.6: Test MSE at the end of the epoch with the lowest validation MSE for the TIMIT task.

Model	n	Val. MSE	Test MSE
dtriv1	224	6.55	6.54
dtriv100	224	4.80	4.77
dtriv∞	224	**4.75**	**4.71**
exprnn	224	5.34	5.30
scornn	224	9.26	8.50
scurnn	128	9.42	7.23
lstm	84	15.42	14.30
rgd	128	15.07	14.58
eunn	158	15.57	18.51
dtriv1	322	4.56	4.55
dtriv100	322	3.80	3.76
dtriv∞	322	**3.39**	**3.76**
exprnn	322	4.42	4.38
scornn	322	8.48	7.82
lstm	120	13.93	12.95
rgd	192	15.10	14.50
eunn	256	15.90	15.31
dtriv1	425	4.21	4.17
dtriv100	425	2.02	1.99
dtriv∞	425	**2.00**	**1.97**
exprnn	425	5.52	5.48
scornn	425	7.97	7.36
scurnn	258	4.40	3.39
eunn	378	16.00	15.15
lstm	158	13.66	12.62
rgd	256	14.96	14.69

of this experiment, divides by a larger number than it should when computing the average MSE of a batch, hence reporting a lower MSE than the correct one. Even in this unfavourable scenario, our parametrisation is able to get results that are twice as good—the MSE loss function is a quadratic function—as those from the other architectures.

In the experiments in SCURNN they explicitly mention that they are computing the MSE without discarding the zeros used to pad the variable-length sequences (Maduranga, Helfrich, and Ye 2019). As such, when computing the MSE, they are dividing by an incorrect number—the longest element in the batch times the elements in the batch—rather than by the correct one—the sum of the lengths of all the elements in the batch. We computed the correct validation and test loss in Table 6.6.

Bibliography

Absil, Pierre-Antoine, Robert Mahony, and Rodolphe Sepulchre (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press. ISBN: 978-0-691-13298-3. DOI: 10.1515/9781400830244.

Agarwal, Naman et al. (2020). "Adaptive regularization with cubics on manifolds". In: *Mathematical Programming*. ISSN: 0025-5610. DOI: 10.1007/s10107-020-01505-1.

Ahn, Kwangjun and Suvrit Sra (2020). "From Nesterov's estimate sequence to Riemannian acceleration". In: *Conference on Learning Theory, COLT*, pp. 84–118.

Alexandrino, Marcos M. and Renato G. Bettiol (2015). *Lie groups and geometric aspects of isometric actions*. Springer. ISBN: 978-3-319-16612-4. DOI: 10.1007/978-3-319-16613-1.

Allen-Zhu, Zeyuan (2018). "Natasha 2: Faster non-convex optimization than SGD". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 2680–2691.

Allen-Zhu, Zeyuan and Yuanzhi Li (2018). "Neon2: Finding local minima via first-order oracles". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 3720–3730.

Ambrose, Warren Arthur, Richard Sheldon Palais, and Isadore Manuel Singer (1960). "Sprays". In: *Anais da Academia Brasileira de Ciências* 32, pp. 163–178. ISSN: 0001-3765.

Arjovsky, Martin, Amar Shah, and Yoshua Bengio (2016). "Unitary evolution recurrent neural networks". In: *International Conference on Machine Learning, ICML*, pp. 1120–1128.

Arora, Sanjeev, Nadav Cohen, and Elad Hazan (2018). "On the optimization of deep networks: Implicit acceleration by overparameterization". In: *International Conference on Machine Learning, ICML*, pp. 244–253.

Arsigny, Vincent, Olivier Commowick, et al. (2006). "A log-Euclidean framework for statistics on diffeomorphisms". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention, MICCAI*, pp. 924–931.

Arsigny, Vincent, Pierre Fillard, et al. (2006). "Geometric means in a novel vector space structure on symmetric positive-definite matrices". In: *SIAM Journal on Matrix Analysis and Applications* 29.1, pp. 328–347. ISSN: 0895-4798. DOI: 10.1137/050637996.

Bačák, Miroslav (2014). *Convex analysis and optimization in Hadamard spaces*. Vol. 22. Nonlinear Analysis and Applications. De Gruyter, Berlin. ISBN: 978-3-11-036103-2. DOI: 10.1515/9783110361629.

Bader, Philipp, Sergio Blanes, and Fernando Casas (2019). "Computing the matrix exponential with an optimized Taylor polynomial approximation". In: *Mathematics* 7.12. ISSN: 2227-7390. DOI: 10.3390/math7121174.

Bah, Bubacarr et al. (2019). "Learning deep linear neural networks: Riemannian gradient flows and convergence to global minimizers". In: arXiv: 1910.05505.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate". In: *International Conference on Learning Representations, ICLR*.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181.

Bento, Glaydston C., Orizon P. Ferreira, and Jefferson G. Melo (2017). "Iteration-complexity of gradient, subgradient and proximal point methods on Riemannian manifolds". In: *Journal of Optimization Theory and Applications* 173.2, pp. 548–562. ISSN: 0022-3239. DOI: 10.1007/s10957-017-1093-4.

Bérard-Bergery, Lionel (1975). "Sur certaines fibrations d'espaces homogènes riemanniens". In: *Compositio Mathematica* 30, pp. 43–61. ISSN: 0010-437X.

Berg, Rianne van den et al. (2018). "Sylvester normalizing flows for variational inference". In: *Uncertainty in Artificial Intelligence, UAI*, pp. 393–402.

Berger, Marcel (1960). "Sur quelques variétés riemanniennes suffisamment pincées". In: *Bulletin de la Société Mathématique de France* 88, pp. 57–71. ISSN: 0037-9484. DOI: 10.24033/bsmf.1543.

Besse, Arthur L. (2008). *Einstein manifolds*. Classics in Mathematics. Reprint of the 1987 original. Springer. ISBN: 978-3-540-74120-6. DOI: 10.1007/978-3-540-74311-8.

Bloch, Anthony M. and Arieh Iserles (2005). "Commutators of skew-symmetric matrices". In: *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering* 15.3, pp. 793–801. ISSN: 0218-1274. DOI: 10.1142/S0218127405012417.

Bonnabel, Silvère (2013). "Stochastic gradient descent on Riemannian manifolds". In: *IEEE Transactions on Automatic Control* 58.9, pp. 2217–2229. ISSN: 0018-9286. DOI: 10.1109/TAC.2013.2254619.

Boumal, Nicolas, Pierre-Antoine Absil, and Coralia Cartis (2019). "Global rates of convergence for nonconvex optimization on manifolds". In: *IMA Journal of Numerical Analysis* 39.1, pp. 1–33. ISSN: 0272-4979. DOI: 10.1093/imanum/drx080.

Boumal, Nicolas, Bamdev Mishra, et al. (2014). "Manopt, a MATLAB toolbox for optimization on manifolds". In: *Journal of Machine Learning Research, JMLR* 15.1, pp. 1455–1459. ISSN: 1533-7928. DOI: 10.5555/2627435.2638581.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex optimization*. Cambridge University Press. ISBN: 0-521-83378-7. DOI: 10.1017/CBO9780511804441.

Brockett, Roger Ware (1972). "System theory on group manifolds and coset spaces". In: *SIAM Journal on Control and Optimization* 10.2, pp. 265–284. ISSN: 0363-0129. DOI: 10.1137/0310021.

Brown, Tom B. et al. (2020). "Language models are few-shot learners". In: *Advances in Neural Information Processing Systems, NeurIPS*.

Bubeck, Sébastien and Nicolò Cesa-Bianchi (2012). "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". In: *Foundations and Trends in Machine Learning* 5.1, pp. 1–122. ISSN: 1935-8237. DOI: 10.1561/2200000024.

Buchner, Michael A. (1977). "Simplicial structure of the real analytic cut locus". In: *Proceedings of the American Mathematical Society* 64.1, pp. 118–121. ISSN: 0002-9939. DOI: 10.2307/2040994.

Burer, Samuel and Renato D. C. Monteiro (2003). "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization". In: *Mathematical Programming* 95.2, Ser. B, pp. 329–357. ISSN: 0025-5610. DOI: 10.1007/s10107-002-0352-8.

— (2005). "Local minima and convergence in low-rank semidefinite programming". In: *Mathematical Programming* 103.3, Ser. A, pp. 427–444. ISSN: 0025-5610. DOI: 10.1007/s10107-004-0564-1.

Carmon, Yair et al. (2017). "'Convex until proven guilty': Dimension-free acceleration of gradient descent on non-convex functions". In: *International Conference on Machine Learning, ICML*, pp. 654–663.

— (2020). "Lower bounds for finding stationary points I". In: *Mathematical Programming* 184.1–2, pp. 71–120. ISSN: 0025-5610. DOI: 10.1007/s10107-019-01406-y.

Cartan, Élie (1928). *Leçons sur la géométrie des espaces de Riemann*. Gauthier-Villars.

— (1946). *Leçons sur la géométrie des espaces de Riemann*. Second. Gauthier-Villars.

Cartis, Coralia, Nicholas I. M. Gould, and Philippe L. Toint (2010). "On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems". In: *SIAM Journal on Optimization* 20.6, pp. 2833–2852. ISSN: 1052-6234. DOI: 10.1137/090774100.

Chavel, Isaac (1970). "A class of Riemannian homogeneous spaces". In: *Journal of Differential Geometry* 4, pp. 13–20. ISSN: 0022-040X. DOI: 10.4310/jdg/1214429272.

Cheeger, Jeff and David Gregory Ebin (2008). *Comparison theorems in Riemannian geometry*. Reprint of the 1975 original. AMS Chelsea Publishing. ISBN: 978-0-8218-4417-5. DOI: 10.1090/chel/365.

Chentsov, Nikolai N. (1956). "Weak convergence of stochastic processes whose trajectories have no discontinuities of the second kind and the 'heuristic' approach to the Kolmogorov–Smirnov tests". In: *SIAM Theory of Probability and Its Applications* 1.1, pp. 140–144. ISSN: 0040-585X. DOI: 10.1137/1101013.

Chung, Junyoung et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: arXiv: 1412.3555.

Ciresan, Dan Claudiu et al. (2010). "Deep, big, simple neural nets for handwritten digit recognition". In: *Neural Computation* 22.12, pp. 3207–3220. ISSN: 0899-7667. DOI: 10.1162/NECO_a_00052.

Cohen, Taco S. and Max Welling (2016). "Group equivariant convolutional networks". In: *International Conference on Machine Learning, ICML*, pp. 2990–2999.

Collobert, Ronan et al. (2011). "Natural language processing (almost) from scratch". In: *Journal of Machine Learning Research, JMLR* 12, pp. 2493–2537. ISSN: 1533-7928. DOI: 10.5555/1953048.2078186.

Criscitiello, Chris and Nicolas Boumal (2019). "Efficiently escaping saddle points on manifolds". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 5985–5995.

Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4, pp. 303–314. ISSN: 0932-4194. DOI: 10.1007/BF02551274.

Deng, Jia et al. (2009). "ImageNet: A large-scale hierarchical image database". In: *Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 248–255.
Du, Simon S. et al. (2017). "Gradient descent can take exponential time to escape saddle points". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 1067–1077.
Duchi, John C., Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research, JMLR* 12.Jul, pp. 2121–2159. ISSN: 1533-7928. DOI: 10.5555/1953048.2021068.
Dudley, Richard M. (1967). "The sizes of compact subsets of Hilbert space and continuity of Gaussian processes". In: *Journal of Functional Analysis* 1, pp. 290–330. DOI: 10.1016/0022-1236(67)90017-1.
Edelman, Alan, Tomás A. Arias, and Steven Thomas Smith (1998). "The geometry of algorithms with orthogonality constraints". In: *SIAM Journal on Matrix Analysis and Applications* 20.2, pp. 303–353. ISSN: 0895-4798. DOI: 10.1137/S0895479895290954.
Ehresmann, Charles (1952). "Les connexions infinitésimales dans un espace fibré différentiable". In: *Séminaire Bourbaki: Années 1948–1951, exposés 1-49*. Ed. by Nicolas Bourbaki. Séminaire Bourbaki 1. Talk 24. Société mathématique de France, pp. 153–168.
Eichhorn, Jürgen (1991). "The boundedness of connection coefficients and their derivatives". In: *Mathematische Nachrichten* 152, pp. 145–158. ISSN: 0025-584X. DOI: 10.1002/mana.19911520113.
Eldan, Ronen and Ohad Shamir (2016). "The power of depth for feedforward neural networks". In: *Conference on Learning Theory, COLT*, pp. 907–940.
Eldering, Jaap (2012). "Persistence of noncompact normally hyperbolic invariant manifolds in bounded geometry". In: *Comptes Rendus Mathématique. Académie des Sciences. Paris* 350.11-12, pp. 617–620. ISSN: 1631-073X. DOI: 10.1016/j.crma.2012.06.009.
Eschenburg, Jost-Hinrich (1994). *Comparison theorems in Riemannian geometry*. Dipartimento di Matematica: Lecture notes series. Università delli studi di Trento, Dipartimento di matematica.
Eschenburg, Jost-Hinrich and Ernst Heintze (1990). "Comparison theory for Riccati equations". In: *Manuscripta Mathematica* 68.2, pp. 209–214. ISSN: 0025-2611. DOI: 10.1007/BF02568760.
Falorsi, Luca et al. (2019). "Reparameterizing distributions on Lie groups". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*, pp. 3244–3253.
Ferreira, Orizon P., Mauricio Silva Louzeiro, and Leandro F. Prudente (2019). "Iteration-complexity of the subgradient method on Riemannian manifolds with lower bounded curvature". In: *Optimization* 68.4, pp. 713–729. ISSN: 0233-1934. DOI: 10.1080/02331934.2018.1542532.
Ferreira, Orizon P. and Benar F. Svaiter (2002). "Kantorovich's theorem on Newton's method in Riemannian manifolds". In: *Journal of Complexity* 18.1, pp. 304–329. ISSN: 0885-064X. DOI: 10.1006/jcom.2001.0582.
França, Guilherme, Michael Irwin Jordan, and René Vidal (2020). "On dissipative symplectic integration with applications to gradient-based optimization". In: arXiv: 2004.06840.
Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Proceedings of the U.S.-Japan Seminar in Competition and Cooperation in Neural Nets (Kyoto)*. Springer, pp. 267–285. DOI: 10.1016/0031-3203(82)90024-3.
Gallier, Jean and Jocelyn Quaintance (2020). *Differential geometry and Lie groups. A computational perspective*. University of Pennsylvania. URL: https://www.seas.upenn.edu/~jean/diffgeom-spr-I.pdf.
Garofolo, John S. et al. (1993). *DARPA TIMIT acoustic phonetic continuous speech corpus*. LDC Catalog No. LDC93S1. DOI: 10.35111/17gk-bn40.
Ge, Jianquan (2014). "DDVV-type inequality for skew-symmetric matrices and Simons-type inequality for Riemannian submersions". In: *Advances in Mathematics* 251, pp. 62–86. ISSN: 0001-8708. DOI: 10.1016/j.aim.2013.10.010.
Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*, pp. 249–256.
Goodfellow, Ian J., Yoshua Bengio, and Aaron C. Courville (2016). *Deep learning*. The MIT Press. ISBN: 978-0-262-03561-3. DOI: 10.1038/nature14539.
Goodfellow, Ian J., Jean Pouget-Abadie, et al. (2014). "Generative adversarial nets". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 2672–2680.
Graves, Alex, Greg Wayne, and Ivo Danihelka (2014). "Neural Turing machines". In: arXiv: 1410.5401.
Greene, Robert E. (1978). "Complete metrics of bounded curvature on noncompact manifolds". In: *Archiv der Mathematik* 31.1, pp. 89–95. ISSN: 0003-889X. DOI: 10.1007/BF01226419.

Gromov, Mikhael (1981). *Structures métriques pour les variétés riemanniennes*. Textes Mathématiques. CEDIC, Paris. ISBN: 2-7124-0714-8.
Gulliver, Robert (1975). "On the variety of manifolds without conjugate points". In: *Transactions of the American Mathematical Society* 210, pp. 185–201. ISSN: 0002-9947. DOI: 10.2307/1997131.
Hamming, Richard Wesley (1962). *Numerical methods for scientists and engineers*. International Series in Pure and Applied Mathematics. McGraw-Hill.
Handel, Ramon van (2014). *Probability in high dimension*. Princeton univesity. URL: https://web.math.princeton.edu/~rvan/APC550.pdf.
Helfrich, Kyle E., Devin Willmott, and Qiang Ye (2018). "Orthogonal recurrent neural networks with scaled Cayley transform". In: *International Conference on Machine Learning, ICML*, pp. 1974–1983.
Helgason, Sigurður (1978). *Differential geometry, Lie groups, and symmetric spaces*. Vol. 80. Academic Press. ISBN: 0-12-338460-5.
Henaff, Mikael, Arthur Szlam, and Yann LeCun (2016). "Recurrent orthogonal networks and long-memory tasks". In: *International Conference on Machine Learning, ICML*, pp. 2034–2042.
Hermann, Robert (1960). "A sufficient condition that a mapping of Riemannian manifolds be a fibre bundle". In: *Proceedings of the American Mathematical Society* 11, pp. 236–242. ISSN: 0002-9939. DOI: 10.2307/2032963.
Higham, Nicholas J. (2008). *Functions of matrices: Theory and computation*. Vol. 104. SIAM. ISBN: 978-0-89871-646-7. DOI: 10.1137/1.9780898717778.
— (2009). "The scaling and squaring method for the matrix exponential revisited". In: *SIAM Review* 51.4, pp. 747–764. ISSN: 0036-1445. DOI: 10.1137/090768539.
Hildebrandt, Stefan, Jürgen Jost, and Kjell-Ove Widman (1980). "Harmonic mappings and minimal submanifolds". In: *Inventiones Mathematicae* 62.2, pp. 269–298. ISSN: 0020-9910. DOI: 10.1007/BF01389161.
Hinton, Geoffrey Everest et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* 29. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597.
Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.
Hornik, Kurt (1991). "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2, pp. 251–257. ISSN: 0893-6080. DOI: 10.1016/0893-6080(91)90009-T.
Huang, Lei et al. (2018). "Orthogonal weight normalization: Solution to optimization over multiple dependent Stiefel manifolds in deep neural networks". In: *AAAI Conference on Artificial Intelligence*, pp. 3271–3278.
Huang, Wen, Kyle A. Gallivan, and Pierre-Antoine Absil (2015). "A Broyden class of quasi-Newton methods for Riemannian optimization". In: *SIAM Journal on Optimization* 25.3, pp. 1660–1685. ISSN: 1052-6234. DOI: 10.1137/140955483.
Huang, Yanping et al. (2019). "GPipe: Efficient training of giant neural networks using pipeline parallelism". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 103–112.
Hyland, Stephanie L. and Gunnar Rätsch (2017). "Learning unitary operators with help from $\mathfrak{u}(n)$". In: *AAAI Conference on Artificial Intelligence*, pp. 2050–2058.
Iserles, Arieh, Hans Z. Munthe-Kaas, et al. (2000). "Lie-group methods". In: *Acta Numerica* 9, pp. 215–365. ISSN: 0962-4929. DOI: 10.1017/S0962492900002154.
Iserles, Arieh and Syvert P. Nørsett (1999). "On the solution of linear differential equations in Lie groups". In: *Philosophical Transactions of the Royal Society of London. Series A* 357.1754, pp. 983–1019. ISSN: 1364-503X. DOI: 10.1098/rsta.1999.0362.
Itoh, Jin-ichi and Minoru Tanaka (1998). "The dimension of a cut locus on a smooth Riemannian manifold". In: *Tohoku Mathematical Journal* 50.4, pp. 571–575. ISSN: 0040-8735. DOI: 10.2748/tmj/1178224899.
Jing, Li et al. (2017). "Tunable efficient unitary neural networks (EUNN) and their application to RNNs". In: *International Conference on Machine Learning, ICML*, pp. 1733–1741.
Jost, Jürgen (2017). *Riemannian geometry and geometric analysis*. Seventh. Springer. ISBN: 978-3-319-61859-3. DOI: 10.1007/978-3-319-61860-9.
Kalchbrenner, Nal and Phil Blunsom (2013). "Recurrent continuous translation models". In: *Conference on Empirical Methods in Natural Language Processing, EMNLP*, pp. 1700–1709.
Kankaanrinta, Marja (2005). "Proper smooth G-manifolds have complete G-invariant Riemannian metrics". In: *Topology and its Applications* 153.4, pp. 610–619. ISSN: 0166-8641. DOI: 10.1016/j.topol.2005.01.034.

Karcher, Hermann (1970). "A short proof of Berger's curvature tensor estimates". In: *Proceedings of the American Mathematical Society* 26, pp. 642–644. ISSN: 0002-9939. DOI: 10.2307/2037127.
Kasai, Hiroyuki, Hiroyuki Sato, and Bamdev Mishra (2018). "Riemannian stochastic recursive gradient algorithm with retraction and vector transport and its convergence analysis". In: *International Conference on Machine Learning, ICML*, pp. 2521–2529.
Kaul, Helmut (1976). "Schranken für die Christoffelsymbole". In: *Manuscripta Mathematica* 19.3, pp. 261–273. ISSN: 0025-2611. DOI: 10.1007/BF01170775.
Kingma, Diederik P. and Jimmy Lei Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations, ICLR*.
Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 10215–10224.
Kingma, Diederik P. and Max Welling (2014). "Auto-encoding variational Bayes". In: *International Conference on Learning Representations, ICLR*.
Kobayashi, Shoshichi and Katsumi Nomizu (1963). *Foundations of differential geometry*. Vol. I. John Wiley & Sons.
— (1969). *Foundations of differential geometry*. Vol. II. John Wiley & Sons.
Kowalski, Oldřich (1980). *Generalized symmetric spaces*. Vol. 805. Lecture Notes in Mathematics. Springer. ISBN: 3-540-10002-4.
Kozlov, Sergei E. (2000). "Geometry of real Grassmann manifolds. Part III". In: *Journal of Mathematical Sciences* 100.3, pp. 2254–2268. ISSN: 1072-3374. DOI: 10.1007/s10958-000-0009-1.
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Everest Hinton (2012). "ImageNet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 1106–1114.
Kuznetsova, Alina et al. (2020). "The open images dataset V4". In: *International Journal of Computer Vision* 128.7, pp. 1956–1981. ISSN: 0920-5691. DOI: 10.1007/s11263-020-01316-z.
Lang, Serge (1999). *Fundamentals of differential geometry*. Vol. 191. Graduate Texts in Mathematics. Springer. ISBN: 0-387-98593-X. DOI: 10.1007/978-1-4612-0541-8.
LeCun, Yann, Corinna Cortes, and Christopher C. J. Burges (1998). *The MNIST database of handwritten digits*. Accessed: 2019-02-06. URL: http://yann.lecun.com/exdb/mnist/.
Ledoux, Michel and Michel Talagrand (1991). *Probability in Banach spaces. Isoperimetry and processes*. Vol. 23. Springer. ISBN: 3-540-52013-9. DOI: 10.1007/978-3-642-20212-4.
Lee, Jason D. et al. (2016). "Gradient descent only converges to minimizers". In: *Conference on Learning Theory, COLT*, pp. 1246–1257.
Lezcano-Casado, Mario (2019). "Trivializations for gradient-based optimization on manifolds". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 9154–9164.
Lezcano-Casado, Mario and David Martínez-Rubio (2019). "Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group". In: *International Conference on Machine Learning, ICML*, pp. 3794–3803.
Lichnerowicz, André (1958). *Géométrie des groupes de transformations*. Travaux et Recherches Mathématiques, III.
Lu, Zhiqin (2012). "Remarks on the Böttcher-Wenzel inequality". In: *Linear Algebra and its Applications* 436.7, pp. 2531–2535. ISSN: 0024-3795. DOI: 10.1016/j.laa.2011.09.016.
Luenberger, David G. (1972). "The gradient projection method along geodesics". In: *Management Science* 18, pp. 620–631. ISSN: 0025-1909. DOI: 10.1287/mnsc.18.11.620.
Maduranga, Kehelwala D. G., Kyle E. Helfrich, and Qiang Ye (2019). "Complex unitary recurrent neural networks using scaled Cayley transform". In: *AAAI Conference on Artificial Intelligence*, pp. 4528–4535.
Magnus, Wilhelm (1954). "On the exponential solution of differential equations for a linear operator". In: *Communications on Pure and Applied Mathematics* 7.4, pp. 649–673. ISSN: 0010-3640. DOI: 10.1002/cpa.3160070404.
Mahoney, Matt (2006). *Large text compression benchmark*. Accessed: 18-12-2020. URL: http://mattmahoney.net/dc/text.html.
Mahony, Robert and Jonathan H. Manton (2002). "The geometry of the Newton method on non-compact Lie groups". In: *Journal of Global Optimization* 23.3-4, pp. 309–327. ISSN: 0925-5001. DOI: 10.1023/A:1016586831090.
Manton, Jonathan H. (2015). "A framework for generalising the Newton method and other iterative methods from Euclidean space to manifolds". In: *Numerische Mathematik* 129.1, pp. 91–125. ISSN: 0029-599X. DOI: 10.1007/s00211-014-0630-4.

Massart, Estelle and Pierre-Antoine Absil (2020). "Quotient geometry with simple geodesics for the manifold of fixed-rank positive-semidefinite matrices". In: *SIAM Journal on Matrix Analysis and Applications* 41.1, pp. 171–198. ISSN: 0895-4798. DOI: 10.1137/18M1231389.

Mathias, Roy (1996). "A chain rule for matrix functions and applications". In: *SIAM Journal on Matrix Analysis and Applications* 17.3, pp. 610–620. ISSN: 0895-4798. DOI: 10.1137/S0895479895283409.

McAlpin, John Harris (1965). "Infinite dimensional manifolds and Morse theory". Ph.D. dissertation. Columbia University.

Meghwanshi, Mayank et al. (2018). "McTorch, a manifold optimization library for deep learning". In: arXiv: 1810.01811.

Mhammedi, Zakaria et al. (2017). "Efficient orthogonal parametrisation of recurrent neural networks using Householder reflections". In: *International Conference on Machine Learning, ICML*, pp. 2401–2409.

Milnor, John Willard (1976). "Curvatures of left invariant metrics on Lie groups". In: *Advances in Mathematics* 21, pp. 293–329. ISSN: 0001-8708. DOI: 10.1016/S0001-8708(76)80002-3.

Mishra, Bamdev et al. (2014). "Fixed-rank matrix factorizations and Riemannian low-rank optimization". In: *Computational Statistics* 29.3-4, pp. 591–621. ISSN: 0943-4062. DOI: 10.1007/s00180-013-0464-z.

Al-Mohy, Awad H. and Nicholas J. Higham (2009). "A new scaling and squaring algorithm for the matrix exponential". In: *SIAM Journal on Matrix Analysis and Applications* 31.3, pp. 970–989. ISSN: 0895-4798. DOI: 10.1137/09074721X.

Moler, Cleve and Charles Van Loan (1978). "Nineteen dubious ways to compute the exponential of a matrix". In: *SIAM Review* 20.4, pp. 801–836. ISSN: 0036-1445. DOI: 10.1137/1020098.

— (2003). "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later". In: *SIAM Review* 45.1, pp. 3–49. ISSN: 0036-1445. DOI: 10.1137/S00361445024180.

Munthe-Kaas, Hans Z. and Olivier Verdier (2016). "Integrators on homogeneous spaces: Isotropy choice and connections". In: *Foundations of Computational Mathematics* 16.4, pp. 899–939. ISSN: 1615-3375. DOI: 10.1007/s10208-015-9267-7.

Murty, Katta G. and Santosh N. Kabadi (1987). "Some NP-complete problems in quadratic and nonlinear programming". In: *Mathematical Programming* 39.2, pp. 117–129. ISSN: 0025-5610. DOI: 10.1007/BF02592948.

Myers, Sumner Byron (1935). "Connections between differential geometry and topology I. Simply connected surfaces". In: *Proceedings of the National Academy of Sciences of the United States of America* 21.4, p. 225. ISSN: 0012-7094. DOI: 10.1215/S0012-7094-35-00126-0.

Myers, Sumner Byron and Norman Earl Steenrod (1939). "The group of isometries of a Riemannian manifold". In: *Annals of Mathematics* 40, pp. 400–416. ISSN: 0003-486X. DOI: 10.2307/1968928.

Naber, Gregory L. (2011). *Topology, geometry, and gauge fields*. Second. Vol. 25. Texts in Applied Mathematics. Springer. ISBN: 978-1-4419-7253-8. DOI: 10.1007/978-1-4419-7254-5.

Nemirovsky, Arkadi S. and David Berkovich Yudin (1983). *Problem complexity and method efficiency in optimization*. Translated from the Russian. John Wiley & Sons. ISBN: 0-471-10345-4. DOI: 10.1137/1027074.

Nesterov, Yurii (2004). *Introductory lectures on convex optimization: A basic course*. Vol. 87. Applied Optimization. Kluwer Academic Publishers. ISBN: 1-4020-7553-7. DOI: 10.1007/978-1-4419-8853-9.

Nomizu, Katsumi (1954). "Invariant affine connections on homogeneous spaces". In: *American Journal of Mathematics* 76.1, pp. 33–65. ISSN: 0002-9327. DOI: 10.2307/2372398.

Nomizu, Katsumi and Hideki Ozeki (1961). "The existence of complete Riemannian metrics". In: *Proceedings of the American Mathematical Society* 12.6, pp. 889–891. ISSN: 0002-9939. DOI: 10.2307/2034383.

O'Neill, Barrett (1966). "The fundamental equations of a submersion". In: *Michigan Mathematical Journal* 13, pp. 459–469. ISSN: 0026-2285. DOI: 10.1307/mmj/1028999604.

— (1983). *Semi-Riemannian geometry*. Vol. 103. Pure and Applied Mathematics. Academic Press. ISBN: 0-12-526740-1.

Ott, Myle et al. (2018). "Scaling neural machine translation". In: *Conference on Machine Translation, WMT*, pp. 1–9. DOI: 10.18653/v1/w18-6301.

Ozay, Mete and Takayuki Okatani (2016). "Optimization on submanifolds of convolution kernels in CNNs". In: arXiv: 1610.07008.

Palais, Richard Sheldon (1961). "On the existence of slices for actions of non-compact Lie groups". In: *Annals of Mathematics* 73, pp. 295–323. ISSN: 0003-486X. DOI: 10.2307/1970335.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *International Conference on Machine Learning, ICML*, pp. 1310–1318.

Petersen, Peter (2016). *Riemannian geometry*. Third. Graduate Texts in Mathematics. Springer. ISBN: 978-3-319-26652-7. DOI: 10.1007/978-3-319-26654-1.

Rasmussen, Carl Edward and Christopher K. I. Williams (2005). *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning. The MIT Press. ISBN: 978-0-262-18253-9.
Rauch, Harry Ernest (1951). "A contribution to differential geometry in the large". In: *Annals of Mathematics* 54, pp. 38–55. ISSN: 0003-486X. DOI: 10.2307/1969309.
— (1959). *Geodesics and curvature in differential geometry in the large*. Yeshiva University, Graduate School of Mathematical Sciences.
— (1966). "Geodesics and Jacobi equations on homogeneous Riemannian manifolds". In: *Proceedings of the U.S.-Japan Seminar in Differential Geometry (Kyoto)*. Nippon Hyoronsha, Tokyo, pp. 115–127.
Al-Rfou, Rami et al. (2016). "Theano: A Python framework for fast computation of mathematical expressions". In: arXiv: 1410.5401.
Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *Annals of Mathematical Statistics* 22, pp. 400–407. ISSN: 0003-4851. DOI: 10.1214/aoms/1177729586.
Rossmann, Wulf (2002). *Lie groups: An introduction through linear groups*. Oxford Graduate Texts in Mathematics. Oxford University Press. ISBN: 0-19-859683-9. DOI: 10.1017/S0025557200174716.
Saon, George et al. (2017). "English conversational telephone speech recognition by humans and machines". In: *INTERSPEECH*, pp. 132–136.
Sard, Arthur (1965). "Hausdorff measure of critical images on Banach manifolds". In: *American Journal of Mathematics* 87, pp. 158–174. ISSN: 0002-9327. DOI: 10.2307/2373229.
Sato, Hiroyuki, Hiroyuki Kasai, and Bamdev Mishra (2019). "Riemannian stochastic variance reduced gradient algorithm with retraction and vector transport". In: *SIAM Journal on Optimization* 29.2, pp. 1444–1472. ISSN: 1052-6234. DOI: 10.1137/17M1116787.
Scarselli, Franco et al. (2009). "The graph neural network model". In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80. ISSN: 1045-9227. DOI: 10.1109/TNN.2008.2005605.
Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press. ISBN: 978-1-10-705713-5. DOI: 10.1017/CBO9781107298019.
Smith, Steven Thomas (1993). "Geometric optimization methods for adaptive filtering". Ph.D. dissertation. Harvard University.
Spivak, Michael (1999). *A comprehensive introduction to differential geometry*. Third. Vol. III. Publish or Perish. ISBN: 978-0914098720.
Steenrod, Norman Earl (1951). *The topology of fibre bundles*. Princeton Mathematical Series, vol. 14. Princeton University Press. DOI: 10.1515/9781400883875.
Sun, Yue, Nicolas Flammarion, and Maryam Fazel (2019). "Escaping from saddle points on Riemannian manifolds". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 7274–7284.
Talagrand, Michel (1987). "Regularity of Gaussian processes". In: *Acta Mathematica* 159.1-2, pp. 99–149. ISSN: 0001-5962. DOI: 10.1007/BF02392556.
— (1992). "A simple proof of the majorizing measure theorem". In: *Geometric and Functional Analysis* 2.1, pp. 118–125. ISSN: 1016-443X. DOI: 10.1007/BF01895708.
— (1994). "Constructions of majorizing measures, Bernoulli processes and cotype". In: *Geometric and Functional Analysis* 4.6, pp. 660–717. ISSN: 1016-443X. DOI: 10.1007/BF01896658.
Tomczak, Jakub M. and Max Welling (2016). "Improving variational auto-encoders using Householder flow". In: arXiv: 1611.09630.
Tripuraneni, Nilesh et al. (2018). "Averaging stochastic gradient descent on Riemannian manifolds". In: *Conference on Learning Theory, COLT*, pp. 650–687.
Vandereycken, Bart (2013). "Low-rank matrix completion by Riemannian optimization". In: *SIAM Journal on Optimization* 23.2, pp. 1214–1236. ISSN: 1052-6234. DOI: 10.1137/110845768.
Vandereycken, Bart, Pierre-Antoine Absil, and Stefan Vandewalle (2013). "A Riemannian geometry with complete geodesics for the set of positive semidefinite matrices of fixed rank". In: *IMA Journal of Numerical Analysis* 33.2, pp. 481–514. ISSN: 0272-4979. DOI: 10.1093/imanum/drs006.
Vapnik, Vladimir Naumovich (1995). *The nature of statistical learning theory*. Springer. ISBN: 978-1-4757-2442-4. DOI: 10.1007/978-1-4757-2440-0.
Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 5998–6008.
Vershynin, Roman (2018). *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge University Press. ISBN: 9781108415194. DOI: 10.1017/9781108231596.
Vilms, Jaak (1970). "Totally geodesic maps". In: *Journal of Differential Geometry* 4.1, pp. 73–79. ISSN: 0022-040X. DOI: 10.4310/jdg/1214429276.
Vorontsov, Eugene et al. (2017). "On orthogonality and learning recurrent networks with long term dependencies". In: *International Conference on Machine Learning, ICML*, pp. 3570–3578.

Wengert, Robert E. (1964). "A simple automatic derivative evaluation program". In: *Communications of the ACM* 7.8, pp. 463–464. DOI: 10.1145/355586.364791.

Wibisono, Andre, Ashia C. Wilson, and Michael Irwin Jordan (2016). "A variational perspective on accelerated methods in optimization". In: *Proceedings of the National Academy of Sciences of the United States of America* 113.47, E7351–E7358. ISSN: 0027-8424. DOI: 10.1073/pnas.1614734113.

Wilson, Ashia C. (2018). "Lyapunov arguments in optimization". Ph.D. dissertation. U.C. Berkeley.

Wisdom, Scott et al. (2016). "Full-capacity unitary recurrent neural networks". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 4880–4888.

Wong, Yung-chow (1968). "Sectional curvatures of Grassmann manifolds". In: *Proceedings of the National Academy of Sciences of the United States of America* 60, pp. 75–79. ISSN: 0027-8424. DOI: 10.1073/pnas.60.1.75.

Yu, Tao and Christopher De Sa (2019). "Numerically accurate hyperbolic embeddings using tiling-based models". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 2021–2031.

Zhang, Hongyi, Sashank J. Reddi, and Suvrit Sra (2016). "Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds". In: *Advances in Neural Information Processing Systems, NeurIPS*, pp. 4592–4600.

Zhang, Jiong, Qi Lei, and Inderjit Dhillon (2018). "Stabilizing gradients for deep neural networks via efficient SVD parametrization". In: *International Conference on Machine Learning, ICML*, pp. 5801–5809.