JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

Optimisation Methods with Algebraic Manifold Constraints

Optimierungsmethoden mit Nebenbedingungen durch algebraische Mannigfaltigkeiten

Dissertation

zur Erlangung des Doktorgrads
der naturwissenschaftlichen Fachbereiche
der Justus-Liebig-Universität Gießen

vorgelegt von:
Tobias Durchholz

Betreuer:	Prof. Dr. Martin Buhmann
2. Gutachter:	Prof. Dr. Oleg Davydov
3. Prüfer:	Prof. Dr. Bernhard Mühlherr
4. Prüfer:	PD Dr. Matthias Wendlandt

Vielen Dank an meinen Betreuer Herrn Prof. Martin Buhmann für jegliche Hilfe und Unterstützung, welche mir bei der Erstellung dieser Arbeit zuteil geworden ist. Insbesondere möchte ich mich aber für die Möglichkeit bedanken, dass ich weiter an der JLU und damit in Gießen meiner Promotion nachkommen konnte.
Außerdem möchte ich mich bei den Mitarbeitern des Instituts bedanken, welche mich auch auf meinem Weg begleitet haben. Besonderer Dank geht dabei an die aktuellen und ehemaligen Doktoranden für die häufigen fachlichen und noch häufiger nicht-fachlichen Gespräche.
Der größte Dank gilt jedoch meiner Familie für die Unterstützung meinen eigenen Weg zu finden und zu gehen.

Zusammenfassung

In dieser Dissertation wird ein Ansatz zum Lösen von Optimierungsproblemen mit Nebenbedingungen mit Mehrschrittverfahren untersucht. Diese Nebenbedingungen werden durch algebraische Mannigfaltigkeiten - also implizit durch Polynome beschrieben. Auf derartigen Mannigfalltigkeiten werden Algorithmen entwickelt, welche die Bewegung auf der Mannigfaltigkeit, den Transport von alten Such- und Abstiegsrichtungen berechnen und das logarithmische Problem lösen. Dann werden diese Methoden benutzt, um die Methode des steilsten Abstiegs, eine konjugierte Gradienten Methode und ein LBFGS-Verfahren auf eine Reihe von Testproblemen der CUTEr/st Reihe anzuwenden.

Abstract

In this dissertation an approach to multistep algorithms on constrained optimisation problems is discussed. Constraints will be given by algebraic manifolds, therefore the manifold will be also be an affine variety. Algorithms for moving along geodesics, parallel transport of previous search and descend directions and the logarithmic problems are developed. Then those are tested together with the steepest descent method, a conjugate gradient method and BFGS algorithm on test functions from the CUTEr/st problem set.

Contents

Chapter 1

Introduction

Optimisation is the field of numerical analysis where one is interested in algorithmically approximating minimal values and the according argument of some objective function $L : \mathbb{R}^n \to \mathbb{R}$. We will write such a problem as

$$\operatorname*{argmin}_{x \in \mathbb{R}^n} L(x). \tag{1.1}$$

Often one is not only interested in finding the minimum in $\mathbb{R}^n$, but in some subset $\mathcal{M} \subset \mathbb{R}^n$ often described by some equations and inequalities. Those problems are called *constrained* and one writes

$$\operatorname*{argmin}_{x \in \mathcal{M}} L(x). \tag{1.2}$$

In recent years the interest in constrained optimisation with $\mathcal{M}$ being a manifold has grown. Most works on manifolds (like [AMS09; Bou14]...) use the existence and knowledge of retractions which give some understanding about moving on the manifold by mapping elements of the tangent space to elements of the manifold, while also transporting vectors from one tangent space to another.
In this dissertation we will develop algorithms to make unconstrained optimisation methods applicable to manifolds which are described by polynomial equality constraints. Manifolds are locally diffeomorphic to a Euclidean space on which an unconstrained method is applicable. The Euclidean structure will induce a structure on the manifold itself. This structure will then be used to construct algorithms to calculate movement and transport of directions along the manifold.
In the second chapter we will give a brief introduction to the unconstrained

optimisation algorithms which we would like to make applicable to manifolds. This includes single and multistep algorithm. We will later test the algorithms using the steepest descent method, a conjugate gradient method and a LBFGS method. Also approaches to do the line search are discussed.
In the third chapter methods to approach differential equations will be discussed. Those will later on be applied to the differential equations we develop in the fourth chapter. Those differential equations will describe the movement along the manifold. Moving in a direction will give an *initial value problem*, while transporting tangent vectors along the manifold will include solving the logarithmic problem. This means, given two points on the manifold, to find the tangent vector, moving along which one gets from the first to the second point. In terms of differential equations this is a *boundary value problem*. We will try to solve those with a *shooting method*, for which we will discuss an iterative corrector.
The fourth chapter then will give a general introduction to the geometry and objects of a manifold. Also different methods to implement a manifold in a computer program are discussed. Of those we will further on consider manifolds which are also describable by affine varieties. Therefore those manifolds will be implicitly described by polynomials. If the implicit function theorem is applicable to those polynomials we are able to rewrite the differential equations describing geodesics on this manifold into a simpler form. Also with Groebner bases and elimination theory a criteria is developed to check if the implicit function theorem is applicable to a given set of polynomials.
Chapter five finally uses the geometry of algebraic manifolds to develop algorithms to calculate geodesic movement. The geodesic movement will then also be used to approximate the parallel transport along a geodesics by *ladder schemes*, which also involves solving the logarithmic problem.
The last chapter will summarize the application of beforehand developed algorithms to test problems. Different manifolds will be tested including the Stiefel manifold. The objective function are chosen from the CUTEr/st problem list. Those problems might themself include constraints, which we will replace with manifold constraints.
The actual implementation and used methods have been published in the GitHub repository [Dur23].

Chapter 2

Introduction to unconstrained Optimisation

Optimisation in $\mathbb{R}^n$ is a very broad topic. Thus in this chapter we will only introduce those optimisation methods, which we will later try to adjust to manifolds. In every algorithm the next point is iteratively calculated by

$$x_{k+1} := x_k + \alpha_k d_k \tag{2.1}$$

The *descent direction* d_k is calculated depending on the regarded method, while $\alpha_k > 0$ is determined by a one dimensional *line search*, discussed in section 2.3. First of all we will consider the Steepest Descent algorithm, which just calculates the gradient at each point and searches in the opposing direction. This has the benefit that previous directions can be omitted. Thereafter multistep algorithms are discussed, for which conjugate gradient methods and methods of the Broyden-Linear-Family, explicitly the BFGS algorithm, will be considered. For these previous descent directions and gradients are needed. On a manifold transporting elements from one tangent space to another is in general not simple and also numerically very costly. Therefore a variation of the BFGS method, which only stores a limited number of those directions, is considered. Those variations are called *limited-memory BFGS-algorithms*, shortly referred to by *L-BFGS*.

2.1 Single Step Method

Single step methods use only the information at the current iterate, which makes them in general easy to apply. The most basic approach is a gradient

descent method. The descent direction in (2.1) is given by $d_k := -\nabla L(x_k)$. By first order approximation of $L(x)$ by the Taylor polynomial

$$L(x_k - \alpha \nabla L(x_k)) = L(x_k) - \alpha \nabla L(x_k)^T \nabla L(x_k) + \mathcal{O}(\alpha^2) \tag{2.2}$$

one can easily see that there exists some, possible very small, $\alpha > 0$ such that $L(x_k - \alpha \nabla L(x_k)) < L(x_k)$ if $\nabla L(x_k) \neq 0$. This method is called *steepest descent method*. In practice the convergence can be very slow, for the method easily gets stuck in flat areas.

2.2 Multistep Methods

While single step methods only use information available at the current iterate other methods try to increase the rate of convergence by taking into account prior descent directions. These are used to better refine the current descent direction in each iteration or even to approximate the Hessian matrix itself. In Euclidean space vectors are just transported from one point to another by changing the root, this means the vector remains unchanged in each iteration - this changes on manifolds and will be discussed in Chapter 4.

2.2.1 Conjugate Gradient Methods

Initially developed by Hestenes and Stiefel in the 1950s conjugate gradient methods are initially used to solve systems of linear equations. One tries to solve the linear equation system by minimizing an inducing quadratic function. Later on it was extended to a wider range of applications, including unconstrained optimization. The key advantage of conjugate gradient methods is their ability to more efficiently explore the solution space. Another notable benefit of the conjugate gradient methods is their required memory, especially compared to the later discussed Quasi-Newton methods. The conjugate gradient methods require to store only one previous iterate, compared to Quasi-Newton methods, which require to store the approximate to the Hessian to calculate the next approximate. Their ability to exploit conjugate directions, coupled with memory efficiency and numerical stability, make them well-suited for a wide range of applications in science, engineering, and data analysis.

In general the algorithm to calculate the descent direction has the following form.

Algorithm 2.2.1: Conjugate Gradient Direction

Data: Previous gradient g_{k-1}, current iterate $x_k \in \mathbb{R}^n$.

Global: Gradient of the objective function ∇L, variation formula β.

Result: New descent direction $s = d_k$.

Function: CGDirection(x_k, d_{k-1}, g_{k-1}):

1 $g_k \leftarrow \nabla L(x_k)$

2 $\beta_k \leftarrow \beta(g_k, g_{k-1}, d_{k-1})$

3 $s \leftarrow -g_k + \beta_k d_{k-1}$

4 **return** s

There exist different variations of choosing β_k. Common formulas are

	$\beta(g_1, g_2, d)$
Fletcher-Reeves	$\frac{g_1^T g_1}{g_2^T g_2}$
Polak-Ribière	$\frac{g_1(g_1-g_2)}{g_2^T g_2}$
Hestenes-Stiefel	$\frac{g_1^T(g_1-g_2)}{d^T(g_1-g_2)}$
Dai-Yuah	$\frac{g_1^T g_1}{d^T(g_1-g_2)}$

To apply the conjugate gradient methods transporting the old gradient along the line search is necessary.

2.2.2 Quasi-Newton Methods - Broyden-Linear-Family

Like the conjugate gradient method before, the Quasi-Newton methods are a class of optimisation algorithms to solve unconstrained problems. They are particularly effective in finding solutions for high dimensional problems when computing the Hessian matrix is impractical or expensive. Newton's method utilizes the Hessian at each iteration to calculate the descent direction in (2.1) by

$$d_k = -H_L(x_k)^{-1} \nabla L(x_k). \tag{2.3}$$

Calculating the Hessian at every point can be very costly. To reduce these costs one tries to approximate the Hessian. In general the idea is to construct

a sequence of iterates that converge to an optimum by updating an approximation to the Hessian. The approximation should be efficient to compute and update, while retaining beneficial properties, like positive definiteness, thus ensuring convergence to a solution. Either a new approximation is calculated at each iteration or the approximation is updated in each iteration. In either way previous descent directions and gradients are used.

The storage of previous gradients or the full approximation may still turn out to be very costly for high dimensions. Also updating the approximate of the Hessian will usually require non-trivial matrix multiplication as equation (2.5) indicates. To avoid these burdens on the memory and computation there are variations of the methods which only store a limited number of vectors and use efficient methods to solve for new descent direction.

For methods of the Broyden-Linear family we require the objective function L to be at least $\mathcal{C}^1$. Then additionally to (2.1) we denote

$$\begin{aligned} g_i :=& \nabla L(x_i) \\ d_k :=& -B_k^{-1} g_k = -H_k g_k \\ \gamma_i :=& g_{i+1} - g_i \\ \delta_i :=& x_{i+1} - x_i \end{aligned} \tag{2.4}$$

Using (2.1) and (2.4) an approximate to the Hessian, precisely its inverse, is given by the Broyden-Linear-Family by

$$B_{k+1} = B_k - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k} + \frac{\gamma_k \gamma_k^T}{\delta_k^T \gamma_k} + \mu_k \left(\frac{\gamma_k}{\delta_k^T \gamma_k} - \frac{B_k \delta_k}{\delta_k^T B_k \delta_k^T} \right) \left(\frac{\gamma_k}{\delta_k^T \gamma_k} - \frac{B_k \delta_k}{\delta_k^T B_k \delta_k^T} \right)^T. \tag{2.5}$$

The factor μ_k is depending on δ_k and γ_k and defines different members of the Broyden-Linear-Family. The most notable updating formulas of this family are the DFP- and BFGS-Algorithm.

$$DFP : B_{k+1} := B_k - \frac{B_k \gamma_k \gamma_k^T B_k}{\gamma_k B_k \gamma_k} + \frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k} \tag{2.6}$$

$$BFGS : B_{k+1} := B_k - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k B_k \delta_k^T} + \frac{\gamma_k \gamma_k^T}{\delta_k^T \gamma_k}. \tag{2.7}$$

With suitable conditions the update B_k is positive definite and thus d_k is a direction of descent.

2.2.3 Limited Memory BFGS

In data intense fields, like machine learning, the data sets and models are very taxing on the memory and tend to drastically increase in higher dimensions, making the memory a bottleneck in the calculations. So one seeks to improve calculations by reducing memory consumption of the algorithm itself. This includes using a compact representation of the actual problem or using only a limited number of the recent data. Especially the latter can drastically decrease memory usage of multistep algorithms in higher dimensions.

Despite their limitations, limited memory optimization methods are often very effective at solving complex optimisation problems. They are used in a wide range of applications, including image and signal processing, machine learning, and computational finance. Some of the most commonly used limited memory optimization methods include the L-BFGS method, the conjugate gradient method, and other limited-memory quasi-Newton methods.

In the context of this dissertation previous directions do not only need to be stored, but also transported around the surface, which again requires solving of the geodesic and logarithmic problem on the manifold. Thereby working with a limited amount of stored data we also reduce the number of calculations required to transport this data. Additionally we consider this data to get more unreliable in further iterations, due to the manifold structure.

With respect to using only recent updates, among others, one can update in each step the approximate to the Hessian. In [BS21] Siegel and Buhmann introduced a limited memory BFGS algorithm with very satisfying results regarding efficiency. In regard to the method presented here on the other hand updating the Hessian will require transporting a complete basis of the tangent space along the manifold, furthermore it does not ensure that old directions are omitted after a certain number of iterations. Thereby we prefer to use limited memory methods which store less data and recalculate the approximate of the Hessian at each step.

We will consider the following L-BFGS algorithm with $B_0 = \eta I$

$$B_k = B_0 - \sum_{i=0}^{k-1} a_i a_i^T + \sum_{i=0}^{k-1} b_i b_i^T, \tag{2.8}$$

with a_i, b_i defined by

$$\begin{aligned} a_i &= \frac{B_i \delta_i}{\sqrt{\delta_i^T H_i \delta_i}}, \\ b_i &= \frac{\gamma_i}{\sqrt{\gamma_i^T \delta_i}}. \end{aligned} \tag{2.9}$$

One can also have η to be dependent on k, often chosen to be $\eta_k = \frac{\delta_{k-1}^T \gamma_{k-1}}{||\gamma_{k-1}||_2^2}$. Then there exists the following alternative representation of $B_k = H_k^{-1}$.

$$\begin{aligned} H_k =& (V_0 \cdots V_{k-1})^T H_0 (V_0 \cdots V_{k-1}) \\ &+ \frac{1}{\gamma_0^T \delta_0} (V_1 \cdots V_{k-1})^T \delta_0 \delta_0^T (V_1 \cdots V_{k-1}) \\ &+ \frac{1}{\gamma_1^T \delta_1} (V_2 \cdots V_{k-1})^T \delta_1 \delta_1^T (V_2 \cdots V_{k-1}) \\ &+ \cdots + \frac{1}{\gamma_{k-1}^T \delta_{k-1}} \delta_{k-1} \delta_{k-1}^T, \end{aligned} \tag{2.10}$$

where $V_i := I - \frac{1}{\gamma_i^T \delta_i} \gamma_i \delta_i^T$, also one should consider pre calculating and storing $\xi_i = \frac{1}{\gamma_i^T \delta_i}$. Limiting the Update-Formula to ℓ vectors, one gets

$$\begin{aligned} H_k =& (V_{k-\ell} \cdots V_{k-1})^T H_0 (V_{k-\ell} \cdots V_{k-1}) \\ &+ \xi_{k-\ell} (V_{k-\ell+1} \cdots V_{k-1})^T \delta_{k-\ell} \delta_{k-\ell}^T (V_{k-\ell+1} \cdots V_{k-1}) \\ &+ \xi_{k-\ell+1} (V_{k-\ell+2} \cdots V_{k-1})^T \delta_{k-\ell+1} \delta_{k-\ell+1}^T (V_{k-\ell+2} \cdots V_{k-1}) \\ &+ \cdots + \xi_{k-1} \delta_{k-1} \delta_{k-1}^T, \end{aligned} \tag{2.11}$$

This represents the update of the BFGS-algorithm with ℓ previous iterations, but in [EM12] a convenient two-loop algorithm to compute $r = B_k^{-1} z = H_k z$ in exactly this situation is introduced. This gives an algorithm which relies on recalculating an approximation to the Hessian in each iteration instead of updating one. Therefore not even the approximation to the Hessian but only the descent directions and gradients from the previous ℓ iterations are required. The algorithm is given in Algorithm 2.2.2, which is then applied to $-\nabla L(x_k)$ to calculate the next descent direction.

Algorithm 2.2.2: LimMemDescent

Data: Initial scaling $\eta \in \mathbb{R}$, vector $z \in \mathbb{R}^n$,
previous descent directions $\Delta = (\delta_i)_{i=1}^k$,
previous gradient differences $\Gamma = (\gamma_i)_{i=1}^k$.
Result: $r = B_k^{-1} z$.
Function: LimMemDescent(η, Δ, Γ):

$q \leftarrow z$
for $1 \leq i \leq k$ **do**
  $j \leftarrow k - t + 1$
  $\alpha_j \leftarrow \frac{\delta_j^T q}{\gamma_j^T \delta_j}$
  $q \leftarrow q - \alpha_j \gamma_j$
end
$r \leftarrow \eta^{-1} q$
for $1 \leq i \leq k$ **do**
  $\beta \leftarrow \frac{\gamma_i^T r}{\gamma_i^T \delta_i}$
  $r \leftarrow r + (\alpha_i - \beta)\delta_i$
end
return r

2.3 Line Search

The previous sections have introduced ways to calculate the descent direction d_k in (2.1). Now given d_k one still has to determine how far one searches in such a given direction. This is determined by the *step length* $\alpha_k \in \mathbb{R}$. In theory and best case one performs a so called *perfect line search*. Then α_k is chosen as the next or even global minimum

$$\underset{\alpha \in \mathbb{R}_+}{\operatorname{argmin}}\ g_k(\alpha), \tag{2.12}$$

with $g_k = L(x_k + \alpha d_k)$. This is of course a one dimensional optimisation problem itself and thus a perfect line search is likely impossible or very costly itself. Instead one switches to inexact line search methods, which state conditions under which a step length α is accepted.

With Quasi-Newton methods, like those of the Broyden class, often the *Armijo-Wolfe conditions* are used. They were firstly introduced in [Arm66] and later expanded in [Wol69].

Definition 2.3.1
A step length α_k is said to satisfy the *Armijo conditions*, associated to the search direction d_k, if it subjects to the following inequalities:

(A1) $L(x_k + \alpha_k d_k) \leq L(x_k) + c_1 \alpha_k d_k^T \nabla L(x_k)$

(A2) $-d_k^T \nabla L(x_k + \alpha_k d_k) \leq -c_2 d_k^T \nabla L(x_k) = c_2 \left| d_k^T \nabla L(x_k) \right|$

with $0 < c_1 \ll c_2 < 1$. It is to note that the assumption $d_k^T \nabla f(x_k) < 0$ is enforced by the optimisation method.

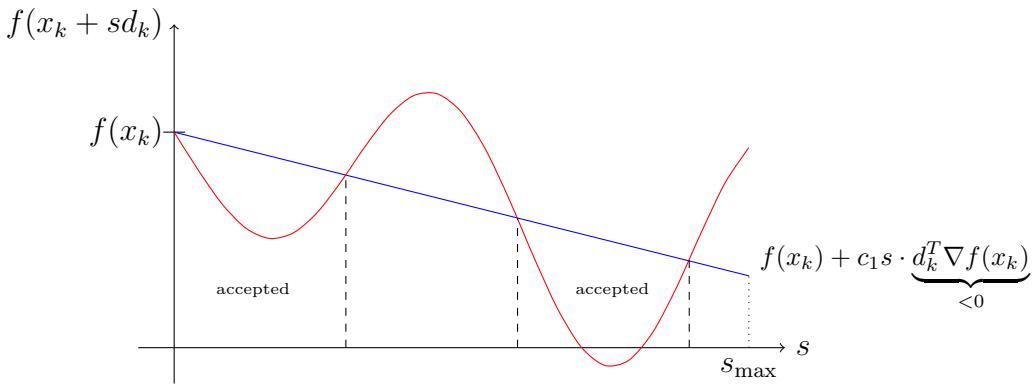


Figure 2.1: Visualization of Armijo condition A1)

One can replace (A2) with the *strong Wolfe condition*

(W2) $|d_k^T \nabla L(x_k + \alpha_k d_k)| \leq c_2 |d_k^T \nabla L(x_k)|$

which typically forces a_k to be closer to a minimum of $g_k(\alpha)$.

The Armijo conditions alone do not determine the step length α_k, it only gives criteria by which a step length is accepted. Instead one starts with a maximal step length s_{max} and iteratively reduces the step length by a constant factor $0 < \beta < 1$. So for $\ell \in \mathbb{N}$ one sets

$$s_{k,\ell} = \beta^\ell s_{max}, \tag{2.13}$$

and then checks if $x_k + s_{k,\ell} d_k$ satisfies conditions (A1) and (A2). Once satisfied the current value $s_{k,\ell}$ is returned as the step length α_k, elsewise ℓ is increased by one. This approach is useful if evaluating L and ∇L itself is costly and a reduction of evaluations is desirable.

Algorithm 2.3.2: Unconstrained Armijo

Data: Starting point x_k, search direction d_k.

Global: Maximal step length s_{max}, objective function L, gradient of the objective function ∇L, Armijo variables $0 < c_1 < c_2 < 1$, decline rate $0 < \beta < 1$.

Result: Step size $s \leq s_{\max}$ such that (A1) and (A2) hold.

Function: Armijo(x_0, d_k):

$k \leftarrow 0$
$\kappa \leftarrow d_k^T \nabla L(x_k)$
$m \leftarrow L(x_k)$
while *True* **do**
    $s \leftarrow s_{max} \cdot \beta^k$
    if $L(x_k + sd_k) < m + sc_1\kappa$ *and* $-d_k^T \nabla L(x_k + sd_k) \leq c_2\kappa$ **then**
        return s
    end
    $k \leftarrow k + 1$
end

In the unconstrained case points are chosen in a one dimensional affine subspace of $\mathbb{R}^n$ and therefore the points $x_k + s_{k,l}d_k$ are easily calculated. In the constrained case on the other hand one has to admit the line search to the constraints, which in the presented algorithms enforces calculation of new points by approximating a differential equation. Therefore points will be calculated beforehand to a maximum distance and are discrete. Then those points have to be checked by (A1) and (A2). This might result in calculation of unnecessary points, which may be inefficient compared to evaluating the cost function. Additionally further memory will be required to store those precalculated points. The number of points stored can be further reduced by storing only those points which will actually be checked with the Armijo conditions. The following proposition gives an upper bound on the number of points needed to store and therefore also an upper bound on the memory required.

Proposition 2.3.3
Let $P = \{p_1, \dots, p_m\} \subset \mathbb{R}^n$ be a sequence of points along a curve. Further let $0 < \beta < 1$ be the decline at which those points are checked with the Armijo conditions, then

$$-\left\lceil \frac{\ln(m) + \ln(1-\beta)}{\ln(\beta)} \right\rceil \leq k \leq \min\left(m-1, \left\lceil \frac{\ln(2) - \ln(m)}{\ln(\beta)} \right\rceil\right) \tag{2.14}$$

points have to be checked and additionally stored.

Proof. A point p_i is only to be checked and stored if $i = \lceil \beta^k \cdot m \rceil$ for some $k \in \mathbb{N}$. The last point to be checked for the Armijo condition is p_2, thus to find an upper bound one simply has to find $k \in \mathbb{N}$ minimal such that $m\beta^{k+1} < 2$. Then as $\beta < 1$ this is equivalent to

$$k + 1 > \log_\beta\left(\frac{2}{m}\right) = \frac{\ln(2) - \ln(m)}{\ln(\beta)} \Leftrightarrow k + 1 = \left\lceil \frac{\ln(2) - \ln(m)}{\ln(\beta)} \right\rceil$$

considering that at most $m - 1$ iterates can be additionally stored this gives as an upper bound

$$k = \min\left(m, \left\lceil \frac{\ln(2) - \ln(m)}{\ln(\beta)} \right\rceil\right) - 1. \tag{2.15}$$

This is only an upper bound, as $\lceil m\beta^i \rceil = \lceil m\beta^{i+1} \rceil > 1$ is possible for some $i \in \mathbb{N}$. In this case thought $m\beta^i - m\beta^{i+1} < 1$ which is equivalent to

$$i \geq \left\lceil -\frac{\ln(m) + \ln(1-\beta)}{\ln(\beta)} \right\rceil = -\left\lceil \frac{\ln(m) + \ln(1-\beta)}{\ln(\beta)} \right\rceil + 1.$$

Therefore for $i \leq -\left\lceil \frac{\ln(m)+\ln(1-\beta)}{\ln(\beta)} \right\rceil$ holds $\lceil m\beta^i \rceil \neq \lceil m\beta^{i+1} \rceil$.

□

An additionally problem might occur when an actual minimum in problem (2.12) lies just between two steps $s_{k,l}$ and $s_{k,l-1}$ of which $s_{k,l}$ is accepted. A later iteration of the optimisation method might therefore intersect the current line search (compare figure 4.1). This is particularly problematic on manifolds, as in general parallel transport of an element of the tangent space along a closed curve does not return the initial vector (compare Subsection 4.5.5). Also when a line search is getting close to the starting point Algorithm 2.3.2 can generate arbitrary small steps for α_k. For a line search by approximation this is not easily accessible as movement along a geodesic will be represented by discrete data. Therefore the algorithm can not check arbitrary close points to the starting point without redoing a line search. For this case we will store the point with the highest decrease in the objective function and return this point on failure instead. On failure of returning a

point by the Armijo conditions we will always initiate a restart of the method at the current iterate.

Algorithm 2.3.4: Armijo with discrete line search algorithms

Data: Search direction d_k, sequence of points $P = \{p_1 = x_k, \ldots, p_{|P|}\}$ generated by a curve approximation.

Global: Objective function L, gradient of the objective function ∇L, Armijo variables $0 < c_1 \leq c_2 < 1$, decline rate $0 < \beta < 1$.

Result: Index s such that p_s satisfies (A1) and (A2) or such that $f(p_s)$ is smallest among those points tested if non satisfies (A1) and (A2).

Function: ManArmijo(P):

1 $k \leftarrow 0$
2 $\kappa \leftarrow d_k^T \nabla L(x_k)$
3 $m \leftarrow f(x_k)$
4 min $\leftarrow m$
5 $s_{fail} \leftarrow 1$
6 **while** *True* **do**
7 $s \leftarrow \min(\lceil |P| \cdot \beta^k \rceil, s-1)$
8 **if** $s = 1$ **then**
9 **return** s_{fail}
10 **else if** $f(p_s) <$ min **then**
11 $s_{fail} \leftarrow s$
12 $min \leftarrow f(p_s)$
13 **if** $f(p_s) < m + sc_1\kappa$ *and* $-d_k^T \nabla f(p_s) \leq c_2\kappa$ **then**
14 **return** s
15 **end**
16 $k \leftarrow k + 1$;
17 **end**

In Algorithm 2.3.2 the maximal step length is considered a global variable as it might be externally adjusted by the optimisation method itself. In Algorithm 2.3.4 we consider an approximation to the curve on which the line search is to be done. The length of the sequence of points this approximation gives can be considered a maximal step length. Also line 7 of Algorithm 2.3.2 is formulated in such a way, that the same point is not checked multiple times.

Chapter 3

Ordinary Differential equations

Movement on a manifold is more complex compared to movement in a Euclidean space. A line search will always include a starting point and a descent direction - which will in this chapter also be referred to by *velocity*. In the most coveted case one has an explicit formula which describes the path of such a movement. This then gives a function in only one variable - namely the duration of the movement or the *time*. Having such a formula makes calculations simpler and one can do the line search by a one dimensional optimization algorithm (compare Section 2.3). But often one does not have such a formula, maybe one even lacks an explicit description of the manifold itself. We must assume some kind of description of the manifold is given. In this dissertation we will later assume this to be an implicit description of manifolds as *varieties*. Having an implicit description of the manifold one can set up an ordinary differential equations of the form

$$\ddot{y}(t) = f(t, y(t), \dot{y}(t)) \tag{3.1}$$

to which a curve y on the manifold must oblige. For the line search equation (3.1) will be expanded with a starting point $p = y(t_0)$ and an initial velocity $\dot{y}(t_0)$ altogether forming an *initial value problem* - if not stated otherwise we will without loss generality assume $t_0 = 0$. For the parallel transport of old direction along these curves so called ladder schemes will be used. They require solving the logarithmic problem, that is finding the initial velocity $v = \dot{y}(t)$ such that for given $p, q \in \mathcal{M}$ the respectable solution of (3.1) gives $y(0) = p$ and $y(1) = q$. This turns (3.1) into a *boundary value problem*. Solutions to boundary value problems will be approximated by a shooting

method, which tries to solve the boundary value problem with the given starting point and an arbitrary, or guessed, starting velocity v_0. Over multiple iterations a corrector is applied to the previous starting velocity v_{k-1} until $||y_{v_k}(1) - q||$ is sufficiently small.
Also (3.1) can and will in this dissertation be rewritten into a first order differential equation by doubling the dimension. Instead of

$$y : I \to \mathbb{R}^n,\ t \mapsto y(t) \tag{3.2}$$

we will consider the function

$$\tilde{y} : I \to \mathbb{R}^{2n},\ t \mapsto \begin{pmatrix} y(t) \\ \dot{y}(t) \end{pmatrix} \tag{3.3}$$

and thereby turning (3.1) into

$$\dot{\tilde{y}} = \tilde{f}\left(\tilde{y}\right) = \begin{pmatrix} \dot{y} \\ f(t, y, \dot{y}) \end{pmatrix}. \tag{3.4}$$

Additionally the differential equation encountered in this dissertation will not be explicitly depend on t and together with (3.4) one can replace (3.1) without loss of generality by

$$\dot{y} = f(y). \tag{3.5}$$

Now a short introduction to the later applied numerical solvers will be given.

3.1 Initial Value Problem

3.1.1 Euler's Method

The most basic numerical solver for the initial value problem (3.5) with starting point $y_0 = y(t_0)$ is the (forward) Euler method. It uses only the derivative at the current iterate. One approximates y by its first order Taylor polynomial

$$y(t_i + t) = y(t_i) + (t - t_i) \cdot \dot{y}(t_i) + \mathcal{O}(t - t_i). \tag{3.6}$$

Substituting (3.5) in (3.6) then gives recursively for $n \in \mathbb{N}$ the temporal equidistant approximates

$$y(t_0 + nh) \approx y_n = y_{n-1} + hf(t, y_{n-1}). \tag{3.7}$$

3.1.2 Multistep Methods

In general a multistep method is a numerical solver defined by taking into account multiple previous approximates $y_i \approx y(t_i)$ and their respective time. We will only consider equidistant time steps thus $t_i = t_0 + ih$. The general form of an s-step method is then given by

$$\sum_{j=0}^{s} a_j y_{n+j} = h \sum_{j=0}^{s} b_j f(t_{n+j}, y_{n+j}). \tag{3.8}$$

We will later use the explicit 5-step Adam-Bashford method. The respective coefficients are specified by setting $a_s = 1, a_1 = \cdots = a_{s-1} = 0, a_0 = -1$, $b_s = 0$ and substituting f by its approximation by Lagrange interpolation. This gives for $0 \leq j \leq s-1$

$$b_j = \frac{(-1)^j}{j!(s-j)!} \int_0^1 \prod_{\substack{i=0 \\ i \neq j}}^{s} (u+i) \, \mathrm{d}u. \tag{3.9}$$

The respective coefficients for the 5-step Adam-Bashford method are now stated in the following algorithm.

Algorithm 3.1.1: AB5

Data: Ordinary differential equation $\ddot{\gamma}(t) = f(\gamma(t), \dot{\gamma}(t))$, previous approximates $P = [p_0 \approx \gamma(t_{n-4}), \ldots, p_4 \approx \gamma(t_n)]$, $V = [v_0 \approx \dot{\gamma}(t_{n-4}), \ldots, v_4 \approx \dot{\gamma}(t_n)]$.

Result: Approximates $p_5 \approx \gamma(t_{n+1}), v_5 \approx \dot{\gamma}(t_{n+1})$.

Global: Step length $h \in \mathbb{R}$.

Function: $\mathrm{AB5}(f, p_0, \ldots, p_4, v_0, \ldots, v_4, h) = AB5(f, P, V, h)$:

for $0 \leq i \leq 4$ **do**

  $t_i \leftarrow f(p_i, v_i)$

end

$$\text{step} \leftarrow \frac{1901}{720}\begin{pmatrix} v_4 \\ t_4 \end{pmatrix} - \frac{2774}{720}\begin{pmatrix} v_3 \\ t_3 \end{pmatrix} + \frac{2616}{720}\begin{pmatrix} v_2 \\ t_2 \end{pmatrix} - \frac{1274}{720}\begin{pmatrix} v_1 \\ t_1 \end{pmatrix} + \frac{251}{720}\begin{pmatrix} v_0 \\ t_0 \end{pmatrix}$$

$$\begin{pmatrix} p_5 \\ v_5 \end{pmatrix} \leftarrow \begin{pmatrix} p_4 \\ v_4 \end{pmatrix} + h \cdot \text{step}$$

return (p_5, v_5)

For an s-step algorithm previous directions are calculated by algorithms that require less steps. We will initially use the forward Euler method for the first step and then incrementally increase the steps required from the 2-step

Adam-Bashford to the 4-step Adam-Bashford to get the setup for Algorithm 3.1.1.

Algorithm 3.1.2: AB5Setup

Data: Ordinary differential equation $\ddot{\gamma}(t) = f(\gamma(t), \dot{\gamma}(t))$,
$p = \gamma(0), v = \dot{\gamma}(0) \in \mathbb{R}^n$

Result: $p_1 \approx \gamma(h), \ldots, p_4 \approx \gamma(4h), v_1 \approx \dot{\gamma}(h), \ldots, v_4 \approx \dot{\gamma}(4h)$.

Global: step length $h \in \mathbb{R}$

Function: AB5Setup(f, p_0, v_0, h) :

`/* Euler */`

1 $t_0 \leftarrow f(p_0, v_0)$

2 $\begin{pmatrix} p_1 \\ v_1 \end{pmatrix} \leftarrow \begin{pmatrix} p_0 \\ v_0 \end{pmatrix} + h \begin{pmatrix} v_0 \\ t_0 \end{pmatrix}$

`/* 2-Step Adam-Bashford */`

3 $t_1 \leftarrow f(p_1, v_1)$

4 $\begin{pmatrix} p_2 \\ v_2 \end{pmatrix} \leftarrow \begin{pmatrix} p_1 \\ v_1 \end{pmatrix} + \frac{3h}{2} \begin{pmatrix} v_1 \\ t1 \end{pmatrix} - \frac{h}{2} \begin{pmatrix} v_0 \\ t_0 \end{pmatrix}$

`/* 3-Step Adam-Bashford */`

5 $t_2 \leftarrow f(p_2, v_2)$

6 $\begin{pmatrix} p_3 \\ v_3 \end{pmatrix} \leftarrow \begin{pmatrix} p_2 \\ v_2 \end{pmatrix} + \frac{23h}{12} \begin{pmatrix} v_2 \\ t_2 \end{pmatrix} - \frac{4h}{3} \begin{pmatrix} v_1 \\ t_1 \end{pmatrix} + \frac{5h}{12} \begin{pmatrix} v_0 \\ t_0 \end{pmatrix}$

`/* 4-Step Adam-Bashford */`

7 $t_3 \leftarrow f(p_3, v_3)$

8 $\begin{pmatrix} p_4 \\ v_4 \end{pmatrix} \leftarrow \begin{pmatrix} p_3 \\ v_3 \end{pmatrix} + \frac{55h}{24} \begin{pmatrix} v_3 \\ t_3 \end{pmatrix} - \frac{59h}{24} \begin{pmatrix} v_2 \\ t_2 \end{pmatrix} + \frac{37h}{24} \begin{pmatrix} v_1 \\ t_1 \end{pmatrix} - \frac{3h}{8} \begin{pmatrix} v_0 \\ t_0 \end{pmatrix}$

9 **return** $P = [p_0, \ldots, p_4], V = [v_0, \ldots, v_4]$

Algorithm (3.1.2) and (3.1.1) also make it obvious that storing and passing the temporary variables $t_i = f(y_i)$ makes calculations faster, while doubling the required memory.

3.2 Boundary Value Problem

Initial value problems specify the behaviour of the solution at its starting point, therefore one is able to approximate a solution from that point onward. Boundary value problems lack this initial information and are defined by

conditions imposed at multiple points on the considered interval. In general this will be a starting and an endpoint to which the solution has to oblige.

3.2.1 Shooting Method

A common method to approximate a boundary value problem is the *shooting method*. Over multiple iterations one considers the initial value problem defined by (3.1), initial starting point $\gamma(0) = p$ and a guess for the initial velocity $\dot{\gamma}(0) = v_0$. In each iteration v_k is then adjusted by some correction method until an approximation criteria is satisfied. In [Ha01] a correction method by the Newton–Raphson method is introduced for one dimensional differential equations. In the following the same approach is used but generalised to higher dimensions.
Consider the boundary value problem

$$\begin{aligned} \ddot{\gamma} &= f(t, \gamma, \dot{\gamma}) \\ \gamma(0) &= p \\ \gamma(1) &= q. \end{aligned} \tag{3.10}$$

We use a solver for the initial value problem given by the same differential equation but with initial values $\gamma(0) = x$ and and an arbitrary initial velocity $\dot{\gamma}(0) = v$. Thereby we create the initial value problem

$$\begin{aligned} \ddot{\gamma} &= f(t, \gamma, \dot{\gamma}) \\ \gamma(0) &= x \\ \dot{\gamma}(0) &= v. \end{aligned} \tag{3.11}$$

The given approximation $\tilde{y} \approx \gamma(1)$ will then be used to make a correction to the initial velocity v. To apply the Newton-Raphson method a Jacobi matrix has to be calculated, which requires a second initial value problem to be solved parallelly.
To do so in higher dimensions we now assume γ to be a function in the initial velocity and thus obtain the function $\gamma(t, v)$. Thus we want to find a solution to the equation $\gamma(1, v) - y = 0$. By Newton's method we update a guess v_k by

$$v_{k+1} = v_k - (J_v\gamma)(1, v_k)^{-1} \left(\gamma(1, v_k) - y\right), \tag{3.12}$$

with Jacobian-Matrix $J_v\gamma$ and derivatives with respect to v. Now consider component-wise

$$\begin{aligned}
&J_v(\ddot{\gamma})(t)_{i,j} = \partial_{v_j}\ddot{\gamma}_i \\
&\quad = \partial_{v_j} f_i(t, \gamma(t,v), \dot{\gamma}(t,v)) \\
&\quad = \sum_{k=1}^{n} \left(\partial_{x_k} f_i\right)(t,\gamma,\dot{\gamma}) \frac{\partial x_k}{\partial v_j} + \sum_{k=1}^{n} \left(\delta_{\gamma_k} f_i\right)(t,\gamma,\dot{\gamma}) \frac{\partial \gamma_k}{\partial v_j} + \sum_{k=1}^{n} \left(\partial_{\dot{\gamma}_k} f_i\right)(t,\gamma,\dot{\gamma}) \frac{\partial \dot{\gamma}_k}{\partial v_j} \\
&\quad = \sum_{k=1}^{n} \left(J_\gamma f\right)_{i,k} \left(J_v\gamma\right)_{k,j} + \sum_{k=1}^{n} \left(J_{\dot{\gamma}} f\right)_{i,k} \left(J_v\dot{\gamma}\right)_{k,j}.
\end{aligned} \tag{3.13}$$

In the last equality of (3.13) we used that t and v are independent variables by assumption. From (3.13) one now can derive

$$J_v\ddot{\gamma} = (\ddot{J_v\gamma}) = J_\gamma f \cdot J_v\gamma + J_{\dot{\gamma}} f \cdot (\dot{J_v\gamma}). \tag{3.14}$$

Substituting $z = J_v\gamma$ in (3.14) one gets the differential equation

$$\ddot{z} = J_\gamma f \cdot z + J_{\dot{\gamma}} f \cdot \dot{z}. \tag{3.15}$$

The initial values in (3.11) by construction now induce initial values

$$z(0) = (J_v\gamma)(0) = 0 \text{ and } \dot{z}(0) = (J_v\dot{\gamma})(0) = I. \tag{3.16}$$

for the differential equation (3.15), thereby creating a secondary initial value problem.

$$\begin{aligned}
\ddot{z} &= J_\gamma f \cdot z + J_{\dot{\gamma}} f \cdot \dot{z} \\
z(0) &= 0 \\
\dot{z}(0) &= I.
\end{aligned} \tag{3.17}$$

Now one has to solve this secondary initial value problem parallel to the primary initial value problem (3.11). The secondary initial value problem will then return an approximation to $\left((J_v\gamma)(1,v_k)\right)^{-1}$, which can be used to calculate a correction to the current starting velocity v_k.

Algorithm 3.2.1: CorStep

Data: Approximate $p \approx \gamma(t_i) \in \mathbb{R}^n$ of the initial value problem 3.11, approximates $P \approx z(t_i), V \approx \dot{z}(t_i) \in \mathbb{R}^{n \times n}$ of the secondary initial value problem 3.17.

Result: Approximate $w \approx J_v \ddot{\gamma}(t_i)$.

Global: ODE $\ddot{\gamma}(t) = g(\gamma(t), \dot{\gamma}(t))$ with Jacobians $J_\gamma g, J_{\dot{\gamma}} g$.

Function: CorStep(p, P, V):

1 $J_1 \leftarrow J_\gamma g(p)$
2 $J_2 \leftarrow J_{\dot{\gamma}} g(p)$
3 $w \leftarrow J_1 \cdot P + J_2 \cdot V$
4 **return** w

One can then create a substitute initial value problem induced by (3.11) and (3.17) which is then used together with an ODE-solver to solve the initial value problems in parallel.

Algorithm 3.2.2: ODECorStep

Data: Approximates $p \approx \gamma(t_i), v \approx \dot{\gamma}(t_i) \in \mathbb{R}^n$ and approximates $P \approx z(t_i), V \approx \dot{z}(t_i) \in \mathbb{R}^{n \times n}$ of the substitute initial value problem induced by (3.11) and (3.17).

Result: Approximate $w \approx (\ddot{\gamma}(t_i) \mid J_v \ddot{\gamma}(t_i))$ to the substitute initial value problem of 3.11 and 3.15.

Global: ODE $\ddot{\gamma}(t) = g(\gamma(t), \dot{\gamma}(t))$ with Jacobians $J_\gamma g, J_{\dot{\gamma}} g$.

Function: ODECorStep$((p \mid I), (v \mid V_0))$:

1 $w \leftarrow g(p, v)$
2 $w \leftarrow (w \mid \text{CorStep}(p, P, V))$
3 **return** w

Solving the substitute initial value problem might actually lead to unnecessary calculations. Instead of solving the substitute problem one can also first solve the primary initial value problem (3.11) and check if the current velocity returns a sufficient approximation. A solution to the secondary initial value problem is only then to be solved when an approximation fails. Only in this case the points generated by the primary initial problem need to be stored and used for Algorithm 3.2.2. In the following we will compare the difference in memory of both approaches.

Assuming a step length $0 < h \leq 1$, then for solving the primary initial value problem (3.11) $\frac{1}{h}$ steps have to be calculated. Without loss of generality we assume $\frac{1}{h} \in \mathbb{N}$. Solving the secondary initial value problem (3.17) only on failure then requires the storage of all $\frac{1}{h}$ iterates or $n \cdot \frac{1}{h}$ numbers. Additionally a k-step ODE-solver will require storage of k iterates of dimension $2n^2$, totalling in $\frac{n}{h} + 2kn^2$ numbers.
Otherwise considering the substitute initial value problems all iterates need not to be stored outside of the ODE-solver itself. Although those approximates are now of dimension $2(n+n^2)$. Further use of memory hence depends only on the ODE-solver used. For a k-step solver therefore k iterates have to be stored totalling in $2n(n+1)k$ numbers.
Therefore the difference is

$$\frac{n}{h} + 2kn^2 - 2n(n+1)k = n\left(\frac{1}{h} - 2k\right).$$

Usually $\frac{1}{h} > 2k$. Therefore if the dimension is very big and storage is problematic solving the substitute problem is preferable. Evaluating the performance is difficult as it requires estimating the necessary number of iterations to accept a solution to the boundary value problem. On a failed attempt in both cases the same amount of calculations are necessary. An improvement of performance does only occur in the final iteration. In the application of the later developed algorithms often the initial guess or the first correction was accepted, highly favouring a sequenced approach.

Chapter 4

Introduction to Riemannian manifolds

In this chapter we will given an introduction to the core concept of the considered optimisation constraints. A Riemannian manifold is an important object in differential geometry and consists at its core out of two parts: a manifold and a Riemannian metric. A manifold is a space that represents a Euclidean space locally, but is in general not globally one. The Riemannian metric then allows us to endow this manifold with a geometry. This structure will allow us to define a concept of movement along the manifold, which we will later need to make single and multistep optimisation algorithm applicable and ensure that each iterate remains on the manifold. We will start with a short summary of the necessary concepts and definitions to discuss manifolds, starting by charts up to connections, geodesics and parallel transport.
With those theoretical structures given we will then continue by making manifolds applicable to a computer program. We do so by considering Riemannian manifolds, which are also affine varieties. This algebraic structure will then be used to develop implementable algorithms.

4.1 Charts and manifolds

A manifold is a space that is locally modelled like a Euclidean space. This property is represented by local diffeomorphisms $\phi : U \to \mathcal{M} \subset \mathbb{R}^n$ defined on open subsets $U \subset \mathbb{R}^m$, which we will call *charts*. Those charts then allow to carry the Euclidean analysis over to the manifold and develop all the necessary theory.

Convention 4.1.1
In this dissertation we will always assume $\mathcal{M}$ to be a connected subset of the Euclidean space $\mathbb{R}^n$ if not stated otherwise. This induces important topological structures, for example is $\mathcal{M}$ with the induced topology Hausdorff.

Definition 4.1.2
A *chart* of $\mathcal{M}$ is a pair (U, ϕ), with $U \subset \mathcal{M}$ open and $\phi : U \to V \subset \mathbb{R}^m$ is a homeomorphism. We call U the domain of ϕ and m the chart's *dimension*. For a given $x \in \mathcal{M}$ we call $\phi(p) = (x_1, \ldots, x_m)$ the *coordinates* of (U, ϕ).
Two charts $(U, \phi), (V, \psi)$ of $\mathcal{M}$ of dimensions n and m respectively, are compatible if $U \cap V = \emptyset$ or

(Ch1) $\phi(U \cap V)$ is an open set of $\mathbb{R}^n$;

(Ch2) $\psi(U \cap V)$ is an open set of $\mathbb{R}^m$;

(Ch3) $\psi \circ \phi^{-1} : \phi(U \cap V) \to \psi(U \cap V)$ is a $\mathcal{C}^\infty$-diffeomorphism.

Definition 4.1.3
Let $\mathcal{M} \subset \mathbb{R}^n$ and $\mathcal{A} = \{(U_i, \phi_i) | i \in I\}$ a family of compatible charts. We then call $\mathcal{A}$ an *atlas* of $\mathcal{M}$ if $\bigcup_{i \in I} U_i$ is a covering of $\mathcal{M}$.
Given two atlases $\mathcal{A}_1, \mathcal{A}_2$ of $\mathcal{M}$ we call them *compatible* if $\mathcal{A}_1 \cup \mathcal{A}_2$ is again an atlas.
If we assume $\mathcal{A}_1$ to be an atlas of $\mathcal{M}$ such that if $\mathcal{A}_2$ is an compatible atlas to $\mathcal{A}_1$ this implies $\mathcal{A}_2 \subseteq \mathcal{A}_1$ we call $\mathcal{A}_1$ *complete*. By unionising compatible atlases one can without loss of generality assume without loss of generality a given atlas to be complete.
A *(smooth) manifold* is now a pair $(\mathcal{M}, \mathcal{A})$ with a complete atlas $\mathcal{A}$ of $\mathcal{M}$.
In an atlas of a set $\mathcal{M}$ it is by property (Ch3) and $\mathcal{M}$ being connected not possible to have charts of different dimensions. Therefore we will define the *dimension of a manifold* $\mathcal{M}$ to be the dimension of the charts of $\mathcal{A}$.

Theorem 4.1.4
Let $\mathcal{M}$ be a manifold and $\mathcal{N} \subset \mathcal{M}$ open, then $\mathcal{N}$ is also a manifold.

Proof. Let $\mathcal{A}$ be an atlas of $\mathcal{M}$ and $(U, \varphi) \in \mathcal{A}$ a chart. Then $U \cap \mathcal{N}$ is open in $\mathcal{N}$ and the restriction $\varphi|_{\mathcal{N}} : \mathcal{N} \cap U \to \phi(U \cap \mathcal{N})$ forms a chart of $\mathcal{N}$. Thereby each chart of $\mathcal{M}$ induces a chart of $\mathcal{N}$. Applying this to each chart of $\mathcal{A}$ induces an atlas $\tilde{\mathcal{A}}$ of $\mathcal{N}$. □

Convention 4.1.5
If $\mathcal{M}$ is a manifold of dimension m, we will call $\mathcal{M}$ a m-manifold. Also in this chapter if not stated otherwise $\mathcal{M}$ will always be a m-manifold.

In general we will not really distinguish between a manifold and its set. From the context it should be obvious if one is referring to the set on which the

atlas is defined or the structure and atlas itself. By the assumption $\mathcal{M} \subset \mathbb{R}^n$ we also ensure that the dimension of $\mathcal{M}$ is always finite.
With charts we can now extend maps between manifolds to maps between the Euclidean spaces of the respective charts. Such a map can now be investigated for differentiability.

Definition 4.1.6
Let $\mathcal{M}, \mathcal{N}$ be manifolds and $f : \mathcal{N} \to \mathcal{M}$ a map. We say f is of class $\mathcal{C}^k$ if for all charts $(U, \phi), (V, \psi)$ of $\mathcal{N}$ and $\mathcal{M}$ respectively holds

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

is $\mathcal{C}^k$ and write $f \in \mathcal{C}^k(\mathcal{N}, \mathcal{M})$.

It is important to note that this definition does not depend on an explicit choice of charts, but only on the atlas in general. A lot of results in differential geometry can be proven without use of explicit charts, one calls this *coordinate-free*.
As we have no inherent Euclidean structure on the manifold itself we are also not able define the derivative of a function $c : \mathbb{R} \to \mathcal{M}$ by the classical approach, namely the limit of the differential quotient

$$c'(x) = \lim_{h \to 0} \frac{c(x+h) - c(x)}{h},$$

as the term $\frac{c(x+h)-c(x)}{h}$ itself is in general not well-defined. However we are are able to pull the Euclidean structure onto the manifold via the charts of $\mathcal{M}$.

Definition 4.1.7
Let $\mathcal{M}$ be a manifold, (U, ϕ) a chart of $\mathcal{M}$ and $p \in U$, furthermore let $c : I \to U \subset \mathcal{M}$ a differentiable curve on $\mathcal{M}$ such that $0 \in I$, $c(0) = p$. Then $\phi \circ c$ is a differential curve in $\mathbb{R}^m$ which we can derive at 0. Now let

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

be the set of all differential curves c on $\mathcal{M}$ passing p at $t = 0$. On C_p we define the equivalence relation $\sim$ by

$$c_1 \sim c_2 :\Leftrightarrow \frac{d}{dt}\phi \circ c_1(t)|_{t=0} = \frac{d}{dt}\phi \circ c_2(t)|_{t=0}.$$

A simple proof can show that this definition is coordinate free and independent of the choice of charts. It also allows one to define a vector space

structure on $C_p/\sim$.

Definition 4.1.8
Let $\mathcal{M}$ be a smooth manifold and $p \in \mathcal{M}$, then one defines the *tangent space* of p at $\mathcal{M}$ by

$$T_p\mathcal{M} := C_p/\sim = \{[c] \mid c \in C_p\},$$

with *tangent vectors* $[c] := \{x \in C_p | x \sim c\}$. The mapping

$$\Phi_p^\varphi : T_p\mathcal{M} \to \mathbb{R}^m : [c] \mapsto \frac{d}{dt}\phi \circ c(t)|_{t=0},$$

is then well-defined, bijective and allows the definition of a vector space structure on $T_p\mathcal{M}$ by

$$a[c_1] + b[c_2] := \left(\Phi_p^\varphi\right)^{-1} \left(a\Phi_p^\varphi([c_1]_\sim) + b\Phi_p^\varphi([c_2]_\sim)\right).$$

One can once again check that the induced structure is independent of the choice of the charts and therefore coordinate free.
The *tangent bundle*

$$T\mathcal{M} = \coprod_{p\in\mathcal{M}} T_p\mathcal{M} := \bigcup_{p\in\mathcal{M}} \{p\} \times T_xM$$

of $\mathcal{M}$ is now the disjoint union of all tangent spaces.

Even thought one often does not distinguish between a tangent vector ν as an element of $T_p\mathcal{M}$ or $T\mathcal{M}$ we later need to distinguish between different tangent spaces. In this case the projection $\pi : T\mathcal{M} \to \mathcal{M}$ will give the reference point of the tangent space, such that $\pi(\nu) = p$ if and only if $\nu \in T_p\mathcal{M}$. We then call p the *root* of v. The tangent bundle with the induced structure of $\mathcal{M}$ and each tangent space itself defines a smooth manifold, which makes it possible to define smooth mapping from $\mathcal{M}$ to $T\mathcal{M}$, namely vector fields.

Definition 4.1.9
A *vector field* X is a smooth mapping from $\mathcal{M}$ to $T\mathcal{M}$ such that $\pi \circ X$ is the identity. So

$$X : \mathcal{M} \to T\mathcal{M},\ x \mapsto X_p := X(p) \in T_p\mathcal{M}.$$

The set of all vector fields is denoted $\mathfrak{X}(\mathcal{M})$.

Another important structure induced by tangent vectors are the directional derivates of real valued functions defined on $\mathcal{M}$.

Definition 4.1.10
A scalar field on $\mathcal{M}$ is a smooth function $f : \mathcal{M} \to \mathbb{R}$. The set of all scalar fields on $\mathcal{M}$ will be denoted by $\mathfrak{F}(\mathcal{M})$.

4.1.1 Directional Derivative

As seen above one cannot simply define the derivative of a scalar field on $\mathcal{M}$ the same way one would define the derivative in an euclidean space. For a tangent vector $v \in T_p\mathcal{M}$ the element $p + h \cdot v$ is not defined and thus for a scalar field $f \in \mathfrak{F}(\mathcal{M})$ the limit

$$D_v(f) = \lim_{h \to 0} \frac{f(p + hv) - f(p)}{h}$$

is not defined. But by choosing a representative γ of $v = [\gamma] \in T_p\mathcal{M}$ one can instead define a derivative in the following way.

Definition 4.1.11
Let $v = [\gamma] \in T_p\mathcal{M}$ and $f \in \mathfrak{F}(\mathcal{M})$. Then

$$df_p(v) = df_p([\gamma]) := \left.\frac{d}{dt}\right|_{t=0} (f \circ \gamma)(t)$$

is the *directional derivative* of f in the direction $v \in T_p\mathcal{M}$.

The directional derivative is well-defined and again coordinate free. Actually one can enhance the definition of a directional derivate by vector fields.

Definition 4.1.12
Let $f \in \mathfrak{F}(\mathcal{M}), X \in \mathfrak{X}(\mathcal{M})$, then for every $p \in \mathcal{M}$ one has $X(p) \in T_p\mathcal{M}$ and letting $\gamma_p \in X(p)$ be a representative of $X(p)$, then one defines

$$X(f) : \mathcal{M} \to \mathbb{R},\ p \mapsto X(f)(p) := df_p(X(p)) = \left.\frac{d}{dt}\right|_{t=0} (f \circ \gamma_p)(t).$$

This was only a short and rough summary of the necessary vocabulary in differential geometry. Further details and proofs can be found in any introductory book about this topic, as [Bou14].

4.2 Embedded submanifolds

It is obvious that a Euclidean space $\mathbb{R}^n$ is globally, and also locally, diffeomorphic to itself and a chart is given just by the identity map. But we are

interested in optimisation in $\mathbb{R}^n$ with constraints. Explicitly we want to constrain functions to subsets of $\mathbb{R}^n$ which are again a manifold,we call such a manifold a *submanifold*.

Definition 4.2.1
Let $\mathcal{M}$ be a manifold and $\mathcal{N} \subset \mathcal{M}$ such that for each point $x \in \mathcal{N}$ there exists a chart (U, ϕ) of $\mathcal{M}$ such that

$$\mathcal{M} \cap U := \{x \in U \mid \phi(x) \in \mathbb{R}^d \times \{0\}^{m-d}\}.$$

Then $\mathcal{N}$ with the induced structure of $\mathcal{M}$ is called an *embedded submanifold* of $\mathcal{M}$. Additionally $d \in \mathbb{N}$ is uniquely defined and $\operatorname{codim}_{\mathcal{M}}(\mathcal{N}) = m - d$ is the codimension of $\mathcal{N}$ in $\mathcal{M}$.

An embedded submanifold $\mathcal{N} \subset \mathcal{M}$ inherits naturally the topological structure, such that the restriction $f|_{\mathcal{N}}$ of a smooth function $f : \mathcal{M} \to \mathbb{R}$ to $\mathcal{N}$ is again smooth.

Theorem 4.2.2
Let $\mathcal{M}$ be a subset of $\mathbb{R}^n$. Then the following statements are equivalent:

i) $\mathcal{M}$ is a smooth embedded submanifold of $\mathbb{R}^n$ of dimension $n - m$;

ii) For all $x \in \mathcal{M}$, there exists an open neighbourhood $V \subset \mathbb{R}^n$ of x and a smooth function $f : V \to \mathbb{R}^m$ such that the differential $Df(x) : \mathbb{R}^n \to \mathbb{R}^m$ has rank m and $V \cap \mathcal{M} = f^{-1}(0)$.

Furthermore, the tangent space at x is given by $T_x\mathcal{M} = \ker Df(x)$.

For a proof see [RS22, p. 34]. Thought the proof ii) implies i) follows also directly by the implicit function theorem.

Theorem 4.2.3 (Implicit Function Theorem)
Let $m, n \in \mathbb{N}, f \in \mathcal{C}^\infty(\mathbb{R}^{n+m}, \mathbb{R}^m)$ and $\tilde{x} = (a, b) \in \mathbb{R}^{n+m} = \mathbb{R}^n \times \mathbb{R}^m$ such that $f(\tilde{x}) = 0$. Furthermore let

$$J_{f,y}(a, b) = \left[\frac{\delta f_i}{\delta y_j}(a, b)\right] \tag{4.1}$$

be the Jacobian of f regarding the last m coordinates.
If $J_{f,y}(a, b)$ is invertible, then there exists an open neighbourhood $U \subset \mathbb{R}^n$ of a and a unique smooth function $g \in \mathcal{C}^\infty(U, \mathbb{R}^m)$, such that

i) $g(a) = b$;

ii) $f(x, g(x)) = 0$ for all $x \in U$.

For a proof see [RS22, p. 79].
Note that with the implicit function theorem the choice of functions is very important.

Example 4.2.4
A cone itself is not a manifold, as in the peak at the origin the surface is not smooth, but by widening the describing equations by an arbitrary value greater zero, one gets a manifold again. So let

$$\begin{aligned} f_1 &= x^2 + y^2 - z^2 + w^2 - 1 \\ f_2 &= x^2 + y^2 - z^2 - w^2 - 1. \end{aligned} \tag{4.2}$$

and consider the variety $\mathcal{M} = f_1^{-1}(0) \cap f_2^{-2}(0) = \mathcal{V}(f_1, f_2)$ (compare Definition 4.7.2). One can easily see that as ideals of polynomials

$$\langle f_1, f_2 \rangle = \langle w^2, x^2 + y^2 - z^2 - 1 \rangle. \tag{4.3}$$

This implies for the variety

$$\mathcal{V}(f_1, f_2) = \mathcal{V}(w^2, x^2 + y^2 - z^2 - 1) = \mathcal{V}(w, x^2 + y^2 - z^2 - 1). \tag{4.4}$$

Now set

$$\begin{aligned} g_1 &:= w \\ g_2 &:= x^2 + y^2 - z^2 - 1 \end{aligned} \tag{4.5}$$

and compare the gradients. On the one hand one has

$$\nabla f_1 = \begin{pmatrix} 2x \\ 2y \\ -2z \\ 2w \end{pmatrix}, \nabla f_2 = \begin{pmatrix} 2x \\ 2y \\ -2z \\ -2w \end{pmatrix}, \tag{4.6}$$

then for every $p = (x, y, z, w) \in \mathcal{M}$ one has $w = 0$ and thus $\nabla f_1(p) = \nabla f_2(p)$. On the other hand

$$\nabla g_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \nabla g_2 = \begin{pmatrix} 2x \\ 2y \\ -2z \\ 0 \end{pmatrix}. \tag{4.7}$$

As $(0, 0, 0, 0) \notin \mathcal{V}(f_1, f_2)$ one see that $\nabla f_1(p), \nabla f_2(p)$ are linearly independent for all $p \in \mathcal{V}(f_1, f_2)$.

4.3 Riemannian structure and gradients

By construction a tangent space is again a vector space isomorphic to a Euclidean space. This allows the introduction of the Riemannian metric as an inner product on the tangent space.

Definition 4.3.1
Let V be a vector space, then an *inner product* is a bilinear, symmetric, positive definite form $V \times V \to \mathbb{R},\ (v, w) \mapsto \langle v, w\rangle$, such that

(P1) $\langle \lambda v + \mu w, u\rangle = \lambda\langle v, u\rangle + \mu\langle w, u\rangle$,

(P2) $\langle w, v\rangle = \langle v, w\rangle$,

(P3) $\langle v, v\rangle \geq 0$, with $\langle v, v\rangle = 0$ if and only if $v = 0$,

holds for all $u, v, w \in V$,$\lambda, \mu \in \mathbb{R}$.
An inner product induces a norm $||v|| = \sqrt{\langle v, v\rangle}$ and thereby a metric on V.

Definition 4.3.2
Let $\mathcal{M}$ be a smooth manifold. A *Riemannian metric* g assigns smoothly to each $p \in \mathcal{M}$ an inner product

$$g_p : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R},\ (v, w) \mapsto g_p(v, w) := \langle v, w\rangle_p.$$

As the elements themself uniquely define the tangent space we will often omit the index.
A smooth manifold $\mathcal{M}$ equipped with a Riemannian metric g is then called a *Riemannian manifold*.

Thereby in a Riemannian manifold the map

$$p \mapsto g_p(X_p, Y_p),$$

for vector fields $X, Y \in \mathfrak{X}(\mathcal{M})$, is itself a smooth map. This allows the definition of a *gradient* of a scalar field on a Riemannian manifold, which indicates the direction of growth of the scalar field in the tangent space.

Definition 4.3.3
Let $\mathcal{M}$ be a Riemannian manifold and $f \in \mathfrak{F}(\mathcal{M})$ a scalar field. Then there exists a unique element $v_{p,f} \in T_p\mathcal{M}$ such that for all $w \in T_p\mathcal{M}$ holds

$$Df(p)[w] = \langle v_{p,f}, w\rangle_p.$$

The vector field $\text{grad}(f) : \mathcal{M} \to T\mathcal{M}\ p \mapsto v_{p,f}$ which maps each point to this unique vector is called *gradient* of f.

4.4 Riemannian submanifolds

A submanifold of a Riemannian manifold induces in a natural way also a Riemannian metric on this submanifold. As a Euclidean space forms a Riemannian manifold and the manifolds we consider are submanifolds of a Euclidean space this naturally gives us a Riemannian structure on our manifolds.

Definition 4.4.1
Let $(\mathcal{N}, \overline{g}), (\mathcal{M}, g)$ be Riemannian manifolds and $\mathcal{M}$ be a submanifold of $\mathcal{N}$. Then $(\mathcal{M}, g)$ is a *Riemannian submanifold* of $\mathcal{N}$ if for all $p \in \mathcal{M}$ and $v, w \in T_p\mathcal{M} \subset T_p\mathcal{N}$ holds

$$g_p(v, w) = \overline{g}_p(v, w).$$

As a subspace of $T_p\mathcal{N}$ the tangent space $T_p\mathcal{M}$ of a Riemannian manifold has a normal compliment in regard to the inner product. This defines again a subspace

$$N_p\mathcal{M} := T_p\mathcal{M}^\perp := \{v \in T_p\mathcal{N} \mid \langle v, w\rangle = 0 \ \forall v \in T_p\mathcal{N}\},$$

which we call the *normal space* of $\mathcal{M}$ in $\mathcal{N}$.
Naturally for $\mathcal{M}$ being a submanifold of the Riemannian manifold $(\mathcal{N}, \tilde{g})$, then by restriction $(\mathcal{M}, \tilde{g}|_{\mathcal{M}})$ is a Riemannian submanifold of $\mathcal{N}$.

Now for a Riemannian submanifolds all vectors $v \in T_p\mathcal{N}$ can be uniquely decomposed in orthogonal components

$$v = \mathrm{proj}_{T_p\mathcal{M}}(v) + \mathrm{proj}_{N_p\mathcal{M}}(v),$$

with $\mathrm{proj}_{T_p\mathcal{M}}, \mathrm{proj}_{N_p\mathcal{M}}$ the orthogonal projections to the tangent or normal space respectively.

Theorem 4.4.2
Let $\mathcal{N}$ be Riemannian manifold, $\mathcal{M} \subset \mathcal{N}$ a Riemannian submanifold and $\overline{f} : \mathcal{N} \to \mathbb{R}$ be a scalar field on $\mathcal{M}$. Then the restriction $f := \overline{f}|_{\mathcal{M}} : \mathcal{M} \to \mathbb{R}$ is a scalar field on $\mathcal{M}$ with

$$\mathrm{grad}\ f(p) = \mathrm{proj}_{T_p\mathcal{M}}\mathrm{grad}\ \overline{f}(p). \tag{4.8}$$

Proof. Indeed for all $v \in T_p\mathcal{M}$ holds

$$\begin{aligned} Df(p)[v] &= D\overline{f}(p)[v] \\ &= \langle \mathrm{grad}\ \overline{f}(p), v\rangle_p \\ &= \langle \mathrm{proj}_{T_p\mathcal{M}}\mathrm{grad}\ \overline{f}(p) + \mathrm{proj}_{N_p\mathcal{M}}\mathrm{grad}\ \overline{f}(p), v\rangle_p \\ &= \langle \mathrm{proj}_{T_p\mathcal{M}}\mathrm{grad}\ \overline{f}(p), v\rangle_p. \end{aligned}$$

□

4.5 Connections and Hessians

Let $\mathcal{M}$ be a Riemannian manifold and $X, Y \in \mathfrak{X}(\mathcal{M})$ be vector fields. Similarly to the derivative of a scalar field we want to define the derivate of Y at a point $p \in \mathcal{M}$ along a direction $X_p = X(p)$. If $\mathcal{M}$ would be a Euclidean space the definition would again be straightforward

$$DY(p)[X_p] = \lim_{t \to 0} \frac{Y(p + tX_p) - Y(p)}{t}.$$

Unfortunately $p + tX_p$ is again for an arbitrary manifold not even defined, and even assuming we would know how to handle this expression (which we later will), then $Y(p+t_pX_p)$ and $Y(p)$ are elements of different vector spaces, so the sum is again ill defined.
To make objects in different tangent spaces comparable one introduces the notion of connections. They will allow us to transport a tangent vector $v \in T_p\mathcal{M}$ along a curve $\gamma : I \to \mathcal{M}$ with $\gamma(0) = p$ to the tangent space $T_{\gamma(t)}\mathcal{M}$ for some $t \in I$. In Riemannian manifolds the connection of interest are the *Riemannian*, or *Levi-Civita connection*. They allow a coordinate free approach to define acceleration along a curve γ and enable us to talk about higher derivates, namely the Hessian.

Definition 4.5.1
Let $\mathcal{M}$ be a manifold, an *affine connection* ∇ on $\mathcal{M}$ is then a map

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

satisfying for $X, Y, Z \in \mathfrak{X}(\mathcal{M}), f, g \in \mathfrak{F}(\mathcal{M}), \lambda_1, \lambda_2 \in \mathbb{R}$

(C1) $\nabla_{f_1X_1+f_2X_2}Y = f_1\nabla_{X_1}Y + f_2\nabla_{X_2}Y$;

(C2) $\nabla_X(\lambda_1Y_1 + \lambda_2Y_2) = \lambda_1\nabla_X Y_1 + \lambda_2\nabla_X Y_2$;

(C3) $\nabla_X(fY) = X(f)Y + f\nabla_X Y$.

The vector field $\nabla_X Y$ is then called the *covariant derivative* of Y with respect to X in regard to the affine connection ∇. In this notation X depends only on $p \in \mathcal{M}$. Thereby one can also use this to define the covariant derivate pointwise for $v \in T_p\mathcal{M}$. Define

$$\nabla_v Y := \nabla_X Y(p),$$

for arbitrary $X \in \mathfrak{X}(\mathcal{M})$ such that $X(p) = v$.

For a manifold $\mathcal{M}$ there exist infinitely many connections, but assuming the following properties defines a unique affine connection.

Theorem 4.5.2
Let $\mathcal{M}$ be a Riemannian manifold. Then there exists a unique affine connection ∇ such that

(LC1) $\nabla_X Y - \nabla_Y X = [X, Y]$ and

(LC2) $Zg(X, Y) = g(\nabla_Z X, Y) + g(X, \nabla_Z Y)$,

for all $X, Y, Z \in \mathfrak{X}(\mathcal{M})$. Here for $X, Y \in \mathfrak{X}(\mathcal{M}), f \in \mathfrak{F}(\mathcal{M})$ we define

$$[X, Y]f = X(Y(f)) - Y(X(f)).$$

This connection is called *Levi-Civita connection* or *Riemannian connection.*

A proof of this can be found in [RS22, p. 171].

We can now use this to define a second order derivative on $\mathcal{M}$, the derivative of the gradient of a scalar field.

Definition 4.5.3
Given a scalar field $f \in \mathfrak{F}(\mathcal{M})$ on the Riemannian manifold $\mathcal{M}$ with Riemannian connection ∇, the *Riemannian Hessian* of f at $p \in \mathcal{M}$ is defined by the linear map Hess $f(p) : T_p\mathcal{M} \to T_p\mathcal{M}$ by

$$\text{Hess } f(p)(v) = \nabla_v \text{grad } f = (\nabla_X \text{grad } f)_p \,,$$

where $X \in \mathfrak{X}(\mathcal{M})$ is any vector field on $\mathcal{M}$ such that $X(p) = v$.

4.5.1 Connections and Hessians on Riemannian submanifolds

In the general case connections and Hessians are not easy to calculate. If the connection and Hessian are known for a manifold, it naturally induces the same structure in its Riemannian submanifold.

Theorem 4.5.4
Let $\mathcal{M}$ be a Riemannian submanifold of a Riemannian manifold $\mathcal{N}$ and let ∇ and $\overline{\nabla}$ denote the Riemannian connections on $\mathcal{M}$ or $\mathcal{N}$ respectively. Then for all $X, Y \in \mathfrak{X}(\mathcal{M})$

$$(\nabla_X Y)_p = \text{proj}_{T_p\mathcal{M}} \left(\overline{\nabla}_X Y\right)_p .$$

Especially if $\mathcal{N}$ is an Euclidean space, then

$$(\nabla_X Y)_p = \text{proj}_{T_p\mathcal{M}} \left(DY(p)(X_p)\right) .$$

So in this case the Riemannian connection can be simply calculated by the classical derivative in $\mathcal{N}$, followed by a projection onto the tangent space of $\mathcal{M}$. Even more for the Riemannian Hessian this implies

$$\text{Hess } f(p)(v) = \text{proj}_{T_p\mathcal{M}}\left(D_x|_{x=p}\left(\text{proj}_{T_x\mathcal{M}}\nabla f(x)\right)(v)\right),$$

which just translates to

1. compute the classical gradient;
2. project it to the tangent space;
3. compute the classical directional derivate of this projection;
4. project again to the tangent space.

4.5.2 Distances and geodesics

In Euclidean space, an optimisation method will calculate a descent direction, in which one will search for an improvement of the objective function. Thus one makes a line search in this direction until a point is found which satisfies the respective line search conditions (for example (A1) and (A2) of the Armijo condition) and repeats this process until a Minima is found. On a manifold the possible directions are the elements of the tangent space. Therefore we need to have an understanding of what moving in a direction along a manifold means. To do so we will generalize the concept of a straight line, which means along the manifold, the *acceleration* of the curve does not change.

Definition 4.5.5
Let $\gamma : I \to \mathcal{M}$ be a smooth curve and $V : I \to T\mathcal{M}$ such that $V(t) \in T_{\gamma(t)}\mathcal{M}$ for every $t \in I$, then we call V a *vector field along* γ. If $V \circ \gamma$ is smooth, we call V a smooth vector field along γ.

The most obvious example of a vector field along a curve γ is its velocity vector $\dot{\gamma}(t) \in T_{\gamma(t)}\mathcal{M}$. For a periodic curve or a non-periodic, non-self-intersecting curve γ this translates to $V \circ \gamma^{-1}$ being a smooth vector field on the embedded manifold $\gamma(I) \subset \mathcal{M}$. This allows us define the acceleration along a curve.

Definition 4.5.6
Let $\mathcal{M}$ be a smooth manifold equipped with a connection ∇. Furthermore

let $\gamma : I \to \mathcal{M}$ be a smooth, not self-intersecting curve on $\mathcal{M}$ and $X \in \mathfrak{X}(\mathcal{M})$ such that $X_{\gamma(t)} = \dot{\gamma}(t)$ for all $t \in I$. Then

$$t \mapsto \nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) := \nabla_{X_{\gamma(t)}}\dot{\gamma}(t) \in T_{\gamma(t)}\mathcal{M}$$

is the *acceleration* along γ. This definition can be extended to self-intersecting curves simply by defining the acceleration on a non-self-intersecting segment of the curve.

Definition 4.5.7
A curve $\gamma : I \to \mathcal{M}$ with $I \subset \mathbb{R}$ open is a geodesic if and only if $\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) = 0$ for all $t \in I$. We therefore define a geodesic to be a curve with no acceleration on all of its domain. Without loss of generality we also assume I to be maximal, in regard to this property.

Theorem 4.5.8 (Characterisation of geodesics)
Let $\mathcal{M}$ be a Riemannian manifold and $\gamma : I \to \mathcal{M}$ a smooth curve. Then the following are equivalent

(i) $\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) = 0$ for all $t \in I$,

(ii) the acceleration is normal to $\mathcal{M}$, that is $\ddot{\gamma}(t) \perp T_{\gamma(t)}\mathcal{M}$ for all $t \in I$.

A proof of this can be found in [RS22, p.177].

In the case of a Riemannian submanifold of an Euclidean space $\mathbb{R}^n$ this reduces to the simple form of

$$\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) = 0 \Leftrightarrow 0 = \mathrm{proj}_{T_{\gamma(t)}\mathcal{M}}\ddot{\gamma}(t) \tag{4.9}$$

for all $t \in I$, with $\ddot{\gamma}(t)$ is the classic second derivative of γ as a curve in $\mathbb{R}^n$. The Riemannian metric also allows us to define the length of a curve on $\mathcal{M}$ and thereby also introduce a concept of distance, which under sufficient requirements will actually turn out to be a metric.

Definition 4.5.9
Let $\gamma : [a, b] \to \mathcal{M}$ be a differentiable curve on the Riemannian manifold $(\mathcal{M}, g)$. Then

$$\mathrm{length}(\gamma) := \int_a^b \sqrt{g_{\gamma(t)}\left(\dot{\gamma}(t), \dot{\gamma}(t)\right)}\ \mathrm{d}t = \int_a^b ||\dot{\gamma}(t)||_{\gamma(t)}\ \mathrm{d}t$$

is the *length of the curve* γ.

Definition 4.5.10
Let $\mathcal{M}$ be a Riemannian manifold then we define *the Riemannian distance* on $\mathcal{M}$ by

$$\text{dist} : \mathcal{M} \times \mathcal{M} \to \mathbb{R},\ (p,q) \mapsto \text{dist}(p,q) := \inf_{\gamma \in \Gamma(p,q)} \text{length}(\gamma),$$

with $\Gamma(p,q) := \{\gamma : [0,1] \to \mathcal{M} \mid \gamma \in \mathcal{C}^1, \gamma(0) = p, \gamma(1) = q\}$.

Assuming $\mathcal{M}$ is Hausdorff the Riemannian distance actually defines a metric on $\mathcal{M}$.

Remark 4.5.11
It is not necessarily true that a geodesic is always the shortest path between two points. One considers the manifold $S^1 := \left\{x \in \mathbb{R} \mid x^T x = 1\right\}$ and the points $p = \begin{pmatrix}1\\0\end{pmatrix}$ and $q = \begin{pmatrix}0\\1\end{pmatrix}$. The geodesic induced by the tangent vector $v_1 = \begin{pmatrix}0\\1\end{pmatrix} \in T_pS^1$ is given by $\gamma_{v_1}(t) = \begin{pmatrix}\cos(t)\\\sin(t)\end{pmatrix}$ and gives indeed the shortest path from p to q while the curve induced by $v_2 = \begin{pmatrix}0\\-1\end{pmatrix} \in T_pS^1$ gives the geodesic $\gamma_{v_2}(t) = \begin{pmatrix}\cos(t)\\-\sin(t)\end{pmatrix}$, which is also a path from p to q of non-minimal length. One has $||\cdot||_{\gamma(t)} = ||\cdot||_2$ and therefore

$$||\dot{\gamma}_{v_i}(t)||_{\gamma(t)} = \cos(t)^2 + \sin(t)^2 = 1$$

for each $i \in \{1,2\}$ and $t \in \mathbb{R}$. Furthermore $\gamma_{v_1}\left(\frac{\pi}{2}\right) = q$ and $\gamma_{v_2}\left(\frac{3\pi}{2}\right) = q$, which indeed gives

$$\text{length}(\gamma_{v_2}) = \frac{3\pi}{2} > \frac{\pi}{2} = \text{length}(\gamma_{v_1}),$$

if we consider $\gamma_{v_1}, \gamma_{v_2}$ as curves from p to q.

In [RS22] another characteristic of geodesics can be found:

Theorem 4.5.12 (Characterization of geodesics II)
Let $\mathcal{M}$ be a Riemannian manifold, $I = [a,b] \subset \mathbb{R}$ a compact interval and $\gamma : I \to \mathcal{M}$ a smooth curve. Then the following are equivalent

(i) $\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) = 0$ for all $t \in I$,

(ii) γ is parametrized proportional to arc length, in explanation either γ is constant or γ is extremal of the length functional, that is every variation

$\{\gamma_s\}_{s\in\mathbb{R}}$ of γ with fixed endpoints satisfies

$$\left.\frac{d}{ds}\right|_{s=0} \text{length}(\gamma_s) = 0.$$

This Theorem is also proven in [RS22, p. 177].
Theorem 4.5.12(ii) now shows that locally a geodesic is the shortest distance between two points.

4.5.3 Exponential and logarithmic maps

By now we have established a way of understanding moving in a direction on a manifold. Also we can now define an expression that relates to $p+v$ for $p \in \mathcal{M}$ and $v \in T_p\mathcal{M}$ by using the related geodesic and a fixed duration.

Definition 4.5.13
Let $\mathcal{M}$ be a smooth manifold, ∇ a connection on $\mathcal{M}$ and $p \in \mathcal{M}$. For every $v \in T_p\mathcal{M}$ there exists a unique geodesic $\gamma_{p,v} : I \to \mathcal{M}$ with $\gamma_{p,v}(0) = p$ and $\dot{\gamma}_{p,v}(0) = v$, which satisfies for all $\lambda \in \mathbb{R}, t \in I$ such that $\lambda t \in I$

$$\gamma_{p,\lambda v}(t) = \gamma_{p,v}(\lambda t).$$

Then

$$\exp_p : T_p\mathcal{M} \to \mathcal{M},\ v \mapsto \gamma_{p,v}(1)$$

is called the *exponential map* at $x \in \mathcal{M}$.

As geodesics are only locally defined we note that there may exist some $p \in \mathcal{M}, v \in T_p\mathcal{M}$ such 1 is not an element of the preimage of $\gamma_{p,v}$. This means that the exponential may not be defined for all $v \in T_p\mathcal{M}$.

Definition 4.5.14
Let $\mathcal{M}$ be a smooth manifold, such that for all $p \in \mathcal{M}$ the exponential map $\exp_p$ is defined for all $v \in T_p\mathcal{M}$. Then we call $\mathcal{M}$ *geodesically complete*.

Definition 4.5.15
Let $\mathcal{M}$ be a smooth manifold, ∇ a connection on $\mathcal{M}$ and $p, q \in \mathcal{M}$. If there exists $v \in T_p\mathcal{M}$ such that $\exp_p(v) = q$, then we say v solves the logarithmic problem $\log(p, q)$ on $\mathcal{M}$.

The logarithmic problem is not necessarily solvable and if it is, it might not be uniquely solvable as Remark 4.5.11 indicates.

4.5.4 Parallel transport

In Euclidean space vectors at one point can be translated to another point by changing the root at which the vector is anchored. Therefore the root in a Euclidean space is in general omitted. But how does one compare the tangent vector $\begin{pmatrix}1\\0\\0\end{pmatrix} \in T_{\binom{0}{0}{1}}S^2$ to another tangent vector in a different tangent space $T_{\binom{1}{0}{0}}S^2$? Here even $\begin{pmatrix}1\\0\\0\end{pmatrix} \notin T_{\binom{1}{0}{0}}S^2$. If one does not use the injection $T_pS^2 \subset \mathbb{R}^3$ a comparison seems even less intuitive. To handle this some method is needed to translate vectors from one tangent space to another while preserving the geometry.

Definition 4.5.16
Let $\gamma : I \to \mathcal{M}$ be a smooth curve on the manifold $\mathcal{M}$. A vector field along γ is a smooth map $X : I \to T\mathcal{M}$ such that $X(t) \in T_{\gamma(t)}\mathcal{M}$ for every $t \in I$. The set of smooth vector fields along γ is a real vector space and will be denoted by

$$\mathrm{Vect}(\gamma) := \{X : I \to T\mathcal{M} \mid X \text{ is smooth and } X(t) \in T_{\gamma(t)}\mathcal{M}\, \forall t \in I\}.$$

Example 4.5.17
Let $\gamma : I \to \mathcal{M}$ be a geodesic on the manifold $\mathcal{M}$. Then $\frac{d}{ds}\gamma(s)|_{s=t} \in T_{\gamma(t)}\mathcal{M}$ and therefore

$$X : I \to T\mathcal{M},\ t \mapsto \dot{\gamma}(t)$$

is an element of $Vect(\gamma)$. Furthermore $\nabla_{\dot{\gamma}(t)}X(t) = 0$ for all $t \in I$. This is exactly the property one wishes to preserve in parallel transport and gives the following definition.

Definition 4.5.18
Let $\mathcal{M}$ be a manifold and $\gamma : I \to \mathcal{M}$ a smooth curve. A vector field X along γ is called *parallel* if

$$\nabla_{\dot{\gamma}(t)}X(t) = 0$$

for all $t \in I$.

A parallel vector field now satisfies the following property:

Theorem 4.5.19
Let $\mathcal{M}$ be a Riemannian manifold and $\gamma : I \to \mathcal{M}$ a smooth curve. Furthermore let $X, Y \in \text{Vect}(\gamma), t_0 \in I$ such that $g(X(t_0), Y(t_0)) = 0$. Then for all $t \in I$ holds $g(X(t), Y(t)) = 0$.

Proof. By property (LC2) of Definition 4.5.2 holds

$$\frac{d}{dt} g(X, Y) = g(\nabla_{\dot{\gamma}(t)} X, Y) + g(X, \nabla_{\dot{\gamma}(t)} Y) = 0. \tag{4.10}$$

□

Thus orthonormality of tangent vectors is preserved by parallel transport.
The following theorem then expands definition 4.5.18 to tangent vectors instead of vector fields.

Theorem 4.5.20
Let $\mathcal{M}$ be a manifold and $\gamma : I \to \mathcal{M}$ a smooth curve. Let $t_0 \in I$ and $v_0 \in T_{\gamma(t_0)}\mathcal{M}$ be given. Then there exists a unique parallel vector field $X \in \text{Vect}(\gamma)$ such that $X(t_0) = v_0$.

A proof of this can be found in [RS22, p. 245].

Definition 4.5.21
Let $\mathcal{M}$ be a Riemannian manifold and $\gamma : I \to \mathcal{M}$ a smooth curve, then we will denote the unique parallel vector field $X \in \text{Vect}(\gamma)$ of Theorem 4.5.20 by Π_γ^w.

4.5.5 Commutator

Given a Riemannian manifold $\mathcal{M}$, $x, y \in \mathcal{M}$ then the parallel transport of a tangent vector $v \in T_x\mathcal{M}$ to y is not unique, but depends on the curve along which the vector is transported. This is mostly observed in the parallel transport along a closed curve. Then a tangent vector is in general not mapped to itself again.

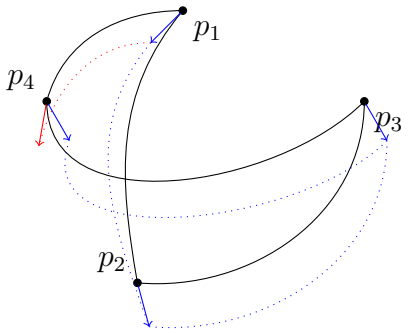


Figure 4.1: Parallel Transport along different curves.

Taking vector fields $V = \sum_{i=1}^{m} V_i \frac{\partial}{\partial x^i}, W = \sum_{i=1}^{m} W_i \frac{\partial}{\partial x^i} \in \mathfrak{X}(\mathcal{M})$ this also leads to a discrepancy when differentiating along those vector fields in alternating order. This discrepancy is given by the vector field

$$[V, W] = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} V_i \frac{\partial W_j}{\partial x_i} - W_i \frac{\partial V_j}{\partial x_i} \right) \frac{\partial}{\partial x_j}. \tag{4.11}$$

Further theoretical details will be omitted. But in regard to this dissertation it is important that we expect old search directions and gradients more and more unreliable by each transport along the manifold. Especially with line search by the Armijo condition it is possible that a line search will intersect the path of a line search of a previous iteration, as indicated in Figure 4.2.

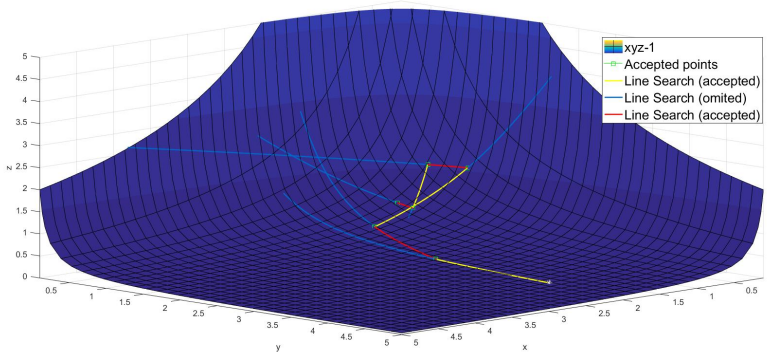


Figure 4.2: Path intersection on the surface $x_1 x_2 x_3 = 1$ by Armijo condition.

A LBFGS method on manifolds generated with Algorithm 5.2.5 and 5.4.3 was used to generate the plot.

4.6 Implementation of manifolds in computer programs

By now we have not mentioned anything about the local structure of the manifold itself. The structure of course depends on the topology of the manifold itself. Taking the sphere $S^2 \subset \mathbb{R}^3$ with the euclidean metric the geodesics are the great circles along the surfaces. Shifting the sphere in $\mathbb{R}^3$

to centre around another location, for example

$$\mathcal{M} = \{x \in \mathbb{R}^3 \mid (x_1 - 2)^2 + (x_2)^2 + (x_3)^2 - 1 = 0\}, \tag{4.12}$$

does not change the geodesics on. They remain remain the great circles along the surface.
Let $\mathcal{P} := (\{x \in \mathbb{R}^3 \mid x_3 > 0\}, d)$ equipped with the metric

$$d : \mathbb{R}^3 \to \mathbb{R};\ (x, y) \mapsto 2 \arcsin\left(\frac{||x - y||_{\text{eucl.}}}{2\sqrt{x_3 y_3}}\right).$$

$\mathcal{P}$ is then called the *Poincaré half space*. If we now consider the manifold $\mathcal{M}$ as a submanifold of $\mathcal{P}$ we get completely different geodesics along the surface, as geodesics highly depend on the inducing metric. We mention this to emphasize that we are a priori not interested in $\mathcal{M}$ as a manifold but as a constrain to our optimization problem. Hence we are only interested in the manifold pointwise and need the geodesic structure on $\mathcal{M}$ only to move along the surface in each iteration of the optimisation method. Thereby we will always consider manifolds to be subsets of the euclidean space $\mathbb{R}^n$. In this chapter we now compare different canonic ways to implement manifolds in computer programs.

4.6.1 Charts

At first we consider the most obvious way to implement a manifold into a computer - a finite atlas $\mathcal{A} = \{(U_i, \phi_i) || 1 \leq i \leq n\}$ of M. The open sets $U_1, \ldots, U_n$ are a covering of M, thereby one can rewrite problem 1.2 into

$$\min_{1 \leq i \leq n} \operatorname*{argmin}_{\phi(U_i) \subset \mathbb{R}^m} L(\phi_i^{-1}(x)).$$

Here one could either solve the partial optimisation problem $L(\phi_i^{-1}(x))$ separately on $\phi_i(U_i)$ and compare the solution for each $1 \leq i \leq n$. This can create unsolvable partial optimisation problems, for example considering a stereographic projection of S^1 onto $\mathbb{R}$. On the other hand one can use an optimisation method that switches between different charts, whenever it is required by the current iterate and search direction. In this case a deciding criteria to switch charts is necessary, for manifolds seldom consist of only one chart. This approach locally reduces the complexity of the problem to $\mathbb{R}^m$ by inverting the chart and avoids the use of differential equations, which we will

later see and use. However finding a sufficient atlas with charts (U_i, ϕ) and the respective inverse functions ϕ_i^{-1} is not simple. Also one has to consider that the function $L \circ \phi_i^{-1}$ will in general be more complex than the function L itself.

Example 4.6.1
Let $\mathcal{M} = S^1$, then $S^1 = \mathcal{V}(x^2 + y^2 - 1) = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 - 1 = 0\}$. By solving this implicit description for the variables x and y one can construct an an atlas $\mathcal{A}$ of $\mathcal{M}$ given by the following charts, respectively their inverses

$$\begin{aligned}
\phi_1^{-1} &: (-1, 1) \to S^1 \; x \mapsto \begin{pmatrix} x \\ \sqrt{1 - x^2} \end{pmatrix}; \\
\phi_2^{-1} &: (-1, 1) \to S^1 \; x \mapsto \begin{pmatrix} x \\ -\sqrt{1 - x^2} \end{pmatrix}; \\
\phi_3^{-1} &: (-1, 1) \to S^1 \; x \mapsto \begin{pmatrix} \sqrt{1 - x^2} \\ x \end{pmatrix}; \\
\phi_4^{-1} &: (-1, 1) \to S^1 \; x \mapsto \begin{pmatrix} -\sqrt{1 - x^2} \\ x \end{pmatrix}.
\end{aligned}$$

Those charts separate the circle into either the open upper and lower or left and right half circles. A simple criteria to switch between each chart could be the Euclidean distance to the points $(\pm 1, 0)$ and $(0, \pm 1)$. In this example then one would have to consider the objective function

$$\tilde{L}(z) = L(z, \sqrt{1 - z^2})$$

instead of $L(x, y)$ on (U, φ) and get the derivative

$$\frac{d}{dz}\tilde{L}(z) = \frac{d}{dx} L\left(z, \sqrt{1 - z^2}\right) + \frac{z}{\sqrt{1 - z^2}} \frac{d}{dy} L\left(z, \sqrt{1 - z^2}\right).$$

For more complex manifolds the charts and derivatives might result in a highly complex optimisation problem.

4.6.2 Grids

Another approach is to discretize the manifold, or to have a discretized approximation of the manifold by points scattered along the surface in a graph. Thus one has points $\mathcal{P} \subset \mathcal{M}$ and edges $\mathcal{E} \subset \mathcal{P}^2$ indicating neighbourhood of points. A pair $(p_1, p_2) \in \mathcal{P}^2$ is then an edge if we consider p_1 and p_2 to be neighbours or next to each other in an understanding of the manifold. This should be linked to the tangent space. For example one could say that

$(p_1, p_2) \in \mathcal{E}$ if there exists $v \in T_{p_1}\mathcal{M}$ such that the geodesic $\gamma : I \to \mathcal{M}$ with $\gamma(0) = p_1$ and $\dot{\gamma}(0) = v$ get sufficiently close to p_2 and does not previously get sufficiently close to another point $p_3 \in \mathcal{P}$.

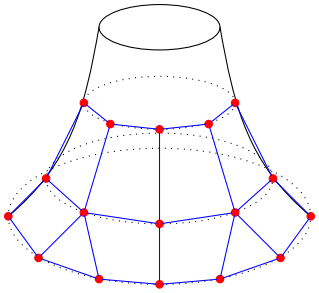


Figure 4.3: Illustration of a grid on the manifold induced by $(x^2 + y^2)z = 1$.

Such data set are not uncommon as they are generated by 3-dimensional scans of an object. Although for our purposes this kind of implementation is not useful as one requires discrete optimisation methods, which we do not want to consider.

4.6.3 Full implementation of the manifold

The most desirable case is having a manifold about which one has full knowledge. That is knowing a least distance projection, geodesics, means of parallel transport and maybe even more. An example where this is known is the Euclidean sphere S^2. One can easily decide if a point is on the manifold, and elsewise calculate the least distance projection of a point by $x \mapsto \frac{x}{||x||}$. Furthermore for a given starting point $p \in \mathcal{M}$ and a tangent vector $v \in T_pS^2 = \left\{x \in \mathbb{R}^3 \mid x^T p = 0\right\}$ one can easily calculate the geodesic.

To see that the geodesics on S^2 are the great circles let $p \in S^2$ and $v \in T_pS^2$, for simplicity we assume $||v|| = 1$. Then let $\gamma(t) = p\cos(t) + v\sin(t)$. Then γ has the derivates

$$\dot{\gamma}(t) = -p\sin(t) + v\cos(t)$$
$$\ddot{\gamma}(t) = -p\cos(t) - v\sin(t) = -\gamma(t).$$

By simple calculation one sees that for all $t \in \mathbb{R}$ holds

$$\begin{aligned}\gamma(t)^T\dot{\gamma}(t) &= (p^T\cos(t) + v^T\sin(t))(-p\sin(t) + v\cos(t)) \\ &= -\underbrace{p^Tp}_{=1}(\cos(t)\sin(t)) + \underbrace{p^Tv}_{=0}(\sin(t)^2 - \cos(t)^2) + \underbrace{v^Tv}_{=1}(\cos(t)\sin(t)), \\ &= 0\end{aligned}$$

furthermore

$$\ddot{\gamma}(t) = -\gamma(t) \in \langle\gamma(t)\rangle = T_{\gamma(t)}S^2.$$

Therefore $\gamma(t)$ is a geodesic by Theorem 4.5.8. For arbitrary $v \in T_p\mathcal{M}$ this has to be rescaled to

$$\gamma_{p,v}(t) = p\cos(||v||t) + \frac{v}{||v||}\sin(||v||t).$$

By the uniqueness of solutions of initial value problems (Picard-Lindelöf Theorem, [For08b, p. 148]) this shows that every geodesic on S^2 is given by such a great circle.

Geometrically this can be understood by smoothly rotating a know geodesic around S^2 by rotation matrices indicated in the figure below.

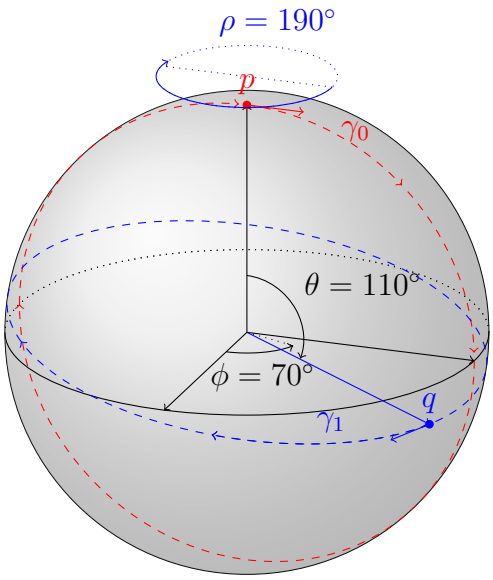


Figure 4.4: Transition of the oriented great circle in the x_1-x_3-plane to another great circle.

Figure 4.4 also indicates that for $p \neq -q$ the logarithmic problem $\log(p, q)$ is solved by a multiple of $\text{proj}_{T_pS^2}(q - p)$. To see this let $v = \frac{q-p^Tq\cdot p}{||q-p^Tq\cdot p||} \in T_p\mathcal{M}$ and consider

$$q = \gamma_{p,\alpha v}(1) = p\cos(\alpha) + \frac{q - p^Tq \cdot p}{||q - p^Tq \cdot p||}\sin(\alpha).$$

Solving this equation for $\alpha \in \mathbb{R}$ then gives $\alpha = \arccos(p^Tq)$, therefore

$$\log(p, q) = \begin{cases} 0 & \text{for } q = p; \\ \arccos(p^Tq)\frac{q-p^Tq\cdot p}{||q-p^Tq\cdot p||} & \text{for } q \neq \pm p; \\ \text{undefined} & \text{for } q = -p. \end{cases}$$

For the parallel transport assume $w \in T_p\mathcal{M}$ to be transported along the geodesic $\gamma_{p,v}$. We also without loss of generality assume here $||v|| = 1$. As $\dim(T_pS^2) = 2$ there exists up to sign a unique perpendicular vector $v^\perp$ such that $\left|\left|v^\perp\right|\right| = 1$ and $v^Tv^\perp = 0$. Then, if $w \notin \langle v\rangle$, one can calculate the orthogonal decomposition of $w \in T_pS^2$ by

$$w = w^Tv \cdot v + w^Tv^\perp \cdot v^\perp.$$

By parallel transport along v the perpendicular component of w remains unchanged while the parallel component will be transported to $w^Tv \cdot \dot\gamma(t)$. Therefore the parallel transport of w to $\gamma_{p,v}(t)$ is given by

$$\Pi^w_{\gamma_{p,w}}(t) = w^Tv \cdot \dot\gamma(t) + w^Tv^\perp \cdot v^\perp.$$

It is obvious that such a representation of a manifold is useful, but also not to be expected to be easily accessible. Therefore we will consider a structure, which allows us to at least approximate all necessary objects.

4.7 Algebraic Manifolds and their implementation

In this chapter we will consider manifolds which at the same time are affine manifolds. This will allow us to construct methods to locally calculate geodesics and solve the logarithmic problem. Thereby we obtain a way to locally approximate the structure of the manifold.

Convention 4.7.1
For a polynomial $f \in \mathbb{R}[x_1, \ldots, x_n]$ we do not distinguish between the polynomial f and the naturally induced polynomial function $f : \mathbb{R}^n \to \mathbb{R}$.

Definition 4.7.2
Let $f_1, \ldots, f_n \in \mathbb{R}[t_1, \ldots, t_n]$. The zero fibre of $f_1, \ldots, f_s$

$$\mathcal{V} = \mathcal{V}(f_1, \ldots, f_n) := \{x \in \mathbb{R}^n \mid f_i(x) = 0 \; \forall \; 1 \leq i \leq n\}$$

is called an *(real) affine variety*.

Definition 4.7.3
Let $\mathcal{M}$ be a manifold, then we call $\mathcal{M}$ an *algebraic manifold* if $\mathcal{M}$ as a set is also an affine variety.

Even thought further and more detailed discussion about algebraic manifolds can be found in [Sha13] and [Per+08], for our application the following theorem will be sufficient.

Theorem 4.7.4
Let $f_1, \ldots, f_m \in \mathbb{R}[x_1, \ldots, x_n]$ and $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subset \mathbb{R}^n$.
If $\nabla f_1(p), \ldots, \nabla f_m(p)$ are linearly independent for all $p \in \mathcal{M}$, then $\mathcal{M}$ is an algebraic submanifold of the Euclidean space $\mathbb{R}^n$. Furthermore

$$N_p\mathcal{M} = \langle \nabla f_1(p), \ldots, \nabla f_m(p) \rangle.$$

Proof. By Theorem 4.2.2 $\mathcal{M}$ is a $n-m$ dimensional manifold. Thus it remains to show that $N_p\mathcal{M}$ is spanned by the gradients $\nabla f_1(p), \ldots, \nabla f_m(p)$.
To see that $N_p\mathcal{M} = \langle \nabla f_1(p), \ldots, \nabla f_m(p) \rangle$, first let $1 \leq i \leq m$ and $\gamma : I \to \mathbb{R}^n$ be a curve in $\mathcal{M}$ with $\gamma(0) = p$ and $\dot{\gamma}(0) = v \in T_p\mathcal{M}$. Then $f_i(\gamma(t)) \equiv 0$ and therefore

$$0 = \frac{d}{dt}\bigg|_{t=0} f_i(\gamma(t)) = \nabla f_i(\gamma(0))^T \dot{\gamma}(0) = \nabla f_i(p)^T v,$$

which implies v is perpendicular to $\nabla f_i(p)$ for all $1 \leq i \leq m$. We will write $v \in \langle \nabla f_i(p) \rangle^\perp$. Thus

$$T_p\mathcal{M} \subseteq \bigcap_{i=1}^{m} \langle \nabla f_i(p) \rangle^\perp = \langle \nabla f_1(p), \ldots, \nabla f_m(p) \rangle^\perp.$$

As orthogonality is symmetric, this further implies

$$\langle \nabla f_1(p), \ldots, \nabla f_m(p) \rangle \subseteq T_p\mathcal{M}^\perp = N_p\mathcal{M}. \tag{4.13}$$

As $\nabla f_1(p), \ldots, \nabla f_m(p)$ are linearly independent by assumption this forms a m dimensional subspace of $N_p\mathcal{M}$. On the other hand

$$\dim(N_p\mathcal{M}) = n - \dim(T_p\mathcal{M}) = n - (n-m) = m.$$

Thereby the inclusion in (4.13) actually is an equality. □

A more general case is covered by the *Affine Jacobi criterion*.

Theorem 4.7.5 (Affine Jacobi criterion)
Let $X \subset \mathbb{R}^n$ be an affine variety with ideal

$$\langle f_1, \ldots, f_m \rangle = I(X) = \{p \in \mathbb{R}[x_1, \ldots, x_n] \mid p(x) = 0 \; \forall x \in X\}$$

and $a \in X$. Then X is smooth at x if and only if the rank of the $m \times n$ Jacobi matrix

$$\left(\left. \frac{\partial f_i}{\partial x_j} \right|_x \right)$$

is at least $n - \operatorname{codim}_X(x)$.

A full proof of this can be found in [Per+08].
With $\nabla f_1(p), \ldots, \nabla f_s(p)$ spanning $N_p\mathcal{M}$ by Theorem 4.7.4 we can now develop an easily calculable projection on $T_p\mathcal{M}$ and $N_p\mathcal{M}$ respectively.

Theorem 4.7.6
Let $p \in \mathcal{M} = \mathcal{V}(f_1, \ldots, f_m)$ and $D = \big(\nabla f_1(p) | \ldots | \nabla f_m(p)\big) \in \mathbb{R}^{n \times m}$.
Let further $D = QR$ be the QR-decomposition of D and $Q = \big(q_1 \mid \cdots \mid q_n\big)$.
Then for $P = \big(q_1 \mid \cdots \mid q_m\big)$ holds

$$\operatorname{proj}_{N_p\mathcal{M}} = PP^T, \tag{4.14}$$

$$\operatorname{proj}_{T_p\mathcal{M}} = I - PP^T. \tag{4.15}$$

Proof. Instead of the QR-decomposition of D we will consider the equivalent QR-decomposition of $\tilde{D} = \big(D \quad 0_{R^{n \times n-m}}\big)$. Then we have the following properties:

(i) $\tilde{D} = QR$;

(ii) $Q = \big(q_1 \mid \cdots \mid q_n\big)$ is an orthonormal matrix;

(iii) R is an upper triangle matrix.

Then (iii) implies for each $1 \leq i \leq n$

$$Re_i = \sum_{k=1}^{i} R_{i,k} e_k \in \langle e_1, \ldots, e_i \rangle. \tag{4.16}$$

Property (i) together with equation (4.16) then gives

$$\nabla f_i(p) = \tilde{D}e_i = QRe_i = \sum_{k=1}^{i} R_{i,k}Qe_k = \sum_{k=1}^{i} R_{i,k}q_k. \tag{4.17}$$

Equation (4.17) now implies

$$N_p\mathcal{M} = \langle \nabla f_1(p), \ldots, \nabla f_m(p) \rangle = \langle q_1, \ldots, q_m \rangle. \tag{4.18}$$

So $q_1, \ldots, q_m$ form a basis of $N_p\mathcal{M}$ and property (ii) makes it an orthonormal basis. Now let $v \in \mathbb{R}^n$ then

$$\begin{aligned} PP^T v &= (q_1 | \ldots | q_m) \begin{pmatrix} q_1^T \\ \vdots \\ q_m^T \end{pmatrix} v \\ &= (q_1 | \ldots | q_m) \begin{pmatrix} q_1^T v \\ \vdots \\ q_m^T v \end{pmatrix} \\ &= q_1^T v \cdot q_1 + \cdots + q_m^T v \cdot q_m \\ &= \mathrm{proj}_{\langle q_1, \ldots, q_m \rangle}(v). \end{aligned} \tag{4.19}$$

Equations (4.18) and (4.19) then imply $PP^T = \mathrm{proj}_{N_p\mathcal{M}}$.
The second claim follows as $N_p\mathcal{M} \perp T_p\mathcal{M}$. □

The respective algorithms are implemented as follows.

Algorithm 4.7.7: Proj$N_p\mathcal{M}$

Data: $p \in \mathcal{M} \subset \mathbb{R}^n, v \in \mathbb{R}^n$.

Result: Projection $v = \mathrm{proj}_{N_pM}(v) \in N_p\mathcal{M}$ to the tangent space $T_p\mathcal{M}$.

Global: Algebraic manifold $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subseteq \mathbb{R}^n$, the respective gradients $\nabla f_1, \ldots, \nabla f_m$.

Function: Proj$N_p\mathcal{M}(p, v)$:

---

1 $D \leftarrow (\nabla f_1(p) \mid \cdots \mid \nabla f_m(p) \mid 0_{R^{n \times n-m}})$

2 $[Q, R] \leftarrow \mathrm{QR}(D)$ `/* QR-decomposition of` D `*/`

3 $Q \leftarrow (Q_{i,j})_{\substack{1 \le i \le n \\ 1 \le i \le m}}$

4 $v \leftarrow QQ^T v$

5 **return** v

Algorithm 4.7.8: Proj$T_p\mathcal{M}$

Data: $p \in \mathcal{M}, v \in \mathbb{R}^n$
Result: $v = \text{proj}_{T_pM}(v) \in T_p\mathcal{M}$.
Global: $\mathcal{M} = \mathcal{V}(f_1, \dots, f_m) \subseteq \mathbb{R}^n$
Function: Proj$T_p\mathcal{M}(p, v)$:

1 $v \leftarrow v - \text{Proj}N_p\mathcal{M}(p, v)$
2 **return** v

Another result that is implied in proof of Theorem 4.7.6 is the following corollary.

Corallary 4.7.9
Let $p \in \mathcal{M} = \mathcal{V}(f_1, \dots, f_m)$ and $D = \big(\nabla f_1(p) | \dots | \nabla f_m(p)\big) \in \mathbb{R}^{n\times m}$.
Let further $D = QR$ be the QR-decomposition of D, $Q = \big(q_1 \mid \cdots \mid q_n\big)$ and $R = \begin{pmatrix} \overline{R} \\ 0_{\mathbb{R}^{n-m\times m}} \end{pmatrix}$. Then for all $1 \leq i \leq m$ holds

$$q_i = \sum_{j=1}^{i} R_{i,j}^{-T} \nabla f_j(p).$$

Furthermore, as calculating the QR-decomposition of a matrix D and matrix inversion is smooth in all components of D or R respectively we can also consider

$$q_i(p) = \sum_{j=1}^{i} \lambda_{i,j}(p) \nabla f_j(p)$$

with smooth functions $\lambda_{i,j}$ and q_i.

Proof. R is an upper triangle matrix and thereby we can decompose R by

$$R = \begin{pmatrix} \overline{R} \\ 0_{\mathbb{R}^{n-m\times m}} \end{pmatrix}.$$

Then $\overline{R} \in GL(m, \mathbb{R})$ and $D = QR$ implies

$$\overline{Q} := \big(q_1 \mid \cdots \mid q_m\big) = D\overline{R}^{-1}.$$

This is now equivalent to

$$q_i = \overline{Q} e_i = D\overline{R}^{-1} e_i \tag{4.20}$$

for all $1 \leq i \leq m$.

Evaluating equation (4.20) gives

$$\begin{aligned} q_i &= D\overline{R}^{-1} e_i \\ &= D\left(\overline{R}_{j,i}^{-1}\right)_{1\leq j\leq m} \\ &= \sum_{j=1}^{m} \nabla f_j(p) R_{j,i}^{-1} \end{aligned}$$

So for $q_i = \sum_{j=1}^{m} \lambda_{i,j} \nabla f_j(p)$ holds $\lambda_{i,j} = R_{j,i}^{-1} = R_{i,j}^{-T}$. □

Theorem 4.7.10 (Derivative of QR-Decomposition)
Let $A(t) \in \mathbb{R}^{n\times m}$ a smooth curve of matrices of rank m with QR-decomposition $A(t) = Q(t)R(t)$. Then

$$dQ = Q\Omega + Q_\perp K \tag{4.21}$$

and Ω and K are uniquely determined by

$$\Omega = Q^T(dA)R - (dR)R^{-1} \tag{4.22}$$
$$K = Q_\perp (dA)R^{-1}, \tag{4.23}$$

with $Q_\perp$ being an orthonormal completion of Q and Ω skew-symmetric.

The proof of this theorem is given by [Tow18]. We will go through the proof as it is constructive and will give the structure of Algorithm 4.7.11.

Proof. By Corollary (4.7.9) there exist smooth curves $Q(t)$ and $R(t)$, such that $A(t) = Q(t)R(t)$ form a QR-decomposition. Also by the properties of the QR-decomposition one has that the rows of Q are orthonormal and thus $Q^TQ = I$. Differentiating this gives

$$dQ^TQ + Q^TdQ = 0. \tag{4.24}$$

Therefore $\Omega := Q^TdQ$ is skew-symmetric. Now choose $Q_\perp$ such that $\begin{pmatrix} Q & Q_\perp \end{pmatrix}$ forms an orthonormal $n \times n$ matrix and then

$$dQ = QQ^T \cdot dQ + Q_\perp Q_\perp^T \cdot dQ \tag{4.25}$$

as QQ^T and $Q_\perp Q_\perp^T$ are projections to complement subspaces of $\mathbb{R}^n$. Then by defining $K := Q_\perp^T dQ$ one has

$$dQ = Q\Omega + Q_\perp K. \tag{4.26}$$

Differentiating the QR decomposition on the other hand gives

$$dA = (dQ)R + Q(dR). \tag{4.27}$$

Substituting dQ in (4.27) by (4.26) results in

$$dA = (Q\Omega + Q_\perp K)R + Q(dR) \tag{4.28}$$

By multiplying (4.28) with Q^T from the left and R^{-1} from the right one gets

$$Q^T(dA)R^{-1} = d\Omega + \underbrace{Q^T Q_\perp}_{=0}(K) + (dR)R^{-1} = \Omega + (dR)R^{-1}. \tag{4.29}$$

As R is always an upper triangle matrix so is R^{-1} and dR. Together with Ω being skew-symmetric this gives a unique decomposition by

$$(dR)R^{-1} = \text{Upper}\left(Q^T(dA)R^{-1} + (Q^T(dA)R^{-1})^T\right) + \text{diag}(Q^T(dA)R^{-1}). \tag{4.30}$$

Here Upper maps a matrix to its strictly upper part. Now multiplying (4.28) with $Q_\perp^T$ from the left and R^{-1} from the right one gets

$$Q_\perp^T(dA)R^{-1} = \underbrace{Q_\perp^T Q}_{=0} d\Omega + dK + \underbrace{Q_\perp^T Q}_{=0}(dR)R^{-1} = K. \tag{4.31}$$

□

The same way one can approach a derivative of the QR decomposition of $A(\gamma) = \left(\nabla f_1(\gamma) \mid \cdots \mid \nabla f_m(\gamma)\right)$ with respect to γ_i for $1 \leq i \leq n$. Note here that

$$\begin{aligned} \frac{d}{d\gamma_i} A(\gamma) &= \begin{pmatrix} \frac{d}{d\gamma_i}\nabla f_1(\gamma) & \dots & \frac{d}{d\gamma_i}\nabla f_m(\gamma) \end{pmatrix} \\ &= \begin{pmatrix} H_{f_1} i,* & \dots & H_{f_m} i,* \end{pmatrix} \\ &= \begin{pmatrix} H_{f_1} e_i & \dots & H_{f_m} e_i \end{pmatrix}. \end{aligned} \tag{4.32}$$

Equation (4.32) can now be used to construct the necessary algorithm to calculate the differentials dQ and dR of the QR decomposition along the geodesics γ.

Algorithm 4.7.11: DiffQR

Data: $p = \gamma(0) \in \mathcal{M}$.

Result: Differentials $dQ = \frac{dQ}{d\gamma_i}(0)$ of the orthonormal matrix Q and $dR = \frac{dR}{d\gamma_i}(0)$ of the upper triangle matrix R of the QR-decomposition of $(\nabla f_1(\gamma(0)), \ldots, \nabla f_m(\gamma(0)))$.

Global: $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subseteq \mathbb{R}^n$, Gradients $\nabla f_1, \ldots, \nabla f_m$, Hessians $H_{f_1}, \ldots, H_{f_m}$.

Function: DiffQR(p, i):

$A \leftarrow (\nabla f_1(p), \ldots, \nabla f_m(p))$

$dA \leftarrow (H_{f_1}(p)e_i, \ldots, H_{f_m}(p)e_i)$

$[\overline{Q}, \overline{R}] \leftarrow \mathrm{QR}(A)$

$R \leftarrow \overline{R}_{1 \le i,j \le m}$

$Q \leftarrow \overline{Q}_{\substack{1 \le i \le n \\ 1 \le j \le m}}$

$Q_\perp \leftarrow \overline{Q}_{\substack{1 \le i \le n \\ m+1 \le j \le n}}$

$dK \leftarrow Q_\perp^T (dA) R^{-1}$

$A \leftarrow Q^T (dA) R^{-1}$

$dRR \leftarrow \mathrm{Upper}(A + A^T) + diag(A)$

$d\Omega \leftarrow A - dRR$

$dQ \leftarrow Q d\Omega + Q_\perp dK$

$dR \leftarrow dRR \cdot R$

return dQ, dR

Theorem 4.7.4 states, that $\mathcal{V}(f_1, \ldots, f_s)$ is a manifold if $\nabla f_1(p), \ldots, \nabla f_s(p)$ are linearly independent for all $p \in \mathcal{V}(f_1, \ldots, f_s)$. This can actually be further refined.

Theorem 4.7.12
Let $\mathcal{V} = \mathcal{V}(f_1, \ldots, f_s)$ and $\mathcal{M} \subset \mathcal{V}$ be a connected component of $\mathcal{V}$ such that for all $p \in \mathcal{M}$ holds $\nabla f_1, \ldots, \nabla f_s$ are linearly independent. Then $\mathcal{M}$ is a manifold.

The proof is the same as the proof of Theorem 4.7.4, as one can still construct an atlas of $\mathcal{M}$ by applying the implicit function theorem to all $p \in \mathcal{M}$. One can consider the following example.

Example 4.7.13
Let $f_1 = x_2^2 - x_1(x_1^2 - 4), f_2 = x_1^2 - x_3^2 \in \mathbb{R}[x_1, x_2, x_3]$ with gradients

$$\nabla f_1 = \begin{pmatrix} 4 - 3x_1^2 \\ 2x_2 \\ 0 \end{pmatrix} \text{ and } \nabla f_2 = \begin{pmatrix} 2x_1 \\ 0 \\ -2x_3 \end{pmatrix}.$$

Then f_1 defines a cubic curve in $\mathbb{R}^2 \times \{x_3\}$ with two connected components as shown in Figure 4.5. On the other hand f_2 implies $x_1^2 = x_3^2$, which as a variety is itself a non-trivial union of the varieties

$$X = \mathcal{V}(x_1 - x_3) \cup \mathcal{V}(x_1 + x_3).$$

Then X is not smooth in $(0, x_2, 0)$. This is also represented by the gradient $\nabla f_2(0, x_2, 0) = 0$. Obviously $(0, 0, 0)$ is now a point of $\mathcal{V}(f_1, f_2) \subset \mathbb{R}^3$ and thus $\mathcal{V}(f_1, f_2)$ does not form a manifold. If one now restricts oneself to the connected component of the cubic which is bound away from the origin the same problem will not occur, as for x_1 bound away from 0 so is x_3 and thus ∇f_1 and ∇f_2 are linearly independent.

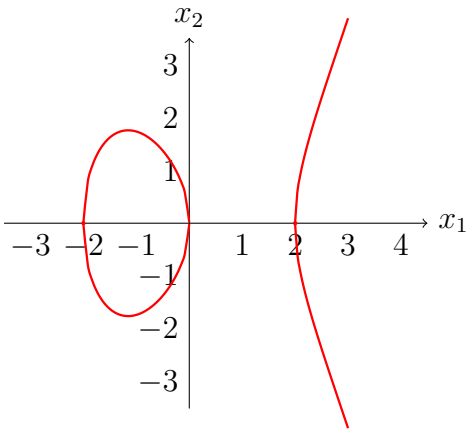


Figure 4.5: Cubic Curve of $f_1 = x_2^2 - x_1(x_1^2 - 4)$.

So either one has to check that the gradients of the generators $f_1, \ldots, f_s$ are never linearly dependent or has to find a connected component of $\mathcal{V}(f_1, \ldots, f_s)$ which satisfies this requirement. This means vice versa if $p \in \mathcal{V}(f_1, \ldots, f_s)$ is a point such that $\nabla f_1(p), \ldots, \nabla f_s(p)$ are linearly dependent, then there exist non-trivial $\lambda_1, \ldots, \lambda_s \in \mathbb{R}$, such that

$$0 = \sum_{i=1}^{s} \lambda_i \nabla f_i(p) = \begin{pmatrix} \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i(p) \\ \vdots \\ \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i(p) \end{pmatrix} \tag{4.33}$$

This can be read as $(p, \lambda) = (p, \lambda_1, \ldots, \lambda_s)$ is a zero of $\sum_{i=1}^{s} \lambda_i \partial_{x_j} f_i(p)$ for all $1 \leq j \leq n$. Therefore

$$(x, \lambda) \in \mathcal{V}\left(f_1, \ldots, f_s, \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i(p), \ldots, \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i(p)\right) \tag{4.34}$$

Obviously

$$\begin{aligned} &\mathcal{V}(f_1, \ldots, f_s) \times \{0_{\mathbb{R}^s}\} \\ &\qquad \subset \mathcal{V}\left(f_1, \ldots, f_s, \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i(p), \ldots, \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i(p)\right) \\ &\qquad \subset \mathcal{V}(f_1, \ldots, f_s) \times \mathbb{R}^s, \end{aligned} \tag{4.35}$$

The implicit function theorem is then not applicable on all of $\mathcal{V}(f_1, \ldots, f_s)$ if

$$\mathcal{V}\left(f_1, \ldots, f_s, \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i(p), \ldots, \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i(p)\right) \backslash \mathcal{V}(f_1, \ldots, f_s) \times \{0_{\mathbb{R}^s}\} \neq \emptyset.$$

This itself is not simple to solve although it may give an indication. Coming back to (4.33) and assuming $\lambda_i \neq 0$ for some $1 \leq i \leq s$, we can then multiply (4.33) by a scalar such that $\lambda_i = 1$. Thus instead of (4.35) it is sufficient to consider

$$\mathcal{V}\left(f_1, \ldots, f_s, \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i(p), \ldots, \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i(p), \lambda_i - 1\right) = \emptyset \tag{4.36}$$

for each $1 \leq i \leq s$.

Example 4.7.14
Continuing example 4.7.13 we get the varieties

$$\begin{aligned} &\mathcal{V}(y^2 - x^3 + 4x, x^2 - z^2, \lambda_1(-3x^2 + 4) + 2\lambda_1 x, 2\lambda_1 y, 2\lambda_2 z, \lambda_1 - 1) \\ &\mathcal{V}(y^2 - x^3 + 4x, x^2 - z^2, \lambda_1(-3x^2 + 4) + 2\lambda_1 x, 2\lambda_1 y, 2\lambda_2 z, \lambda_2 - 1) \end{aligned} \tag{4.37}$$

The varieties now do not depend on the generators themself, but on the ideal generated by the same generators, so for $f_1 \ldots, f_s \in \mathbb{R}[x_1, \ldots, x_n]$ one has

$$\mathcal{V} = \mathcal{V}(f_1, \ldots, f_s) = \mathcal{V}(\langle f_1, \ldots, f_s \rangle).$$

By choosing a suitable set of generators of $\mathcal{V}$ one can therefore simplify the process of determining the elements of $\mathcal{V}$. The object of choice is then a Groebner base of $\langle f_1, \ldots, f_s \rangle$.

4.8 Groebner Bases and the Applicability of the Inverse Function Theorem

Groebner bases are a strong tool to understand ideals of polynomials and their varieties in multiple dimensions and above arbitrary fields. In regard to the topic of this dissertation only $\mathbb{R}$ or $\mathbb{C}$ are required, thought in this chapter $\mathbb{K}$ will always denote an arbitrary field. An introduction and proofs to the following statements in all of its generality can be found in [Sha13][Chapter 2 & 3]. A Groebner basis is strongly connected to the higher dimensional polynomial division algorithm and thus dependent on a monomial ordering. We can use Groebner basis to approach a criteria if the implicit function theorem 4.7.4 and therefore the Algorithms presented in this dissertation are applicable.

Definition 4.8.1
A monomial order $>$ is a well-ordered linear order on $(\mathbb{N}^n, +)$. By the monoid isomorphism

$$(a_1, \dots, a_n) \mapsto x_1^{a_1} \dots x_n^{a_n},$$

one gets an ordering on the set of monomial in $x_1, \dots, x_n$ induced by the monomial order on $\mathbb{N}^n$.

By choosing a monomial order one now is able to determine a leading term of any polynomial in regard to that monomial order and apply a polynomial division algorithm. Still the algorithm is not useable to solve a membership problem, that is for $f_1, \dots, f_m \in \mathbb{K}[x_1, \dots, x_n]$ is $f \in \langle f_1, \dots, f_s \rangle$. A Groebner basis is a specific kind of generators of polynomial ideals solving this problem.

Definition 4.8.2
Let $I \trianglelefteq \mathbb{K}[x_1, \dots, x_n]$ and $>$ be a monomial ordering, then a Groebner basis $G := \{g_1, \dots, g_t\}$ is a finite set of generators of I such that

$$\langle LT(g_1), \dots, LT(g_t) \rangle = \langle LT(I) \rangle$$

Thus a set of generators is a Groebner basis of I if and only if the leading term of each element of I is dividable by one of the leading terms $LT(g_i)$.
By Dickson's Lemma for each ideal I there exists a - non-unique - Groebner basis.

Theorem 4.8.3 (Dickson's Lemma)
Let $A \subset \mathbb{N}^n$ and consider the monomial ideal $I = \langle x^\alpha | \alpha \in A \rangle \subset \mathbb{K}[x_1, \ldots, x_n]$. Then I can be written down in the form

$$I = \langle x^{a_1}, \ldots, x^{a_s} \rangle$$

with $a_1, \ldots, a_s \in A$. In particular, I has a finite basis.

For a proof see [Sha13, p. 71]
Groebner bases now allow one to apply the polynomial division algorithm without regard of order and obtain a unique remainder.

Proposition 4.8.4
Let $G := \{g_1, \ldots, g_s\}$ be a Groebner basis for an ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$ and $f \in \mathbb{K}[x_1, \ldots, x_n]$. Then there is a unique $r \in \mathbb{K}[x_1, \ldots, x_n]$ with the following properties

i) No term of r is divisible by any of $LT(g_1), \ldots, LT(g_s)$.

ii) There exists $g \in I$ such that $f = g + r$.

In particular, r is the remainder on division of f by $g_1, \ldots, g_s$ disregarding the order of $g_1, \ldots, g_s$.

For a proof see [Sha13, p. 82]
Theorem 4.8.4 then implies that the division algorithm paired with Groebner bases solves the ideal membership problem.

Corallary 4.8.5
Let $f \in \mathbb{K}[x_1, \ldots, x_n]$ and $G = \{g_1, \ldots, g_t\}$ a Groebner basis of some ideal $I \trianglelefteq \mathbb{K}[x_1, \ldots, x_n]$. Then $f \in I$ if and only if the remainder of f on division by $g_1, \ldots, g_t$ is zero.

For a proof see [Sha13, p. 82]
Groebner bases then also allow determining solutions to polynomial equations, so determining the elements of the respective variety, by eliminating variables.

Theorem 4.8.6 (The Elimination Theorem)
Let $I \subset \mathbb{K}[x_1, \ldots, x_n]$ be an ideal and let G be a Groebner basis of I with respect to lex order where $x_1 > x_2 > \cdots > x_n$. Then for each $0 \leq l \leq n$ the set

$$G_l := G \cap \mathbb{K}[x_{l+1,\ldots,n}]$$

is a Groebner basis of the l-th elimination ideal $I_l := I \cap K[x_{l+1}, \ldots, x_n]$.

For a proof see [Sha13, p. 116]

The Elimination Theorem now allows us to consider polynomials with fewer variables and try to solves these for zeros. Then one can use these zeros to simplify the remaining polynomials. Before we demonstrate this in Example 4.8.7, there is another important property of Groebner bases.

Let $I = \langle b_1, \ldots, b_t \rangle \subset \mathbb{K}[x_1, \ldots, x_n]$, then there exists an algorithm which calculates a Groebner basis of I in finite time. This Algorithm in its most simple form is called Buchberger-Algorithm [Sha13, §2.7].

Example 4.8.7

We continue with example 4.7.13. We were interested in the varieties

$$\mathcal{V}_1 = \mathcal{V}(x_2^2 - x_1^3 + 4x_1, x_1^2 - x_3^2, \lambda_1(-3x_1^2 + 4) + 2\lambda_1 x_1, 2\lambda_1 x_2, 2\lambda_2 x_3, \lambda_1 - 1)$$
$$\mathcal{V}_2 = \mathcal{V}(x_2^2 - x_1^3 + 4x_1, x_1^2 - x_3^2, \lambda_1(-3x_1^2 + 4) + 2\lambda_1 x_1, 2\lambda_1 x_2, 2\lambda_2 x_3, \lambda_2 - 1).$$

The related Ideals are

$$I_1 = \langle x_2^2 - x_1^3 + 4x_1, x_1^2 - x_3^2, \lambda_1(-3x_1^2 + 4) + 2\lambda_1 x_1, 2\lambda_1 x_2, 2\lambda_2 x_3, \lambda_1 - 1 \rangle \tag{4.38}$$
$$I_2 = \langle x_2^2 - x_1^3 + 4x_1, x_1^2 - x_3^2, \lambda_1(-3x_1^2 + 4) + 2\lambda_1 x_1, 2\lambda_1 x_2, 2\lambda_2 x_3, \lambda_2 - 1 \rangle. \tag{4.39}$$

By applying the Buchberger Algorithm to these set of generators with lex-order $\lambda_1 > \lambda_2 > x_1 > x_2 > x_3$ one gets the Groebner bases (calculated in GAP)

$$G_1 = \{1\} \tag{4.40}$$
$$G_2 = \{x_3, x_2^3, x_2^2 + 4x_1, \lambda_2 - 1, x_2^2 + 8\lambda_2\}. \tag{4.41}$$

(4.40) now translates to $I_1 = \mathbb{R}[x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_s]$ and thus $\mathcal{V}_1 = \emptyset$. This shows there exists no linear combination

$$0 = \lambda_1 \nabla f_1(x) + \lambda_2 \nabla f_2(x)$$

with $\lambda_1 \neq 0$. Considering (4.41) on the other hand one sees that for

$$(x_1, x_2, x_3, \lambda_1, \lambda_2) \in \mathcal{V}_2$$

one has $I_2 \cap \mathbb{R}[x_3] = \langle x_3 \rangle$ and therefore $x_3 = 0$. The next elimination ideal is $I_2 \cap \mathbb{R}[x_2, x_3] = \langle x_2^3, x_3^2 \rangle$, therefore $x_2 = x_3 = 0$. And so forth one gets

$$x_1 = x_2 = x_3 = \lambda_1 = 0 \text{ and } \lambda_2 = 1,$$

therefore for $x = (x_1, x_2, x_3)$ as above one has $x \in \mathcal{V}(f_1, f_2)$ and

$$0 = 0 \cdot \nabla f_1(x) + 1 \cdot \nabla f_2(x) = \nabla f_2(x),$$

by which there exists $x \in \mathcal{V}(f_1, f_2)$ such that $\nabla f_1(x), \nabla f_2(x)$ are not linearly independent.

A Groebner basis does not necessarily guarantee an easy way to check if the Implicit Function Theorem is applicable, but it gives an algorithmic approach to simplify the description of the variety involved. We summarize this in the following theorem:

Theorem 4.8.8
Let $f_1, \ldots, f_s \in \mathbb{R}[x_1, \ldots, x_n]$, then $\nabla f_1, \ldots, \nabla f_s$ are linearly independent for all $p \in \mathcal{V}(f_1, \ldots, f_s)$ if and only if

$$\mathcal{V}\left(f_1, \ldots, f_s, \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i, \ldots, \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i, \lambda_k - 1\right) = \emptyset$$

for all $1 \leq k \leq n$.

Proof. Let $p \in \mathcal{V}(f_1, \ldots, f_s)$ such that $\nabla f_1(p), \ldots, \nabla f_s(p)$ are linearly dependent, then there exists non-trivial $\lambda = (\lambda_1, \ldots, \lambda_s) \in \mathbb{R}^s$ such that $0 = \sum_{i=1}^{s} \lambda_i \nabla f_i(p)$. Without loss of generality we can assume $\lambda_1 \neq 0$ and by rescaling $\lambda_1 = 1$. Now (p, λ) satisfies

$$\begin{aligned} 0 &= f_1(p) = f_1(p, \lambda) = \cdots = f_s(p) = f_s(p, \lambda) \\ 0 &= \lambda_1 - 1 \\ 0 &= \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i(p) = \cdots = \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i(p). \end{aligned} \tag{4.42}$$

Therefore

$$\mathcal{V}\left(f_1, \ldots, f_s, \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i, \ldots, \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i, \lambda_1 - 1\right) \neq \emptyset. \tag{4.43}$$

On the other hand assume with out loss of generality

$$(p, \lambda) \in \mathcal{V}\left(f_1, \ldots, f_s, \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i, \ldots, \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i, \lambda_1 - 1\right) \tag{4.44}$$

then $p \in \mathcal{V}(f_1, \ldots, f_s)$ and

$$0 = \begin{pmatrix} \sum_{i=1}^{s} \lambda_i \partial_{x_1} f_i(p) \\ \vdots \\ \sum_{i=1}^{s} \lambda_i \partial_{x_n} f_i(p) \end{pmatrix} = \sum_{i=1}^{s} \lambda_i \nabla f_i(p). \tag{4.45}$$

By assumption $\lambda_1 \neq 0$ therefore $\nabla f_1(p), \ldots, \nabla f_s(p)$ are linearly dependent at least at $p \in \mathcal{V}(f_1, \ldots, f_s)$. □

4.9 (Matrix-)Lie Groups

Lie Groups combine the topological structure of a manifold with the algebraic structure of a group, thereby entwining the fields differential geometry and algebra. They are to be found in various mathematical fields and other scientific disciplines like physic. In the later they are used - among others - to describe symmetries and transformations. Therefore Lie groups seem to be a reasonable field to test the developed algorithms. To apply the ideas of section 4.7 we obviously need the structure of a variety, which can often be found in Matrix Lie groups.

Definition 4.9.1
Let $(G, \cdot)$ be a group such that the set G is also a smooth manifold. We then call $(G, \cdot)$ a Lie group if furthermore the multiplication map

$$\mu : G \times G \to G,\ (g, h) \mapsto g \cdot h,$$

and the inversion map

$$\iota : G \to G,\ g \mapsto g^{-1}$$

are smooth.

A trivial example of a Lie group is $(\mathbb{R}^n, +)$, another common non-trivial example is

$$GL(n, \mathbb{R}) = \{A \in \mathbb{R}^{n \times n} \mid \det(A) \neq 0\}$$

with the usual matrix multiplication. The determinant is a smooth function $\det : M(n, \mathbb{R}) \to \mathbb{R}$ and thereby preimages of open sets are open again. Then $GL(n, \mathbb{R}) = \det^{-1}(\mathbb{R} \backslash 0)$ is an open subset of $\mathbb{R}^{n \times n}$, thereby by Theorem 4.1.4 $GL(n, \mathbb{R})$ is a manifold. For two matrices $A, B \in GL(n, \mathbb{R})$ the product AB is element-wise a polynomial in the entries of A and B and thereby the multiplication of $GL(n, \mathbb{R})$ is also smooth. The smoothness of the inversion

map is less obvious and requires the inversion by the adjugate matrix. That is

$$\mathrm{Adj}(A) := \left((-1)^{i+j} \det(A_{j,i})\right),$$

where $A_{i,j}$ is the matrix one obtains by deleting row i and column j from A. Then the adjugate matrix satisfies $A \cdot \mathrm{Adj}(A) = \det(A) I_n$, which implies

$$A^{-1} = \frac{1}{\det(A)} \mathrm{Adj}(A).$$

Obtaining $A_{i,j} \in M(n-1, \mathbb{R})$ is a projection and therefore smooth. Together with the determinant being smooth this also means Adj is a smooth function and therefore again the inversion map is smooth.

Remark 4.9.2
Consider the map $c : \mathbb{R}\backslash\{0\} \to GL(n, \mathbb{R}),\ t \mapsto tE_n$. The extension of this map at $t = 0$ by $0 \mapsto 0_{M(n,\mathbb{R})}$ makes it a smooth curve $\tilde{c}$ in $M(n, \mathbb{R})$. Now assume $GL(n, \mathbb{R}) = \mathcal{V}(f_1, \ldots, f_s)$, then for all $1 \leq i \leq s$ holds $f_i(\tilde{c}(t)) = 0$. But $f_i \circ \tilde{c}$ is a polynomial in only one variable, therefore $f_i \circ \tilde{c} \equiv 0$ as a function. But then thereby $0_{M(n,\mathbb{R})} = \tilde{c}(0) \in \mathcal{V}(f_1, \ldots, f_s) = GL(n, \mathbb{R})$, which is a contradiction.
This makes $G = GL(n, \mathbb{R})$ itself not sufficient for our theories. However there are submanifolds of $GL(n, \mathbb{R})$ which subject to polynomial constraints. Often this is because the Lie group is given by a set of functions satisfying some matrix equation. For example $SL(n, \mathbb{R}) := \{M \in GL(n, \mathbb{R}) \mid \det(M) = 1\}$ or the orthogonal group $O(n) := \{M \in GL(n, \mathbb{R}) \mid A^T A = I_n\}$. Another matrix Lie group, closely related to $O(n)$ is chosen for testing and discussed in the following subsection.

4.9.1 The Stiefel Manifold

The Stiefel manifold is the set of all orthonormal p-frames in $\mathbb{R}^n$. This manifold plays an important role in various mathematical and practical applications, such as among others robotics and quantum mechanics. Therefore we will later use it as a test manifold for the developed algorithms. In this subsection we will present all the necessary formulas and equations for later application.

Definition 4.9.3
Let $p \leq n \in \mathbb{N}$, then the *Stiefel manifold* is the set

$$ST(p, n) := \left\{X \in \mathbb{R}^{n \times p} | X^T X = I_p\right\},$$

with $I_p \in \mathbb{R}^{p\times p}$ being again the $p \times p$ identity matrix. For $p = n$ the Stiefel manifold is just the orthonormal group $O(n)$.

To see that $ST(p, n)$ actually forms a manifold we can again just consider our theory of Section 4.7,

Theorem 4.9.4
For $p \leq n \in \mathbb{N}$ the Stiefel manifold $ST(p, n)$ is actually a manifold.

Proof. We will consider $f_{i,j}(X) = (X^T X)_{i,j}$, then

$$ST(p, n) = \mathcal{V}(f_{i,j} \mid 1 \leq i \leq j \leq p).$$

By Theorem 4.7.4 it is therefore sufficient to check, that the gradients $\nabla f_{i,j}(X)$ for $1 \leq i \leq j \leq p$ are linearly independent for all $X \in ST(p, n)$. First of all we need to determine the gradients.

$$\left.\frac{df_{i,j}}{dx_{s,t}}\right|_X = \begin{cases} 2x_{s,t} & \text{if } i = t = j \\ x_{s,i} & \text{if } i \neq t = j \\ x_{s,j} & \text{if } i = t \neq j \\ 0 & \text{else} \end{cases}. \tag{4.46}$$

If we consider $\nabla f_{i,j}$ to be a matrix again and $X = (x_1|\dots|x_n)$ then

$$\nabla f_{i,j}(X) = \begin{cases} (0|\dots|0|2x_i|0|\dots|0) & \text{if } i = j \\ (0|\dots|0|x_j|0\dots|0|x_i|0|\dots|0) & \text{if } i \neq j. \end{cases} \tag{4.47}$$

Now let $\lambda_{i,j} \in \mathbb{R}$ be given such that

$$0 = \sum_{i=1}^{n}\sum_{j=1}^{i} \lambda_{i,j}\nabla f_{i,j}(X). \tag{4.48}$$

We represent $\nabla f_{i,j}(X)$ as matrices again and use equation (4.47). First it is to notice that $\nabla f_{i,j}(X)$ only has entries in the first column if and only if $i = 1$. This gives

$$\begin{aligned} 0 &= \left(\sum_{i=1}^{n}\sum_{j=1}^{i} \lambda_{i,j}\nabla f_{i,j}(X)\right)_{*,1} \\ &= \sum_{j=1}^{p} \lambda_{1,j}\nabla f_{i,j}(X)_{*,1} \\ &= 2\lambda_{1,1}x_1 + \sum_{j=2}^{p} \lambda_{1,j}x_j \end{aligned} \tag{4.49}$$

Now if $X \in ST(p,n)$ the columns $x_1, \ldots, x_p$ are linearly independent. Then equation (4.49) implies $\lambda_{1,1} = \cdots = \lambda_{1,p} = 0$. Now one can repeat this iteratively for the second row until the p-th row and get $\lambda_{i,j} = 0$ for all $1 \leq i \leq j \leq p$, as desired. □

To apply the developed algorithms we require the gradients of $f_{i,j} = (X^T X)_{i,j}$, which are given in (4.47). Furthermore we need the Hessians and third order derivates. The calculation of the Hessian is more easily understood when considering the vector interpretation of X. To do so we use the bijection

$$\mathrm{Vec} : \mathbb{R}^{n\times m} \to \mathbb{R}^{nm},\ X \mapsto \begin{pmatrix} X_{1,1} & \ldots & X_{n,1} & \ldots & X_{1,n} & \ldots, X_{n,n} \end{pmatrix}^T \tag{4.50}$$

and its inverse

$$\mathrm{Mat} : \mathbb{R}^{nm} \to \mathbb{R}^{n\times m},\ x \mapsto \begin{pmatrix} x_1 & x_{n+1} & \ldots & x_{(m-1)n+1} \\ x_2 & x_{n+2} & \ldots & x_{(m-1)n+2} \\ \vdots & & & \vdots \\ x_n & x_{2n} & \ldots & x_{nm} \end{pmatrix}. \tag{4.51}$$

Then the Hessian of $f_{i,j}$ is a $nm \times nm$ matrix, which we will write as a matrix of its $n \times n$ submatrices $A_{i,j}$ by

$$H_{f_{i,j}}(X) = \begin{pmatrix} A_{1,1} & \ldots & A_{1_m} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \ldots & A_{m,m} \end{pmatrix}. \tag{4.52}$$

If $i = j$, then $A_{i,i} = 2I_n$ and $A_{k,l} = 0$ elsewise.
If $i \neq j$, then $A_{i,j} = A_{j,i} = I_n$ and $A_{k,l} = 0$ elsewise.
Matrix multiplication is a polynomial of total degree two. Therefore third order derivatives are always zero.

Chapter 5

Algorithms on Manifolds

Convention 5.0.1
In this chapter we will assume that the assumptions of Theorem 4.7.4 respectively Theorem 4.7.12 hold and thus the algorithms developed in Section 4.7 are applicable. Therefore $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subset \mathbb{R}^n$ with $\nabla f_1(p), \ldots, \nabla f_m(p)$ linearly independent for all $p \in \mathcal{M}$, also we will often consider f_i to be the components of the function $f : \mathbb{R}^n \to \mathbb{R}^m$.

5.1 Steepest Descent on Manifolds

For the objective function $L : \mathcal{M} \to \mathbb{R}$, a naive version of the steepest descent algorithm takes the form

Algorithm 5.1.1: Naive Steepest Descent

Data: Objective function $L : \mathbb{R}^n \to \mathbb{R}$,$\varepsilon > 0$, starting point $x_0 \in \mathcal{M}$.
Global: Manifold $\mathcal{M} \subset \mathbb{R}^n$.
Result: $x \in \mathcal{M}$ such that $\text{proj}_{T_x\mathcal{M}}(\nabla L(x)) \leq \varepsilon$.
Function: Steepest Descent(L, p_0):

$d \leftarrow \text{proj}_{T_x\mathcal{M}} \nabla L(x_0)$
while $||d|| > \varepsilon$ **do**
  $s \leftarrow \underset{0<t\in\mathbb{R}}{\arg\min}\, L(\gamma_{x,-d}(t))$
  $x \leftarrow \gamma_{x,-d}(s)$
  $d \leftarrow \text{proj}_{T_x\mathcal{M}} \nabla L(x)$
end
return x

The gradient of L will be calculated in $\mathbb{R}^n$ and then projected to the tangent space of the current iterate as stated in Theorem 4.4.2. The projection is then

calculated by Algorithm 4.7.8. Now given a starting point p and a descent direction $v \in T_p\mathcal{M}$ it remains to calculate the affiliated geodesic and then pursue the line search on $\mathcal{M}$.

5.2 Line Search on Manifolds

In the general case geodesics are hard to calculate and are locally solutions of a second order differential equation, which require the Christoffelsymbols.

Definition 5.2.1
Let (U, φ) be a chart of the m-manifold $\mathcal{M}$ and let

$$e_i = \frac{\partial}{\partial x_i} = [\varphi^{-1}(\varphi(p) + tx_i)] \in T_p\mathcal{M}$$

Then $e_1, \ldots, e_m$ form a local basis of $T_p\mathcal{M}$, by which we can implicitly define the Christoffelsymbols by

$$\frac{\partial e_i}{\partial x_j} = \sum_{k=1}^{m} \Gamma_{i,j}^k e_k. \tag{5.1}$$

Theorem 5.2.2
Let $\mathcal{M} \subset \mathbb{R}^n$ be an m-dimensional manifold and choose a coordinate chart $\varphi : U \to \Omega$ with inverse

$$\psi := \varphi^{-1} : \Omega \to U.$$

Let $\Gamma_{i,j}^k : \Omega \to \mathbb{R}$ be the Christoffelsymbols defined by equation (5.1) and let $c : I \to \Omega$ be a smooth curve. Then the curve

$$\gamma = \psi \circ c : I \to \mathcal{M}$$

is a geodesic if and only if c satisfies the 2nd order differential equation

$$0 = \ddot{c}_k + \sum_{i,j=1}^{m} \Gamma_{i,j}^k(c)\dot{c}_i\dot{c}_j$$

for each $k = 1, \ldots, m$.

A proof of this can be found in [RS22, p. 206].
In general we can not assume to have the Christoffelsymbols given and calculating them is also not simple. Furthermore for an a m dimensional manifold there are $\frac{m^2(m+1)}{2}$ symbols to be calculated at each point, assuming one has a

given formula. Therefore we will try a different approach making use of the implicit description of algebraic manifolds.

Theorem 5.2.3
Let $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subset \mathbb{R}^n$ an algebraic manifold and $\gamma : I \to \mathcal{M}$ a geodesic in $\mathcal{M}$. Let furthermore $t \in I$ and QR be the QR-decomposition of $D = \big(\nabla f_1(\gamma(t))| \cdots |\nabla f_m(\gamma(t))\big)$, then

$$\ddot{\gamma}(t) = (q_1| \cdots |q_m)\overline{R}^{-T} \begin{pmatrix} \dot{\gamma}(t)^T H_{f_1}(\gamma(t))\dot{\gamma}(t) \\ \vdots \\ \dot{\gamma}(t)^T H_{f_m}(\gamma(t))\dot{\gamma}(t) \end{pmatrix}, \tag{5.2}$$

with $\overline{R} = (R_{i,j})_{1 \leq i,j \leq m}$ and $Q = \big(q_1| \cdots |q_n\big)$.

Proof. Applying Theorem 4.5.8 (ii) to a geodesic $\gamma : [0,1] \subset I \subseteq \mathbb{R} \to \mathcal{M}$ gives for all $t \in I$

$$\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) = 0 \Leftrightarrow \ddot{\gamma}(t) \perp T_{\gamma(t)}\mathcal{M} \tag{5.3}$$

With our assumption $\mathbb{R}^n = T_p\mathcal{M} + N_p\mathcal{M}$ and thus $\ddot{\gamma}(t) \in N_p\mathcal{M}$, by which

$$\nabla_{\dot{\gamma}(t)}\dot{\gamma}(t) = 0 \Leftrightarrow \ddot{\gamma}(t) = \mathrm{proj}_{N_{\gamma(t)M}}\ddot{\gamma}(t) \tag{5.4}$$

Theorem 4.7.4 then gives $N_{\gamma(t)}M = \langle \nabla f_1(\gamma(t)), \ldots, \nabla f_m(\gamma(t)) \rangle_{\mathbb{R}}$ and let $(q_1, \ldots, q_k)$ be an orthonormal frame of $N_{\gamma(t)}M$. Therefore there exist $\lambda_{i,j} \in \mathbb{R}$ such that

$$q_i(t) = \sum_{i=1}^{m} \lambda_{i,j}(t)\nabla f_j(t). \tag{5.5}$$

Using this we get the following representation of $\ddot{\gamma}(t)$:

$$\begin{aligned} \ddot{\gamma}(t) &= \sum_{s=1}^{k} \langle \ddot{\gamma}(t), q_s \rangle \, q_s \\ &= \sum_{s=1}^{m} \sum_{i,j=1}^{m} \lambda_{s,i}\lambda_{s,j} \, \langle \ddot{\gamma}(t), \nabla f_i(t) \rangle \, \nabla f_j(t). \end{aligned} \tag{5.6}$$

By Corollary 4.7.9 we can choose q_i and $\lambda_{i,j}$ to be smooth in t. Now we want to consider that $f(\mathcal{M}) = \{0_{\mathbb{R}^m}\}$. As $\gamma(I) \subset \mathcal{M} = \mathcal{V}(f_1, \ldots, d_m)$ this implies $(f \circ \gamma)(I) = \{0_{\mathbb{R}^m}\}$. By derivation one obtains for all $1 \leq i \leq m$ and $t \in I$

$$0 = \delta_t(f_i \circ \gamma)(t) = \langle \dot{\gamma}(t), \nabla f_i(\gamma(t)) \rangle \tag{5.7}$$

and by the second order derivative

$$\begin{aligned} 0 &= \delta_t \langle \dot{\gamma}(t), \nabla f_i(\gamma(t)) \rangle \\ &= \langle \ddot{\gamma}(t), \nabla f_i(\gamma(t)) \rangle + \dot{\gamma}(t)^T H_{f_i}(\gamma(t)) \dot{\gamma}(t). \end{aligned} \tag{5.8}$$

This gives us, together with (5.6),

$$\ddot{\gamma}(t) = -\sum_{s=1}^{m} \sum_{i,j=1}^{m} \lambda_{s,i} \lambda_{s,j} \dot{\gamma}(t)^T H_{f_i}(\gamma(t)) \dot{\gamma}(t) \nabla f_j(\gamma(t)), \tag{5.9}$$

which we can approximate by some ODE-Solver. For the calculation we will - as stated in Corollary 4.7.9 - use the QR-decomposition to get the coefficients $\lambda_{i,j}$. As the columns of Q also form an orthonormal basis of $N_p\mathcal{M}$, we can simplify this to

$$\begin{aligned} \ddot{\gamma}(t) &= -\sum_{s=1}^{m} \sum_{i=1}^{m} \lambda_{s,i} \dot{\gamma}(t)^T H_{f_i}(\gamma(t)) \dot{\gamma}(t) q_s \\ &= -\sum_{s=1}^{m} q_s \left(\sum_{i=1}^{m} \lambda_{s,i} \dot{\gamma}(t)^T H_{f_i}(\gamma(t)) \dot{\gamma}(t) \right) \end{aligned} \tag{5.10}$$

Let $R = \begin{pmatrix} \overline{R} \\ 0_{\mathbb{R}^{(n-m)\times m}} \end{pmatrix}$, then by Corollary 4.7.9 the elements of $\overline{R}^{-T}$ define $\lambda_{s,i}$. Further considering each element of $\ddot{\gamma}$ separately we get

$$\sum_{i=1}^{m} \lambda_{s,i} \dot{\gamma}(t)^T H_{f_i}(\gamma(t)) \dot{\gamma}(t) = \left(\overline{R}^{-T} \begin{pmatrix} \dot{\gamma}(t)^T H_{f_1}(\gamma(t)) \dot{\gamma}(t) \\ \vdots \\ \dot{\gamma}(t)^T H_{f_m}(\gamma(t)) \dot{\gamma}(t) \end{pmatrix} \right)_s . \tag{5.11}$$

Thus we can omit inverting $\overline{R}$, and can just consider solving a linear equation system with a lower triangle matrix. Inserting this in equation (5.10) we can simplify the differential equation to

$$\ddot{\gamma}(t) = (q_1 | \dots | q_m) \overline{R}^{-T} \begin{pmatrix} \dot{\gamma}(t)^T H_{f_1}(\gamma(t)) \dot{\gamma}(t) \\ \vdots \\ \dot{\gamma}(t)^T H_{f_m}(\gamma(t)) \dot{\gamma}(t) \end{pmatrix}.$$

□

It is important to remember, that $q_1, \dots, q_m$ and R^* by the QR-decomposition depend smoothly on t, furthermore matrix inversion and transposing are also smooth operations. Thereby equation (5.2) is itself an ordinary differential equation to which we can apply an ODE-solver. The necessary steps and its implementation will be presented in the following algorithm.

Algorithm 5.2.4: ODEStep

Data: $p \in \mathcal{M}, v \in T_p\mathcal{M}$.

Result: $\ddot{\gamma}(t)$ for the geodesic γ, s.t $\gamma(t) = p, \dot{\gamma}(t) = v$.

Global: $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subset \mathbb{R}^n$, Hessians $h_{f_1}, \ldots, h_{f_m}$ of $f_1, \ldots, f_m$, Jacobian J_f.

Function: ODEStep(p, v):

$J \leftarrow J_f(p)^T$

$[Q, R] \leftarrow QR(J)$ `/* QR-decomposition */`

$R \leftarrow (R_{i,j})^T_{1 \le i,j \le m}$

$Q \leftarrow (Q_{i,j})_{\substack{1 \le i \le n \\ 1 \le j \le m}}$

$H \leftarrow \left(v^T H_{f_i} v\right)_{1 \le i \le m}$

$T \leftarrow R \backslash H$ `/* R\H solves Rx = H */`

$w = -Q \cdot T$

Thereby we can now also build an algorithm to calculate a geodesic.

Algorithm 5.2.5: LineSearch/Geodesic

Data: $p \in \mathcal{M}, v \in T_p\mathcal{M}$.

Result: Approximated points P on the curve $\gamma_{p,v}$, such that the stopping criteria c is satisfied, optional the transported velocity.

Global: $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subset \mathbb{R}^n$, Hessians $h_{f_1}, \ldots, h_{f_m}$ of $f_1, \ldots, f_m$, Jacobian J_f, stopping criteria c.

Function: LineSearch/Geodesic(p, v, h):

$g \leftarrow$ ODEStep `/* Algorithm 5.2.4 */`

$v \leftarrow \mathrm{Proj}_{T_p\mathcal{M}}(v)$

$[p_1, \ldots, p_5, v_1, \ldots, v_5] \leftarrow$ AB5Setup(g, p, v, h)

$P \leftarrow [p_1, \ldots, p_5]$

$V \leftarrow [v_1, \ldots, v_5]$

while $c = True$ **do**

  $[p_1, \ldots, p_5, v_0, \ldots, v_5] \leftarrow$ AB5$(g, p_1, \ldots, p_5, v_1, \ldots, v_5, h)$

  $P \leftarrow [P, p_5]$

  $V \leftarrow [V, v_5]$

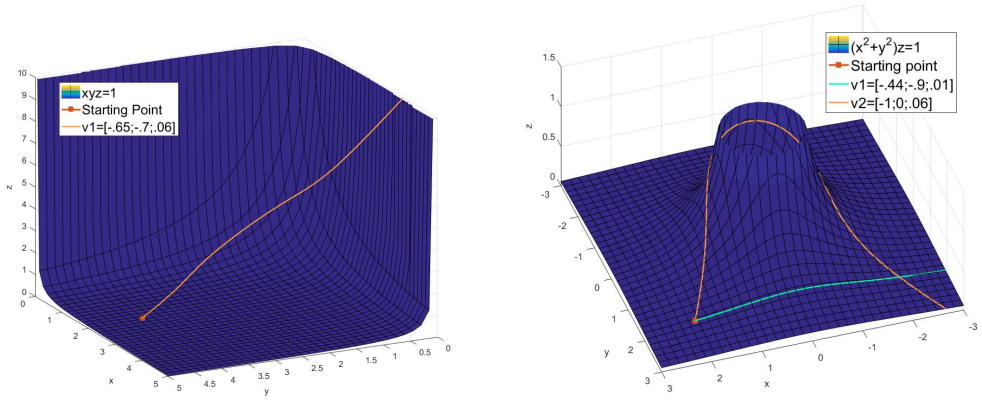
end

return P `/* Optional output V. */`

In general the stopping criteria of Algorithm 5.2.5 will be a certain amount

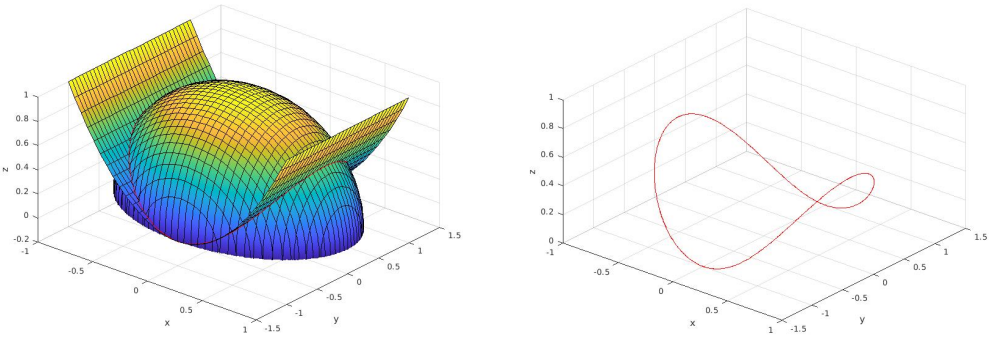
of steps, thus $c(P)$ returns False if $|P| > s_{\max}$ for some $s_{\max} \in \mathbb{N}$. In general the stopping criteria can also be used as line search criteria, for example if $c(p_4, p_5)$ returns False if $L(p_4) < L(p_5)$. Then Algorithm 5.2.5 stops once there is an increase in the objective function, of course then handling of the first four points has to be considered especially.

The Figures 5.1 a) and b) show approximations to geodesics on hypersurfaces by Algorithm 5.2.5. The algorithm can also be used to approximate curves, as Figures 5.2 and 5.3 show.



(a) Geodesic on $\mathcal{V}(x_1x_2x_3 - 1)$ **(b)** Two geodesic on $\mathcal{V}((x_1^2 + x_2^2)x_3 - 1)$

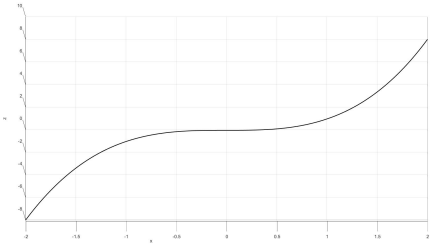
Figure 5.1: Examples of Geodesics calculated on hypersurfaces by Algorithm 5.2.5.



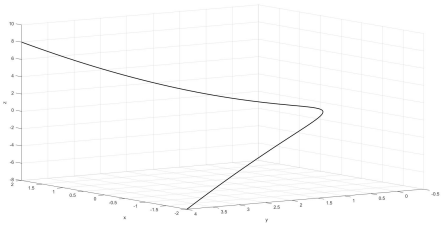
(a) 3D plot by MatLab **(b)** Application of Algorithm 5.2.5

Figure 5.2: Reconstruction of $\mathcal{V}\left((x_1^2 + x_2^2 + x_3^2) - 1\right) \cap \mathcal{V}(x_1^2 - x_2)$ by Algorithm 5.2.5.

(a) Projection to x_1, x_2 plane.



(b) Projection to x_1, x_3 plane.



(c) 3D Image

Figure 5.3: Reconstruction of the Twisted Cubic $\mathcal{V}\left(x_1^2 - x_2\right) \cap \mathcal{V}(x_1^3 - x_3)$ by Algorithm 5.2.5.

5.3 Approximating the Logarithm

5.3.1 Logarithm in Codimension One

In the most elementary case the manifold $\mathcal{M}$ is described by only one polynomial. As the number of constraints determines the codimension of the manifold, this is the case when $\mathcal{M}$ is a hypersurface and $m = 1$. Then we have $f = f_1, \lambda = \lambda_{1,1} = \frac{1}{|\nabla f(\gamma(t))|}$ and

$$\ddot{\gamma}(t) = -\lambda^2 \dot{\gamma}(t)^T H_f(\gamma(t))\dot{\gamma}(t)\nabla f(\gamma(t)) =: g(t, \gamma(t), \dot{\gamma}(t)) \tag{5.12}$$

as $N_p\mathcal{M} = \langle \nabla f(p) \rangle$. Thus we will consider the differential equation

$$\frac{d}{dt}\begin{pmatrix}\gamma(t)\\ \dot{\gamma}(t)\end{pmatrix} = \begin{pmatrix}\dot{\gamma}(t)\\ \frac{-1}{|\nabla f(\gamma(t))|^2}\dot{\gamma}(t)^T H_f(\gamma(t))\dot{\gamma}(t)\nabla f(\gamma(t))\end{pmatrix} \tag{5.13}$$

In preparation to the parallel transport of a tangent vector we will now discuss the logarithmic problem. To construct an algorithm for the logarithmic

problem, one first needs the Jacobians $J_\gamma g$ and $J_{\dot\gamma} g$

$$J_\gamma g = \frac{\dot\gamma^T H_f(\gamma)\dot\gamma}{||\nabla f(\gamma)||} H_f(\gamma) + \left(\nabla f|\ldots|\nabla f\right)^T \left(\frac{\operatorname{diag}\left(\left(\dot\gamma^T H_{\delta_{\gamma_i} f}(\gamma)\dot\gamma\right)_{i=1}^n\right)}{||\nabla f(\gamma)||^2} - \frac{\operatorname{diag}\left(H_f(\gamma)\nabla f(\gamma)\right)}{||\nabla f(\gamma)||^4} \right) \tag{5.14}$$

$$J_{\dot\gamma} g = -2 \cdot \frac{\left(H_f(\gamma)\cdot\nabla f(\gamma)\right)^T}{||\nabla f(\gamma)||^2}. \tag{5.15}$$

The Jacobians together with Algorithm 3.2.2 allow the construction of an algorithm to approximate the logarithm of two given points.

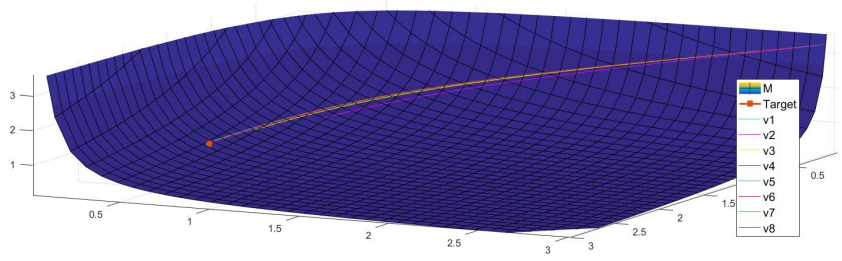


Figure 5.4: Shooting Approach to Logarithmic problem on $\mathcal{V}\left(x_1x_2x_3 - 1\right)$ by Algorithm 5.3.1.

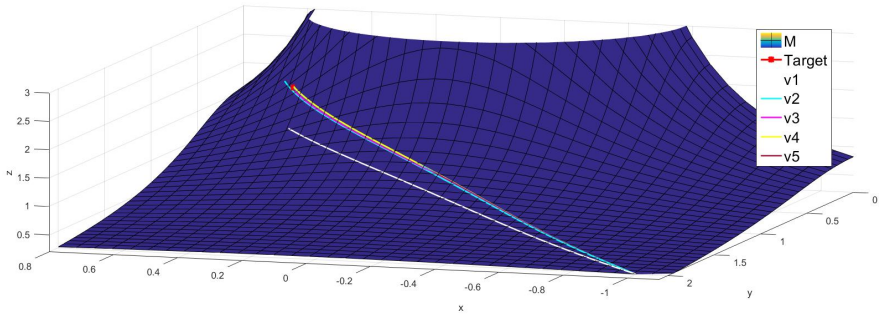


Figure 5.5: Shooting Approach to Logarithmic problem on $\mathcal{V}\left((x_1^2 + x_2^2)x_3 - 1\right)$ by Algorithm 5.3.1.

Algorithm 5.3.1: Log

Data: $p, q \in \mathcal{M}$.

Result: $v \in T_p\mathcal{M}$ such that $||\gamma_{p,v}(1) - q|| < \varepsilon$.

Global: $\mathcal{M} = \mathcal{V}(f) \subset \mathbb{R}^n$, Hessians $H_f, H_{\delta_{x_1} f}, \ldots, H_{\delta_{x_n} f}$, gradient ∇f, step length $h \in \mathbb{R}$, approximation criteria ε.

Function: Log(p, q):

$v \leftarrow \text{Proj}(q - p)$ `/* Algorithm 4.7.7 */`

func $\leftarrow (0 \mid \text{CorStep})$ `/* Algorithm 3.2.1 */`

while *True* **do**

  $[P, V] \leftarrow \text{LineSearch}(p, v, h)$ `/* Algorithm 5.2.5 */`

  if $\left|\left|P_{|P|} - q\right|\right| < \varepsilon$ **then**

    return v

  else

    $\overline{P}_1 \leftarrow (P_1 \mid 0_{\mathbb{R}^{n\times n}})$

    $\overline{V}_1 \leftarrow (V_1 \mid I_n)$

    $[\overline{P}_1, \ldots, \overline{P}_5, \overline{V}_1, \ldots, \overline{V}_5] \leftarrow$ AB5Setup(ODECorStep, $\overline{P}_1, \overline{V}_5, h$) `/* Algorithm 3.2.2 */`

    for $6 \leq k \leq |P|$ **do**

      $[\overline{P}_1, \ldots, \overline{P}_5, \overline{V}_1, \ldots, \overline{V}_5] \leftarrow AB5(func, \overline{P}_1, \ldots, \overline{P}_5, \overline{V}_1, \ldots, \overline{V}_5, h)$

      $(\overline{P}_5)_{*,1} \leftarrow P_k$

      $(\overline{V}_5)_{*,1} \leftarrow V_k$

    end

    $\text{CorM} \leftarrow (\overline{P}_{5i,j})_{\substack{1\leq i\leq n \\ 2\leq j\leq n+1}}$

    $w \leftarrow q - w$

    $\text{CorV} \leftarrow \text{CorM}\backslash w$ `/* CorM\w solves CorM · x = w */`

    $v \leftarrow v - \text{CorV}$

  end

end

The Figures 5.4 and 5.5 show plots of Algorithm 5.3.1 on the varieties $\mathcal{V}\left((x_1^2 + x_2^2)x_3 - 1\right)$ and $\mathcal{V}\left(x_1x_2x_3 - 1\right)$.

For the logarithmic problem it is to remember that geodesics are only locally a minimal curve between points. Therefore, as Figure 5.6 shows, it is possible for multiple geodesics to reach the same point.

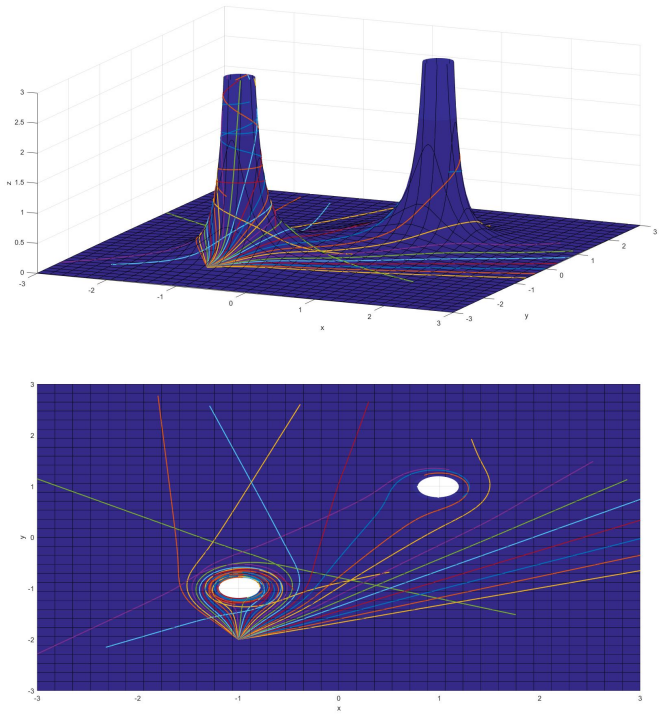


Figure 5.6: Fan of geodesics

$\mathcal{M} = \mathcal{V}\left(\left((x_1-1)^2+(x_2-1)^2\right)\left((x_1+1)^2+(x_2+1)^2\right)-x_3\right)$,
$\gamma_i(0) = (-1,-2,\frac{1}{13})^T), \dot{\gamma}_i(0) = \text{proj}_{T_p\mathcal{M}}\left(\sin((i-10)\frac{\pi}{50}), \cos((i-10)\frac{\pi}{50}), 0\right)$ for $1 \leq i \leq 30$.

In Figure 5.6 the hypersurface

$$\mathcal{V} = \mathcal{V}\left(\left((x_1-1)^2+(x_2-1)^2\right)\left((x_1+1)^2+(x_2+1)^2\right)-x_3\right)$$

was considered for testing Algorithm 5.3.1. There are two important things to note about this figure. First of all wee see multiple geodesics intersect, showing that the logarithmic problem does not necessarily have a unique solution for arbitrary points. The second thing to note about the plot is related to the singularities of $\mathcal{V}$. Around those the geodesics can curve backwards and therefore intersect itself, even thought the geodesics are not periodic. This can lead to high inaccuracies when trying to solve the logarithmic problem with Algorithm 5.3.1. As corrections occur in discrete steps the next iteration might get close to a singularity and therefore worsen the approximation instead.

Another problem, which the algorithm can encounter in general, is if starting and target point lie on opposing sites of a singularity. For example with $\mathcal{V}\left((x_1^2+x_2^2)x_3-1\right)$ and $p=(-1,-1,0.5)^T, q=(1,1,0.5)^T$ and initial guess $v_0=\mathrm{proj}_{T_pM}(p-q)$ the algorithm will at each step try to reach q through the singularity. This could in general be fixed with a randomisation approach to the initial velocity, if the algorithm gets stuck.

5.3.2 Logarithm with Multiple Constraints

In the general case with multiple restrictions one has to derive

$$\ddot{\gamma}=f(\gamma,\dot{\gamma})=Q\overline{R}^{-T}\begin{pmatrix}\dot{\gamma}^T H_{f_1}(\gamma)\dot{\gamma}\\ \vdots \\ \dot{\gamma}^T H_{f_m}(\gamma)\dot{\gamma}\end{pmatrix} \tag{5.16}$$

with $\overline{R}=R_{1\leq i,j\leq m}$, using γ and $\dot{\gamma}$. Now $\frac{d}{d\gamma}Q$ and $\frac{d}{d\gamma}R$ exist by Theorem 4.7.10 and can be calculated by Algorithm 4.7.11. With $\overline{R}=R_{1\leq i,j\leq m}$ and projections being smooth this gives

$$\frac{d}{d\gamma}\overline{R}^{-T}=-\overline{R}^{-T}\left(\frac{d}{d\gamma}\overline{R}\right)^T\overline{R}^{-T} \tag{5.17}$$

Deriving (5.16) by γ_i

$$\begin{aligned}\frac{d}{d\gamma_i}f(\gamma,\dot{\gamma})=&Q\overline{R}^{-T}\begin{pmatrix}\dot{\gamma}^T H_{\frac{df_1}{dx_i}}(\gamma)\dot{\gamma}\\ \vdots \\ \dot{\gamma}^T H_{\frac{df_m}{dx_i}}(\gamma)\dot{\gamma}\end{pmatrix}+\\ &\left(\left(\frac{d}{d\gamma_i}Q\right)\overline{R}^{-T}-Q\overline{R}^{-T}\left(\frac{d}{d\gamma_i}R\right)^T_{1\leq i,j\leq n}\overline{R}^{-T}\right)\begin{pmatrix}\dot{\gamma}^T H_{f_1}(\gamma)\dot{\gamma}\\ \vdots \\ \dot{\gamma}^T H_{f_m}(\gamma)\dot{\gamma}\end{pmatrix}.\end{aligned} \tag{5.18}$$

On the other hand deriving (5.16) by $\dot{\gamma}_i$ gives

$$\frac{d}{d\dot{\gamma}_i}f(\gamma,\dot{\gamma})=2Q\overline{R}^{-T}\begin{pmatrix}(H_{f_1}\dot{\gamma}(t))_i\\ \vdots \\ (H_{f_m}\dot{\gamma}(t))_i\end{pmatrix}. \tag{5.19}$$

Thus one gets

$$J_{\dot\gamma} f = 2Q\overline{R}^{-T} \begin{pmatrix} (H_{f_1}\dot\gamma(t))^T \\ \vdots \\ (H_{f_m}\dot\gamma(t))^T \end{pmatrix}. \tag{5.20}$$

The actual implementation is given in the following algorithm, which can then again be used with Algorithm 3.2.2 to construct an algorithm to approximately solve the logarithmic problem of two given points, similar to Algorithm 5.3.1.

Algorithm 5.3.2: Jacobian $J_\gamma f$

Data: $p \in \mathcal{M}$.

Result: Jacobian $J = J_\gamma f$ for the differential equation $\ddot\gamma = f(\gamma, \dot\gamma)$.

Global: $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m) \subset \mathbb{R}^n$, Hessians $H_{\frac{df_i}{dx_j}}$ for $1 \le i \le m, 1 \le j \le n$.

Function: JacobianGamma(p):

$A \leftarrow (\nabla f_1(p), \ldots, \nabla f_m(p))$

$[Q, R] \leftarrow QR(A)$

$R \leftarrow R^T_{1\le i,j\le m}$

$J \leftarrow \text{zeros}(m, n)$

$v \leftarrow \begin{pmatrix} \dot\gamma^T H_{f_1}(\gamma)\dot\gamma \\ \vdots \\ \dot\gamma^T H_{f_m}(\gamma)\dot\gamma \end{pmatrix}$

$v \leftarrow R\backslash v$

for $1 \le i \le n$ **do**

  $[dQ, dR] \leftarrow \text{DiffQR}(p, i)$ `/* Algorithm 4.7.11 */`

  $dR \leftarrow dR^T_{1\le i,j\le n}$

  $t1v \leftarrow QR \begin{pmatrix} \dot\gamma^T H_{\frac{df_1}{dx_i}}(\gamma)\dot\gamma \\ \vdots \\ \dot\gamma^T H_{\frac{df_m}{dx_i}}(\gamma)\dot\gamma \end{pmatrix}$

  $t2v \leftarrow R\backslash(dR \cdot v)$

  $J_{*,i} \leftarrow dQ \cdot v - Q \cdot t2v + t1v$

end

return J

5.4 Parallel transport on manifolds

In Definition 4.5.18 parallel transport is defined by a vector field $X \in \mathfrak{X}(\mathcal{M})$ that satisfies $\nabla_{\dot{\gamma}(t)} X(t) = 0$, which similarly to a geodesic represents an ordinary differential equation. In this case thought we lack a theorem like Theorem 4.5.8, which allows us to transform the differential equation into a manageable expression. Another approach to avoid the differential equations is to approximate the parallel transport by constructing parallelograms along the curve, which represent the parallel transport. Because of their appearance in three dimensions those are called *ladder schemes*. One of those ladder schemes is Pole Ladder, which is discussed in [Pen18]. About the accurateness of this ladder schemes the author gives the following:

Theorem 5.4.1
In a sufficient small convex normal neighbourhood of a point in an affine connection space $(\mathcal{M}, \nabla)$ with symmetric connection, the error on one step of Pole Ladder to transport the vector $u \in T_{\exp_m(-v)}\mathcal{M}$ along a geodesic segment of a tangent vector $v \in T_m\mathcal{M}$ to $T_{\exp_m(v)}\mathcal{M}$ is given by

$$u'' - u = -\frac{1}{12}(\nabla_v R(u,v)(5u+v) + \nabla_u R(u,v)(v+2u)) + O(||v+u||^5).$$

Here R is the Riemannian curvature tensor

$$R(X,Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z,$$

for $X, Y, Z \in \mathfrak{X}(\mathcal{M})$. Furthermore if $\mathcal{M}$ is a Riemannian symmetric manifold, pole ladder is exact almost surely.

In a Riemannian symmetric manifold, the pole ladder is exact if the points $\exp_m v$ and $\exp_m w$ are not elements of the cut locus $\mathrm{Cut}(\exp_m(-v))$. In Riemannian geometry the cut locus is roughly the set of all other points for which there are multiple minimizing geodesics connecting the two points, for example for $p \in S^n$ one has $\{-p\} = \mathrm{Cut}(p)$. In a Riemannian symmetric space the cut locus has measure zero. For us it is important, that Pole Ladder is of third order in one step. For more details and a proof of Theorem 5.4.1 see [Pen18].

5.4.1 Pole ladder

Given two points $p_1, p_2 \in \mathcal{M}$ we want to transport a tangent vector $v \in T_{p_1}\mathcal{M}$ to $T_{p_2}\mathcal{M}$. Letting $s \in T_{p_1}\mathcal{M}$ such that $p_2 = \gamma_{p_1,s}(1)$ together with v we want to construct a parallelogram on $\mathcal{M}$. We need to calculate the points $x = \gamma_{p_1,v}(1)$ and $m = \gamma_{p_1,s}(\frac{1}{2})$, making m the midpoint between p_1 and p_2. Next we need to find the direction $w := \log(x, m)$ which satisfies $m = \gamma_{x,w}(1)$. Now we calculate $y = \gamma_{x,w}(2)$ and get the transport of v by $v' := -\log(p_2, y)$.

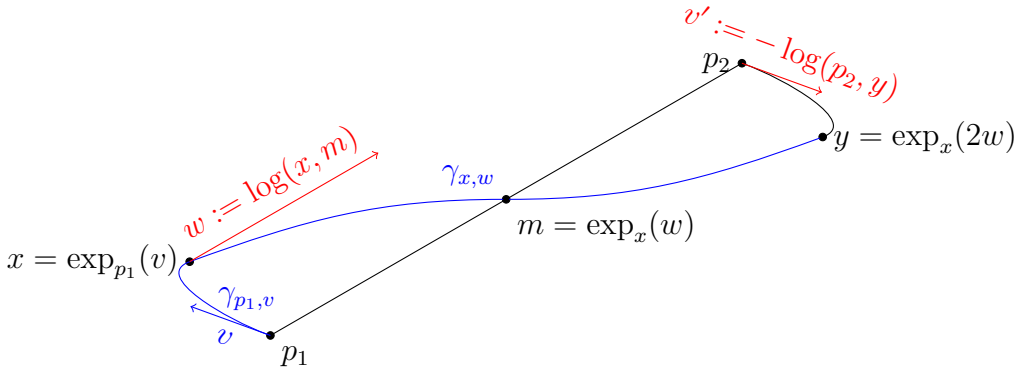


Figure 5.7: Single step of Pole ladder

A single step of Pole Ladder is implemented by the following algorithm.

Algorithm 5.4.2: ParallelStep

Data: Starting point $p = \gamma(0)$, midpoint $m = \gamma(s)$, endpoint $q = \gamma(2s) \in \mathcal{M}, w \in T_p\mathcal{M}$.
Result: Parallel Transport $\tilde{w} \in T_q\mathcal{M}$ of w.
Global: $\mathcal{M} = \mathcal{V}(f_1, \ldots, f_m)$, step length $h \in \mathbb{R}$
Function: ParallelStep(p, m, q, w)

1 $x \leftarrow \text{Geodesic}(p, w, steps = \frac{1}{h})$
2 $v \leftarrow \text{Log}(x, m)$
3 $y \leftarrow \text{Geodesic}(x, v, steps = \frac{2}{h})$
4 $\tilde{w} \leftarrow -\text{Log}(q, y)$
5 **return** $\tilde{w}$

In Algorithm 5.4.2 the transported vector is calculated by the logarithm in line 4. For parallel transport along longer curves it is preferable though to chain multiple applications of Pole Ladder for small segments of the curve,

which results in calculating parallel transports at unnecessary points. But then Pole Ladder has the very advantage, when chaining multiple iterations one actually has to calculate the parallel transport only at the final application, as the following figure indicates.

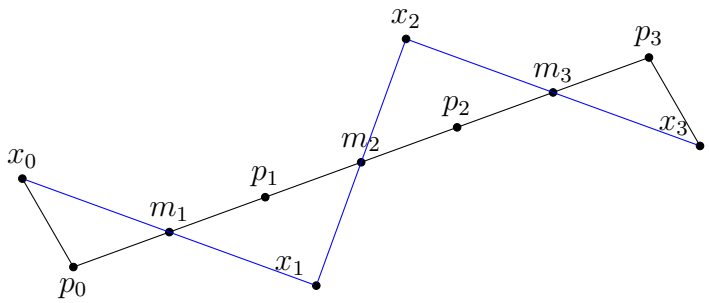
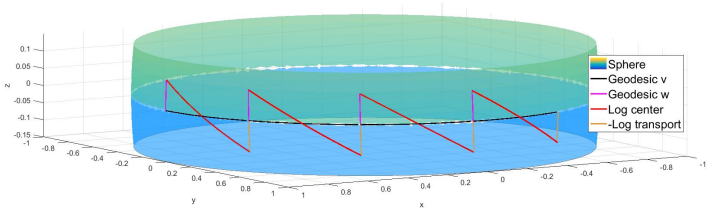
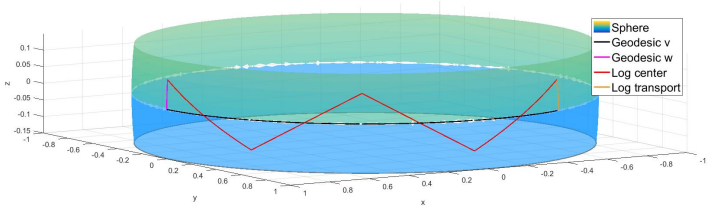


Figure 5.8: Multiple steps of Pole ladder

The following figure shows a comparison of both Algorithms on the sphere. The calculation of the logarithmic problem is solving an boundary value problem. This makes it require more calculations compared to approximating the initial value problems for the geodesics. Therefore it is obvious that Algorithm 5.4.3 is preferable to Algorithm 5.4.2.



(a) Four Steps of Pole Ladder with Log at every step calculated with Algorithm 5.4.2.



(b) Four Steps of Pole Ladder without Log at every step calculated with Algorithm 5.4.3.

Figure 5.9: Comparison of Ladder schemes on the sphere.

The variation and later used implementation for parallel transport is given in

the following algorithm, which calculates the transported vector only once.

Algorithm 5.4.3: Parallel transport along geodesic

Data: Sequence of points $P = \{p_0, \dots, p_{|P|}\}$ along a geodesic, $v \in T_{p_0}\mathcal{M}$.

Result: Approximate of the parallel transport $\tilde{v} \in T_{p_{|P|}}\mathcal{M}$ of v.

Global: $\mathcal{M} = \mathcal{V}(f_1, \dots, f_m)$, initial steps l of displacement in the direction v, distance $k \in \mathbb{N}$ between steps of Pole ladder, step length $h \in \mathbb{R}$.

$j \leftarrow 1$
$q \leftarrow \text{Geodesic}(p_0, v, steps = l)$
$Cont \leftarrow 1$
while $Cont = 1$ **do**
  if $j + k < |P|$ **then**
    $m \leftarrow p_{\lfloor \frac{2j+k}{2} \rfloor}$
    $j \leftarrow j + k$
  else if $j - |P| = 1$ **then**
    $m \leftarrow \frac{p_{|P|-1} + p_{|P|}}{2}$
    $Cont \leftarrow 0$
  else
    $m \leftarrow p_{\lfloor \frac{2j+|P|}{2} \rfloor}$
    $Cont \leftarrow 0$
  end
  $w \leftarrow \text{Log}(q, m)$
  $q \leftarrow \text{Geodesic}(q, w, steps = \frac{2}{h})$
end
$\tilde{v} \leftarrow \frac{1}{l}(-1)^{\lceil \frac{|P|}{k} \rceil}\text{Log}(p_{|P|}, q)$
return $\tilde{v}$

In Theorem 5.4.1 it is stated that Pole Ladder is only almost surely exact if $\mathcal{M}$ is symmetric. The following Figure shows an application of Pole Ladder by Algorithm 5.4.3 on the non-symmetric manifold $\mathcal{V}(x_1x_2x_3 - 1)$.

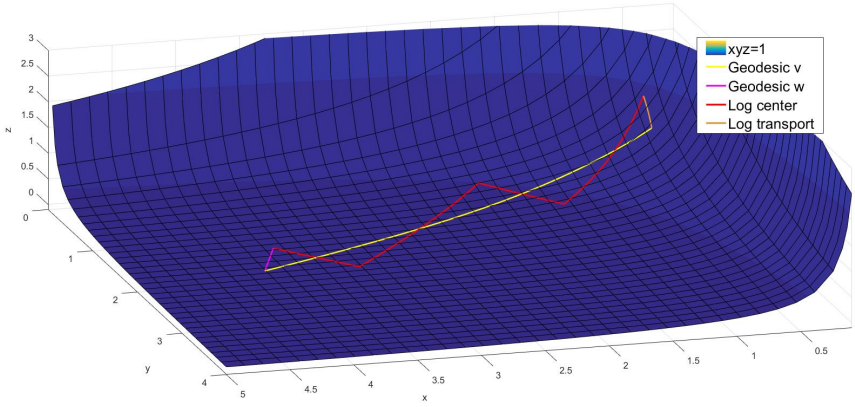


Figure 5.10: Length distortion by parallel transport with Algorithm 5.4.3.

For the starting point $p = \begin{pmatrix} 4 & 2 & \frac{1}{8} \end{pmatrix}^T$ and direction $v = \text{proj}_{T_p\mathcal{M}} \begin{pmatrix} -3 & -1 & 3 \end{pmatrix}^T$ the transport of $w = \text{proj}_{T_p\mathcal{M}} \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$ was calculated.
Figure 5.10 indicates that the logarithmic problem is not only the bottle neck in performance, but also in accuracy. At each iteration of Pole Ladder we do only hit the target point to an assumed accuracy. This error can escalate over multiple iterations. Also the fewer steps between each iteration of Pole Ladder, the more iterations of Pole Ladder are necessary. At each iteration the logarithmic problem also needs to be solved. For demonstration Algorithm 5.4.3 was run on the problem of Figure 5.2 with varying configurations. The movement was always calculated with 40.000 steps and a step length of 10^{-3}. The number of steps between each iteration of Pole Ladder was set to 100 and 1000, furthermore the accuracy with which a solution was accepted was set to either $10^{-2}, 10^{-3}, 10^{-4}$ or 10^{-5}.

Steps	Accuracy	Distortion	Steps	Accuracy	Distortion
100	10^{-2}	0.4420	1000	10^{-2}	0.9388
	10^{-3}	0.7064		10^{-3}	0.9686
	10^{-4}	1.4875		10^{-4}	1.0390
	10^{-5}	1.4880		10^{-5}	1.0391

Table 5.1: Length distortion of parallel transport on $\mathcal{V}(x_1x_2x_3 - 1)$ calculated with Algorithm 5.4.3

For 100 steps 400 iterations are needed to parallel transport from the starting to the end point. With an enlargement of 1.001 at each iteration the error accounts to $1.001^{400} \approx 1.4915$. In comparison for 1000 steps only 40 iterations are needed and assuming the same enlargement the error accounts to a $1.001^{40} \approx 1.0408$. Considering the accuracy it is to note that the initial guess for $\log(p, q)$ is always $\text{proj}_{T_p\mathcal{M}}(q - p)$. Projecting the vector will in general make it smaller and for $p, q \in \mathcal{M}$ the euclidean distance will not taking into account the curvature of the manifold and therefore

$$||q - p||_{eucl.} \leq \text{dist}(p, q).$$

Therefore Algorithm 5.3.1 will for this initial guess always return an approximate $\tilde{q}$ with

$$\text{dist}(p, \tilde{q}) \leq \text{dist}(p, q).$$

Therefore a smaller vector is more likely to be accepted at each iteration, again amplifying with each application of Pole Ladder.

Chapter 6

Application to test problems

In 2021 Buhmann and Siegel published a paper on Broyden class updates for large scale optimization [BS21]. The algorithm was applied to unconstrained optimization problems and uses coordinate transformations to efficiently calculate updates in the BFGS formula with an easy variation for a limited memory variation. Especially for high dimensional problems the approach showed good results, compared to other algorithms by [Noc80, Nocedal], with similar rate of convergence or even better. For application problems from the CUTEr/CUTEst list were chosen. The package is a collection of test problems in different dimensions, with and without constraints. Many of those problems origin from optimisation problems encountered in engineering, chemistry, physics or similar. The problems themself are decoded in Standart Input Format (SIF), decodable to Fortran subroutines by the package *SifDec*. In [GOT15] a compiler is introduced to read those SIF files by various other programming languages and also MATLAB.
In [BS21] the method was implemented in Java. By cross compiling Java and MATLAB with the before mentioned implementation of CUTE into MATLAB a 'black box' was created which evaluated the objective function and its gradient. The problems are initialized by the command 'cutest2matlab' outside of MATLAB and called by MATLAB by the cutest_setup() command. It then passes a description of the problem, including dimension and starting point, to MatLab.
The here discussed methods are itself implemented in MATLAB and will also use the same approach to generate the objective functions by CUTE problems. It is here to note that, while a lot of the CUTE problems are implemented with constraints, we will replace those by our own manifold

constraints. The method is applied to equality constraints and in the case of multistep methods also requires information which is not provided by the CUTE problems, like third order differentials in the form of Hessians $H_{\delta_{x_i} f_j}$. Thereby in this dissertation only the objective function and its gradient are used for testing. The constraints and their derivatives will be implemented in MATLAB directly.

In all tests the step length to approximate the geodesics will be $h = 10^{-3}$ or $h = 10^{-4}$ and will remain constant throughout the applied method. Additionally if $||d_k|| > 1$ we will normalise the descent direction, as elsewise the steps in calculating the geodesics are too big and one is likely to miss the constraints.

After 5000 iterations we will consider the method failed and stop. Other than that the algorithm terminates if $\text{proj}_{T_p\mathcal{M}}(\nabla L(p)) < 10^{-4}$, if achieved we will also store all interesting data once the norm of the gradient falls below 10^{-2} and 10^{-3}. If a steepest descent step is done, either by the method itself or a restart, and no decrease in the function value is achieved then the numbers of iterations will be set to 5000 straight, thus stopping the algorithm, as without decrease of the step length no further improvement is possible.

On one hand we will do the line search by pointwise evaluating the objective function, returning the point with lowest objective function value. This will be an approach resembling a prefect line search. For comparison we will consider a line search by the Armijo conditions (A1) and (A2) and thereby make us of Algorithm 2.3.2. The coefficients for the Armijo conditions are set by $c_1 = 10^{-2}, c_2 = 0.9$ and the decline rate is set to $\beta = 0.8$.

In both cases the points for the line search will be calculated by Algorithm 5.2.5 and the ordinary differential equation will be evaluated at each iterate by Algorithm 5.2.4. Also depending on the step size s returned we will adjust the maximal step length in the following way

$$\text{maxstep} \leftarrow \begin{cases} 3 \cdot \text{maxsteps} & \text{if } 0.8 \cdot \text{maxsteps} \leq s \\ \text{maxsteps} & \text{if } 0.2 \cdot \text{maxstep} \leq s < 0.8 \cdot \text{maxsteps} \\ \frac{\text{maxsteps}}{2} & \text{if } s < 0.2 \cdot \text{maxsteps} \end{cases}$$

Initially we will start with maxstep = 15000.

The logarithm calculated by Algorithm 5.3.1 is considered successful if the distance between the calculated point and the target is less then the step

length.
The following data will be gathered:

L	the value of the objective function...
∇L	the norm of the gradient ∇L...
f	the number of function calls of L...
It	the number of total iterations...
Rest	the number of restarts needed...
LogCor	the number of logarithms that need correction...

at the iterate that satisfies the affiliated criteria. If more than one criteria is satisfied the difference to the previous function value will be given in the first row of the following columns instead of the actual function value. Furthermore f_ϵ will denote the norm of the error of the constrain at the last iterate.

6.1 Test Constraints

The tested manifolds will all be compact ensuring the existence of a minima.

6.1.1 Norm-2-Sphere

The most simple manifold we consider is the n-dimensional norm-2-sphere, for testing we choose

$$f = x_1^2 + \cdots + x_n^2 - 1, \tag{6.1}$$

thus $\mathcal{M} = \mathcal{V}(f) = \left\{x \in \mathbb{R}^n | \, ||x||^2 = 1\right\}$. One then has

$$\nabla f(x) = 2x, \tag{6.2}$$

$$H_f(x) = 2I, \tag{6.3}$$

$$H_{\partial x_k f}(x) = 0 \text{ for all } 1 \leq k \leq n. \tag{6.4}$$

The starting point is $p_0 = \frac{\mathbb{1}}{||\mathbb{1}||_2}$. By [Gor21] S_n is a symmetric space. The respective test data is given in Section A.1 of the appendix (see page 91).

6.1.2 Norm-4-Sphere

The next still simple but more complex manifold is a n-dimensional norm-4-sphere. It is an example with non-zero derivatives of third order. Thereby

$$f = x_1^4 + \cdots + x_n^4 - \frac{||\mathbb{1}||_4^4}{||\mathbb{1}||_2^4}, \tag{6.5}$$

then

$$\begin{aligned} \nabla f(x)_i &= 4x_i^3, \\ H_f(x) &= 12 \cdot \operatorname{diag}(x_1^2, \ldots, x_n^2) \end{aligned} \tag{6.6}$$

and

$$H_{\partial x_i f} = 24 H_{i,i}, \tag{6.7}$$

with $H_{i,j}$ is defined by $H_{i,j_{s,t}} = \delta_{i,s}\delta_{j,t}$. The starting point is the same as for the sphere $p_0 = \frac{\mathbb{1}}{||\mathbb{1}||_2}$. The respective test data is given in Section A.2 of the appendix (see page 97).

6.2 Norm-2-Sphere Intersected

The first manifold described by multiple functions will be given by the variety of

$$\begin{aligned} f_1 &= x_1^4 + x_2^4 - s_1, \\ f_2 &= \sum_{i=1}^{n} x_i^2 - 1, \\ f_3 &= x_1^3 \cdot x_{n-1}^3 - x_n - s_3, \end{aligned} \tag{6.8}$$

with constants s_1 and s_3 are chosen such that the starting point p_0 evaluates as $f_1(p) = f_2(p) = f_3(p) = 0$. The respectable gradients are

$$\nabla f_1 = 4 \begin{pmatrix} x_1^3 \\ x_2^3 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \nabla f_2 = 2x, \nabla f_3 = \begin{pmatrix} 3x_1^2 x_{n-1}^3 \\ 0 \\ \vdots \\ 0 \\ 3x_1^3 x_{n-1}^2 \\ -1 \end{pmatrix} \tag{6.9}$$

and the Hessians are given by

$$H_{f_1} = 12 \begin{pmatrix} x_1^2 & 0 & 0 & \dots & 0 \\ 0 & x_2^2 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}, H_{f_2} = 2I,$$

$$H_{f_3} = \begin{pmatrix} 6x_1x_{n-1}^3 & 0 & \dots & 0 & 9x_1^2x_{n-1}^2 & 0 \\ 0 & \ddots & & & 0 & \vdots \\ \vdots & & \ddots & & & \vdots \\ 0 & \dots & \dots & \dots & 0 & \vdots \\ 9x_1^2x_{n-1}^2 & 0 & \dots & 0 & 6x_1^3x_{n-1} & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 \end{pmatrix}. \tag{6.10}$$

The third order derivatives $H_{\delta_{x_i} f_j}$ are easily derived from (6.10). The respective test data is given in Section A.3 of the appendix (see page 103).

6.3 Stiefel Manifold

The Stiefel manifold consists of matrices, thus either the objective function is also defined on matrices or one has to transform the matrices to column vectors. The respective functions and derivates are given in Section 4.9.1. The CUTE problems have vector input therefore we will transform the matrices to vectors by choosing fitting dimensions. For the respective problems the following dimensions were chosen.

Problem	Dimension	Rows	Columns	Manifold
ACCOP30	72	9	8	ST(8,9)
AVGASA	8	4	2	ST(2,4)
DEGENLPA	20	5	4	ST(4,5)
NASH	72	9	8	ST(8,9)

In contrast to the other manifolds, f_ϵ is evaluated with the Frobeniusnorm of $f(X) = X^TX - I$, not only on the necessary components of f. Also the Stiefel manifold is a compact Lie-group, thus by [Gor21] a symmetric space. The respective test data is given in Section A.4 of the appendix (see page 109).

6.4 Conclusion

6.4.1 Step Size

As expected a smaller step size h gave a smaller error f_ϵ on the manifold. Furthermore in most cases a smaller step size also increased the efficiency of the algorithm. By making finer steps the algorithm was able to get closer a real minima with the perfect line search and was less likely to stop early because no more improvement was possible with the Steepest Gradient method (compare Table A.1). On the other hand it is obvious that further decreasing the step length at some point will lead to a horrendous number of calculations, therefore one is incentivized to choose a step length as large as possible, while remaining as small as necessary. What step size is useable depends on the curvature of the manifold itself. Comparing $f_1 = ||x|| - 10^2$ to $f_2 = ||x|| - 1$ the variety induced by f_2 has a much higher curvature than the unit sphere induced by f_1 and thereby the accuracy profits from a smaller step length. Test were also run with step length $h = 10^{-2}$ but resulted in either worse results than for $h = 10^{-3}$ or even diverged from the manifold. Therefore they were omitted from further testing for the given manifolds.
Another approach was made to start with a fixed step length and reduce the step length if two iterates were relatively close to each other. Unfortunately no useful data was gathered from this approach as step lengths plunged rapidly in the test, leading to iterations in the magnitude of 10^6 before external termination.

6.4.2 Choice of Method

The data in general shows that one cannot a priori decide if the steepest descend, conjugated gradient or a LBFGS method will work best for an arbitrary objective function. This is mostly observed with the objective function AVGASA over the various constraints (compare Tables A.2, A.8, A.14, A.20). Although overall the data suggests that making wider use of the search space improves the optimisation significantly. This is best observed with the functions NASH and ACCOP30 on the 4-Sphere (compare Tables A.7 and A.10), in which the Algorithm only converged with the LBFGS method. Additionally using more than 5 vectors for the L-BFGS algorithm was tested on the sphere and indicated that using more vectors increases the performance.

This is contrary to the expected behaviour as parallel transport along closed curves changes the vector, therefore ill behaviour of old iterates was expected. Still, even with fewer iterations in total required, the actual runtime drastically increases due to the additional parallel transport steps required. Thus it seems preferable to stick to fewer stored iterations.

6.4.3 Choice of Method by Line Search

The data suggests that the LBFGS method profits vastly from an exact line search and therefore with an inexact line search the LBFGS method loses its advantage over the steepest descent method. Still in both cases the number of function calls needed drastically decreases.

6.4.4 Accuracy

The gradient of problem NASH on the returned iterate, before projection to the tangent space, still was in the magnitude of 10^3. This shows that missing the constraints slightly can affect the algorithms and the results massively, emphasising the advantage of a small step length.

6.5 Prospect of Further Applications

6.5.1 Projection and Self Correction

With the developed algorithms an optimisation method can also be applied iteratively to calculate a least distance projection to $\mathcal{M} = \mathcal{V}(f_1, \dots, f_s) \subset \mathbb{R}^n$. Let $x_0 \in \mathbb{R}^n$ a starting point. Then firstly one considers the unconstrained optimisation problem $x_1 = \operatorname{argmin}_{x \in \mathbb{R}^n} f_1^2(x)$ with starting point x_0 and thereafter we consider the constrained problems

$$x_i = \operatorname*{argmin}_{x \in \mathcal{V}(f_1, \dots, f_{i-1})} f_i^2(x) \tag{6.11}$$

with starting point x_{i-1}. At last consider

$$x_{s+1} = \operatorname*{argmin}_{x \in \mathcal{V}(f_1, \dots, f_s)} d(x_0, x), \tag{6.12}$$

with d being the distance function. Because one cannot ensure that a point x_i is actually a minimum of $f_i(x)$ one should chooses f_i^2 instead.

6.5.2 Generalisation to Splines and Zero Fibres

The approach presented makes the here developed algorithms also applicable to a surface approximated by splines of at least forth order. This is especially important as even simple surfaces in the Euclidean space may be complicated to describe by polynomials. The two dimensional torus can be described by

$$\left(\sqrt{x^2+y^2}-r_1\right)^2+z^2=r_2^2 \Leftrightarrow (x^2+y^2+z^2+r_1^2-r_2^2)^2-4r_1^2(x^2+y^2).$$

This is already a polynomial of total degree four. For the three dimensional torus one has

$$\left(\sqrt{\left(\sqrt{x^2+y^2}-r_1\right)^2+z^2}-r_2\right)^2+w^2=r_3^2.$$

Cancelling the roots by squaring already gives a polynomial of total degree 8. By this approach the total degree of the polynomial doubles with each dimension. Furthermore in industrial applications a surface might already be described by splines.

On the other hand the construction mostly depends on the Implicit Function Theorem 4.2.3 and $\nabla f_1, \ldots, \nabla f_s$ being linearly independent at each point, not necessarily $f_1, \ldots, f_s$ being polynomials. Therefore the approach seems extendable to zero fibres in general. This would for example allow the rescaling of gradients by switching to rational functions as the following Lemma indicates.

Lemma 6.5.1
Let $f, g : \mathbb{R} \to \mathbb{R}$ polynomials such that $\mathcal{V}(g) \cap \mathcal{V}(f) = \emptyset$, then for all $x \in \mathcal{V}(f)$ holds

$$\nabla \frac{f}{g}(x) = \frac{\nabla f(x)}{g(x)}.$$

Furthermore $\mathcal{V}\left(\frac{f}{g}\right) = \mathcal{V}(f)$.

Proof. For $x \in \mathbb{R}^n$ such that $g(x) \neq 0$ one has

$$\begin{aligned}
\nabla \frac{f}{g}(x)_i &= \frac{d}{dx_i}\frac{f}{g}(x) \\
&= \frac{\left(\frac{d}{dx_i}f\right)(x)g(x) - \left(\frac{d}{dx_i}g\right)(x)f(x)}{g(x)^2} \\
&= \frac{\left(\frac{d}{dx_i}f\right)(x)}{g(x)} - \frac{\left(\frac{d}{dx_i}g\right)(x)f(x)}{g(x)^2}
\end{aligned}$$

Thus $\nabla \frac{f}{g}(x) = \frac{\nabla f(x)}{g(x)} - \frac{(\nabla g(x))f(x)}{g(x)^2}$ and for $x \in \mathcal{V}(f)$ this gives

$$\nabla \frac{f}{g}(x) = \frac{\nabla f(x)}{g(x)}.$$

□

This could be applied to address ill conditioning in the QR-decomposition in the line search by Algorithm 5.2.5, respectively Algorithm 5.2.4. Addressing zero fibres in general would of course increase the area of application tremendously.

6.5.3 Constraint with Failure on Null Sets

A geodesic was also calculated by Algorithm 5.2.5 on the double cone, which is described by $\mathcal{V}(x^2 + y^2 - z^2)$.

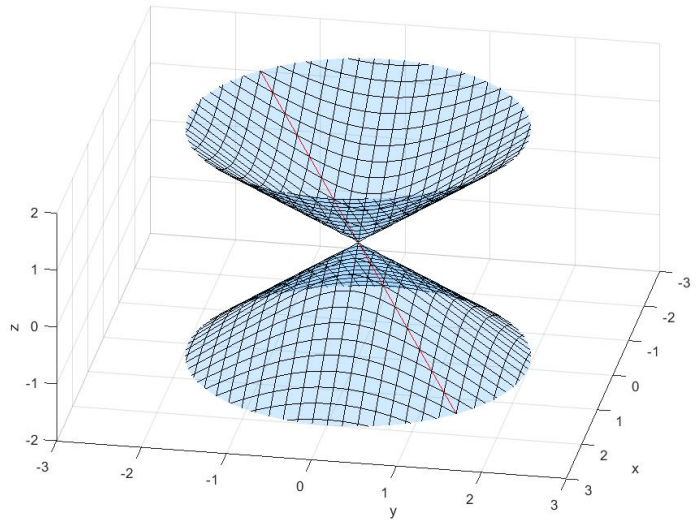


Figure 6.1: Geodesic on the double cone by Algorithm 5.2.5.

Figure 6.1 shows that the algorithm has no problem with the non-smooth point origin, as long this point has not to be evaluated. This suggest if the set of all points with linearly dependent gradients

$$C := \{x \in \mathcal{V}(f_1, \ldots, f_m) \mid \nabla f_1(x), \ldots, \nabla f_m(x) \text{ linearly dependent}\}$$

is a null set then Algorithm 5.2.5 will return an approximation to a geodesic, or a geodesic-like curve, almost surely.

6.5.4 Inequality Constraints

Like other methods which work with inequality constraints (compare [Pow78]) it would be possible to extend the approach discussed here to inequality constraints. The algorithms introduced here allow the transport of previous gradient and descent directions along constraints. This allows their usage in methods which require further descent directions. Considering $f_1, \dots, f_s \in \mathbb{R}[x_1, \dots, x_n]$ and $L : \mathbb{R}^n \to \mathbb{R}$ with the constraints $f_1(x) \geq 0$ and $f_2(x) = \cdots = f_s(x) = 0$ one is able to check if the constraint f_1 is active at the current iterate, that is $f_1(x_k) = 0$, or inactive elsewise. In the later case the constrain needs only to be checked in such a sense that the line search should not return an iterate which violates the constrain. Apart from this the method just considers the constraints $f_2(x) = \cdots = f_n(x) = 0$. Furthermore if f_1 is an active inequality constrain, one would still have to check if the descent direction can lead to a violation of the constrain. If $g_k = \nabla L(x_k)$ and $g_k^T \nabla f_1(x_k) > 0$, then by first order approximation

$$f_1(x_k + \alpha g_k) = f_1(x_k) + \alpha g_k^T \nabla f_1(x_k) > f_1(x_k) = 0. \tag{6.13}$$

Therefore moving in the direction of g_k does not violate f_1 for sufficiently small $\alpha \in \mathbb{R}$. In this case one can again consider the optimisation problem constrained only to $\mathcal{V}(f_2, \dots, f_s)$.
If on the other hand $g_k^T \nabla f_1(x_k) \leq 0$, then we would again consider the constrain $\mathcal{V}(f_1, \dots, f_s)$. In both cases one would have to check, that the descent direction also satisfies $d_k^T g_k \geq 0$ similar to (6.13).

6.5.5 Application to Further Methods

As mentioned in Subsection 6.4.2 using the LBFGS method with more previous data showed an improvement of the method. Therefore testing with other methods which approximate the Hessian on a subspace, compare [BS21], should be tested.

Data

A.1 Test Data - Sphere

A.1.1 Perfect Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 7.01 \cdot 10^{-4}$		
-294.3104	1.2485	4320	5000	$-$	$-$
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.19 \cdot 10^{-4}$		
-294.2967	$2.7398 \cdot 10^{-1}$	2683	5000	22	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.74 \cdot 10^{-5}$		
-294.2942	$8.8710 \cdot 10^{-2}$	2159	5000	1	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 2.20 \cdot 10^{-5}$		
-294.2943	$9.5117 \cdot 10^{-3}$	2240	35	0	0
$* - 5.9604 \cdot 10^{-8}$	$7.2796 \cdot 10^{-4}$	2258	41	0	0
$* - 3.1605 \cdot 10^{-10}$	$4.0688 \cdot 10^{-5}$	2276	47	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 8.48 \cdot 10^{-7}$		
-294.2938	$9.9227 \cdot 10^{-3}$	24161	185	$-$	$-$
$* - 5.7544 \cdot 10^{-7}$	$9.7886 \cdot 10^{-4}$	24748	245	$-$	$-$
$* - 5.7992 \cdot 10^{-9}$	$9.8335 \cdot 10^{-5}$	25319	303	$-$	$-$
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.10 \cdot 10^{-5}$		
-294.2941	$9.7140 \cdot 10^{-3}$	31255	1102	13	0
$* - 5.4115 \cdot 10^{-7}$	$9.7232 \cdot 10^{-4}$	31841	1159	13	0
$* - 5.1303 \cdot 10^{-9}$	$9.8370 \cdot 10^{-5}$	32388	1215	13	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.59 \cdot 10^{-7}$		
-294.2938	$2.6417 \cdot 10^{-3}$	20932	21	0	0
$* - 5.2106 \cdot 10^{-9}$	$6.9319 \cdot 10^{-4}$	20964	23	0	0
$* - 5.2654 \cdot 10^{-10}$	$4.0178 \cdot 10^{-5}$	21002	26	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 2.00 \cdot 10^{-7}$		
-294.2938	$4.8803 \cdot 10^{-3}$	21621	31	0	0
$* - 1.4326 \cdot 10^{-8}$	$6.0119 \cdot 10^{-4}$	21681	34	0	0
$* - 1.8576 \cdot 10^{-10}$	$6.0516 \cdot 10^{-5}$	21745	39	0	0
$D = 72$	$h = 10^{-4}$		$f_\epsilon = 8.67 \cdot 10^{-8}$		
-294.2938	$3.4087 \cdot 10^{-3}$	54245	16	0	0
$* - 7.2541 \cdot 10^{-9}$	$4.7997 \cdot 10^{-4}$	62992	17	0	0
$* - 1.3881 \cdot 10^{-10}$	$6.4384 \cdot 10^{-5}$	63014	20	1	0

Table A.1: Evaluation Perfect-ACCOP30 Sphere

Name:AVGASA			Dim : 8		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.97 \cdot 10^{-6}$		
-7.1609	$2.9326 \cdot 10^{-3}$	826	5	–	–
$* - 3.7450 \cdot 10^{-7}$	$6.4044 \cdot 10^{-4}$	915	6	–	–
$* - 2.1720 \cdot 10^{-8}$	$3.5549 \cdot 10^{-5}$	1106	8	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 2.38 \cdot 10^{-6}$		
-7.1609	$5.1781 \cdot 10^{-3}$	702	4	0	0
$* - 1.1568 \cdot 10^{-6}$	$9.1873 \cdot 10^{-4}$	790	5	0	0
$* - 4.0679 \cdot 10^{-8}$	$4.0999 \cdot 10^{-5}$	975	7	0	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.98 \cdot 10^{-6}$		
-7.1609	$3.7576 \cdot 10^{-3}$	726	4	0	0
$* - 6.5827 \cdot 10^{-7}$	$8.1052 \cdot 10^{-5}$	821	5	0	0
$* - 1.6054 \cdot 10^{-10}$	$6.4610 \cdot 10^{-5}$	928	6	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 1.98 \cdot 10^{-6}$		
-7.1609	$3.7576 \cdot 10^{-3}$	726	4	0	0
$* - 6.3743 \cdot 10^{-7}$	$6.4481 \cdot 10^{-4}$	1351	6	0	0
$* - 2.1025 \cdot 10^{-8}$	$5.3339 \cdot 10^{-5}$	2064	9	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 1.97 \cdot 10^{-8}$		
-7.1609	$2.9691 \cdot 10^{-3}$	8150	5	–	–
$* - 3.9083 \cdot 10^{-7}$	$6.7485 \cdot 10^{-4}$	9039	6	–	–
$* - 2.3618 \cdot 10^{-8}$	$3.8812 \cdot 10^{-5}$	10909	8	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 2.39 \cdot 10^{-8}$		
-7.1609	$6.2096 \cdot 10^{-3}$	6946	4	0	0
$* - 1.6633 \cdot 10^{-6}$	$1.9793 \cdot 10^{-4}$	8726	6	0	0
$* - 1.6982 \cdot 10^{-9}$	$4.0460 \cdot 10^{-5}$	9595	7	0	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.98 \cdot 10^{-8}$		
-7.1609	$5.2605 \cdot 10^{-3}$	7192	4	0	0
$* - 1.2916 \cdot 10^{-6}$	$6.3551 \cdot 10^{-5}$	8128	5	0	0
$* - 1.1915 \cdot 10^{-10}$	$3.5232 \cdot 10^{-5}$	9040	6	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 1.98 \cdot 10^{-8}$		
-7.1609	$5.2605 \cdot 10^{-3}$	7192	4	0	0
$* - 1.2653 \cdot 10^{-6}$	$7.2465 \cdot 10^{-4}$	13007	6	0	0
$* - 2.6404 \cdot 10^{-8}$	$2.9447 \cdot 10^{-5}$	21186	9	0	0
$D = 8$	$h = 10^{-4}$		$f_\epsilon = 1.98 \cdot 10^{-8}$		
-7.1609	$5.2562 \cdot 10^{-3}$	7192	4	0	0
$* - 1.2645 \cdot 10^{-6}$	$7.1342 \cdot 10^{-4}$	13008	6	0	0
$* - 2.5462 \cdot 10^{-8}$	$4.3516 \cdot 10^{-5}$	21047	9	0	0

Table A.2: Evaluation Perfect-AVGASA Sphere

Name:DEGENLPA			Dim : 20		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.01 \cdot 10^{-6}$		
-226.0816	$1.0094 \cdot 10^{-3}$	2141	3	–	–
$* - 2.3481 \cdot 10^{-9}$	$9.6574 \cdot 10^{-5}$	2147	4	–	–
$* - 2.1686 \cdot 10^{-11}$	$9.2397 \cdot 10^{-6}$	2153	5	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.01 \cdot 10^{-6}$		
-226.0816	$1.8652 \cdot 10^{-4}$	2142	3	1	0
$* - 8.0206 \cdot 10^{-11}$	$1.7861 \cdot 10^{-5}$	2148	4	1	0
$* - 8.2423 \cdot 10^{-13}$	$1.7088 \cdot 10^{-6}$	2154	5	1	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.00 \cdot 10^{-6}$		
-226.0816	$4.8781 \cdot 10^{-5}$	2530	2	0	0
$* - 1.1369 \cdot 10^{-13}$	$4.8732 \cdot 10^{-5}$	2533	3	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 1.00 \cdot 10^{-6}$		
-226.0816	$4.8781 \cdot 10^{-5}$	2530	2	0	0
$* - 1.1369 \cdot 10^{-13}$	$4.8732 \cdot 10^{-5}$	2533	3	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.00 \cdot 10^{-8}$		
-226.0815	$2.7666 \cdot 10^{-3}$	21269	2	–	–
$* - 1.6937 \cdot 10^{-8}$	$1.4502 \cdot 10^{-5}$	21315	3	–	–
$* - 1.1369 \cdot 10^{-13}$	$1.3846 \cdot 10^{-5}$	21319	4	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 2.00 \cdot 10^{-8}$		
-226.0815	$2.7666 \cdot 10^{-3}$	21269	2	1	0
$* - 1.6937 \cdot 10^{-8}$	$1.4502 \cdot 10^{-5}$	21315	3	2	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 2.00 \cdot 10^{-8}$		
-226.0815	$2.7666 \cdot 10^{-3}$	21269	2	0	0
$* - 1.6376 \cdot 10^{-8}$	$1.9157 \cdot 10^{-4}$	29980	3	0	0
$* - 8.4043 \cdot 10^{-11}$	$2.9358 \cdot 10^{-6}$	30097	9	1	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 2.00 \cdot 10^{-8}$		
-226.0815	$2.7666 \cdot 10^{-3}$	21269	2	0	0
$* - 1.6376 \cdot 10^{-8}$	$1.9157 \cdot 10^{-4}$	29980	3	0	0
$* - 8.3560 \cdot 10^{-11}$	$4.0357 \cdot 10^{-5}$	30755	9	0	0
$D = 20$	$h = 10^{-4}$		$f_\epsilon = 2.00 \cdot 10^{-8}$		
-226.0815	$2.7666 \cdot 10^{-3}$	21269	2	0	0
$* - 1.6921 \cdot 10^{-8}$	$5.7921 \cdot 10^{-5}$	30432	3	0	0

Table A.3: Evaluation Perfect-DEGE Sphere

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 2.09 \cdot 10^{-6}$		
-1501.0016	$7.9502 \cdot 10^{-3}$	1789	7	–	–
$* - 8.3078 \cdot 10^{-8}$	$9.9839 \cdot 10^{-4}$	1798	10	–	–
$* - 1.3270 \cdot 10^{-9}$	$6.2801 \cdot 10^{-5}$	1810	14	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 2.07 \cdot 10^{-6}$		
-1501.0016	$8.6079 \cdot 10^{-3}$	1780	4	2	0
$* - 9.7293 \cdot 10^{-8}$	$5.3121 \cdot 10^{-4}$	1792	8	2	0
$* - 3.7198 \cdot 10^{-10}$	$6.6664 \cdot 10^{-5}$	1801	11	2	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 2.06 \cdot 10^{-6}$		
-1501.0015	$7.4282 \cdot 10^{-3}$	2496	8	0	0
$* - 1.9826 \cdot 10^{-8}$	$2.9397 \cdot 10^{-4}$	2537	11	0	0
$* - 1.0436 \cdot 10^{-10}$	$5.3039 \cdot 10^{-5}$	2553	17	2	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 2.06 \cdot 10^{-6}$		
-1501.0015	$4.7260 \cdot 10^{-3}$	2411	5	0	0
$* - 7.3558 \cdot 10^{-9}$	$8.9974 \cdot 10^{-4}$	3056	7	0	0
$* - 9.5520 \cdot 10^{-10}$	$9.4308 \cdot 10^{-5}$	3074	14	3	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.18 \cdot 10^{-8}$		
-1501.0000	$1.1331 \cdot 10^{-3}$	17707	4	–	–
$* - 4.3633 \cdot 10^{-10}$	$5.8392 \cdot 10^{-5}$	17716	5	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 3.16 \cdot 10^{-8}$		
-1501.0000	$4.2050 \cdot 10^{-4}$	17711	5	1	0
$* - 6.0936 \cdot 10^{-11}$	$4.1145 \cdot 10^{-5}$	17719	6	1	0
$* - 1.1369 \cdot 10^{-12}$	$3.4959 \cdot 10^{-5}$	17722	7	1	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 4.50 \cdot 10^{-8}$		
-1501.0000	$8.1775 \cdot 10^{-3}$	24130	13	0	0
$* - 2.2627 \cdot 10^{-8}$	$2.6138 \cdot 10^{-4}$	24591	19	0	0
$* - 2.8649 \cdot 10^{-11}$	$2.2107 \cdot 10^{-5}$	24610	22	1	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 3.48 \cdot 10^{-8}$		
-1501.0000	$6.8192 \cdot 10^{-3}$	31441	11	0	0
$* - 1.5679 \cdot 10^{-8}$	$8.9871 \cdot 10^{-5}$	33695	19	1	0
$* - 6.8212 \cdot 10^{-13}$	$8.9859 \cdot 10^{-5}$	33698	20	1	0
$D = 72$	$h = 10^{-4}$		$f_\epsilon = 3.49 \cdot 10^{-8}$		
-1501.0000	$1.1782 \cdot 10^{-4}$	48330	11	0	0
$* - 5.0022 \cdot 10^{-12}$	$1.1731 \cdot 10^{-5}$	48340	13	1	0

Table A.4: Evaluation Perfect-NASH Sphere

A.1.2 Armijo Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 7.18 \cdot 10^{-4}$		
-294.3108	1.2485	5313	5000	$-$	$-$
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 7.27 \cdot 10^{-4}$		
-294.3110	1.2485	5429	5000	726	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 3.03 \cdot 10^{-5}$		
-294.2945	$5.8424 \cdot 10^{-2}$	574	5000	6	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 4.73 \cdot 10^{-5}$		
-294.2950	$9.1373 \cdot 10^{-3}$	645	91	3	0
$* - 6.9684 \cdot 10^{-8}$	$3.9917 \cdot 10^{-4}$	784	118	4	0
$* - 9.2399 \cdot 10^{-10}$	$1.7288 \cdot 10^{-4}$	986	5000	6	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 1.22 \cdot 10^{-6}$		
-294.2938	$6.1008 \cdot 10^{-3}$	1241	184	$-$	$-$
$* - 3.0855 \cdot 10^{-7}$	$9.2902 \cdot 10^{-4}$	1679	217	$-$	$-$
$* - 6.3399 \cdot 10^{-9}$	$8.9985 \cdot 10^{-5}$	2363	274	$-$	$-$
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.22 \cdot 10^{-6}$		
-294.2938	$9.9744 \cdot 10^{-3}$	1375	195	121	0
$* - 2.0181 \cdot 10^{-7}$	$8.3309 \cdot 10^{-4}$	1803	260	121	0
$* - 1.3470 \cdot 10^{-9}$	$8.7975 \cdot 10^{-5}$	2185	318	121	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 4.73 \cdot 10^{-7}$		
-294.2938	$8.7433 \cdot 10^{-3}$	893	130	3	0
$* - 2.4324 \cdot 10^{-8}$	$9.9063 \cdot 10^{-4}$	1240	198	3	0
$* - 2.2277 \cdot 10^{-10}$	$8.7869 \cdot 10^{-5}$	1531	258	3	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 5.86 \cdot 10^{-7}$		
-294.2938	$9.4275 \cdot 10^{-3}$	1104	154	4	0
$* - 6.5100 \cdot 10^{-8}$	$9.2983 \cdot 10^{-4}$	1508	220	4	0
$* - 9.4599 \cdot 10^{-10}$	$8.7434 \cdot 10^{-5}$	1838	275	4	0
$D = 72$	$h = 10^{-4}$		$f_\epsilon = 1.41 \cdot 10^{-7}$		
-294.2938	$9.7602 \cdot 10^{-3}$	321	35	4	0
$* - 5.0404 \cdot 10^{-8}$	$9.2718 \cdot 10^{-4}$	361	43	4	0
$* - 1.5831 \cdot 10^{-8}$	$5.5457 \cdot 10^{-5}$	523	80	4	0

Table A.5: Evaluation Armijo-ACCOP30 Sphere

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 5.00 \cdot 10^{-3}$		
-1504.7471	3.0938	35052	5000	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 5.00 \cdot 10^{-3}$		
-1504.7471	3.0938	35052	5000	4999	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 4.99 \cdot 10^{-3}$		
-1504.7411	3.0551	25165	5000	1	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 1.68 \cdot 10^{-5}$		
-1501.0126	$9.4809 \cdot 10^{-3}$	631	66	3	0
$* - 3.9275 \cdot 10^{-8}$	$7.2326 \cdot 10^{-4}$	870	86	4	0
$* - 2.8217 \cdot 10^{-10}$	$7.8975 \cdot 10^{-5}$	945	94	6	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.68 \cdot 10^{-7}$		
-1501.0002	$2.2791 \cdot 10^{-3}$	306	42	–	–
$* - 1.7656 \cdot 10^{-9}$	$1.1774 \cdot 10^{-4}$	317	43	–	–
$* - 7.0486 \cdot 10^{-12}$	$6.0882 \cdot 10^{-6}$	328	44	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 2.58 \cdot 10^{-7}$		
-1501.0002	$8.9484 \cdot 10^{-3}$	317	41	31	0
$* - 2.7539 \cdot 10^{-8}$	$6.9104 \cdot 10^{-4}$	359	47	31	0
$* - 1.7189 \cdot 10^{-10}$	$9.0381 \cdot 10^{-5}$	400	53	31	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 4.45 \cdot 10^{-7}$		
-1501.0003	$9.3477 \cdot 10^{-3}$	1035	130	0	0
$* - 3.0614 \cdot 10^{-8}$	$9.8078 \cdot 10^{-4}$	1355	170	0	0
$* - 4.6430 \cdot 10^{-10}$	$4.2606 \cdot 10^{-5}$	1836	240	1	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 1.08 \cdot 10^{-7}$		
-1501.0001	$9.2346 \cdot 10^{-3}$	592	61	2	0
$* - 2.9317 \cdot 10^{-8}$	$9.3318 \cdot 10^{-4}$	675	69	2	0
$* - 3.3365 \cdot 10^{-9}$	$9.1600 \cdot 10^{-5}$	703	71	2	0
$D = 72$	$h = 10^{-4}$		$f_\epsilon = 6.26 \cdot 10^{-8}$		
-1501.0000	$5.8723 \cdot 10^{-3}$	221	20	1	0
$* - 1.5313 \cdot 10^{-8}$	$4.9068 \cdot 10^{-4}$	316	23	2	0
$* - 2.1842 \cdot 10^{-8}$	$4.2957 \cdot 10^{-5}$	672	40	5	0

Table A.6: Evaluation Armijo-NASH Sphere

A.2 Test Data 4Sphere

A.2.1 Perfect Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 4.83 \cdot 10^{-5}$		
-234.1395	1.2585	16645	5000	$-$	$-$
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 3.92 \cdot 10^{-6}$		
-234.0998	$2.9964 \cdot 10^{-1}$	2039	5000	10	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.87 \cdot 10^{-6}$		
-234.0981	$7.5147 \cdot 10^{-3}$	2022	128	0	0
$* - 1.8364 \cdot 10^{-6}$	$8.8073 \cdot 10^{-4}$	2218	189	0	0
$* - 1.9209 \cdot 10^{-8}$	$9.9660 \cdot 10^{-5}$	2380	237	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 2.72 \cdot 10^{-6}$		
-234.0988	$8.5861 \cdot 10^{-3}$	2198	147	0	0
$* - 1.0322 \cdot 10^{-6}$	$8.7953 \cdot 10^{-4}$	2369	186	0	0
$* - 3.6964 \cdot 10^{-8}$	$8.7055 \cdot 10^{-5}$	2640	252	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.58 \cdot 10^{-8}$		
-234.0963	$9.8696 \cdot 10^{-3}$	46966	2995	$-$	$-$
$* - 5.5710 \cdot 10^{-5}$	$6.3915 \cdot 10^{-3}$	68623	5000	$-$	$-$
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 4.55 \cdot 10^{-7}$		
-234.0967	$9.8436 \cdot 10^{-3}$	42358	2496	6	0
$* - 6.0628 \cdot 10^{-5}$	$5.3899 \cdot 10^{-3}$	69236	5000	6	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.89 \cdot 10^{-8}$		
-234.0964	$9.5768 \cdot 10^{-3}$	17568	113	0	0
$* - 3.2160 \cdot 10^{-6}$	$9.6012 \cdot 10^{-4}$	18344	161	0	0
$* - 5.3304 \cdot 10^{-8}$	$6.4363 \cdot 10^{-6}$	19224	209	1	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 2.78 \cdot 10^{-8}$		
-234.0964	$8.5038 \cdot 10^{-3}$	18701	125	0	0
$* - 2.9116 \cdot 10^{-6}$	$8.0824 \cdot 10^{-4}$	20088	190	0	0
$* - 3.2583 \cdot 10^{-8}$	$7.5218 \cdot 10^{-5}$	21325	250	0	0

Table A.7: Evaluation Perfect-ACCOP30 4-Sphere

Name:AVGASA			Dim : 8		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.43 \cdot 10^{-6}$		
-6.6974	$9.7596 \cdot 10^{-3}$	1009	13	–	–
$* - 5.0764 \cdot 10^{-6}$	$8.4323 \cdot 10^{-4}$	1431	20	–	–
$* - 5.1714 \cdot 10^{-8}$	$9.9836 \cdot 10^{-5}$	1875	27	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 9.23 \cdot 10^{-7}$		
-6.6974	$8.6800 \cdot 10^{-3}$	790	9	0	0
$* - 5.0019 \cdot 10^{-6}$	$8.3657 \cdot 10^{-4}$	1245	16	0	0
$* - 4.0130 \cdot 10^{-8}$	$7.7009 \cdot 10^{-5}$	1679	23	0	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.19 \cdot 10^{-6}$		
-6.6974	$4.3650 \cdot 10^{-3}$	681	7	0	0
$* - 7.0624 \cdot 10^{-7}$	$6.7172 \cdot 10^{-4}$	828	9	0	0
$* - 1.0867 \cdot 10^{-8}$	$7.9438 \cdot 10^{-5}$	878	10	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 1.18 \cdot 10^{-6}$		
-6.6974	$8.1728 \cdot 10^{-3}$	1001	8	0	0
$* - 2.3702 \cdot 10^{-6}$	$3.6696 \cdot 10^{-4}$	1260	12	0	0
$* - 3.2442 \cdot 10^{-9}$	$2.0112 \cdot 10^{-5}$	1360	14	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 1.43 \cdot 10^{-8}$		
-6.6973	$9.3991 \cdot 10^{-3}$	9971	13	–	–
$* - 4.3347 \cdot 10^{-6}$	$7.4256 \cdot 10^{-4}$	14088	20	–	–
$* - 4.0558 \cdot 10^{-8}$	$8.8132 \cdot 10^{-5}$	18434	27	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 9.22 \cdot 10^{-9}$		
-6.6973	$7.2547 \cdot 10^{-3}$	7709	9	0	0
$* - 3.5985 \cdot 10^{-6}$	$8.9720 \cdot 10^{-4}$	11490	15	0	0
$* - 5.5586 \cdot 10^{-8}$	$9.3011 \cdot 10^{-5}$	15883	22	0	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.20 \cdot 10^{-8}$		
-6.6973	$5.5937 \cdot 10^{-3}$	6690	7	0	0
$* - 1.0024 \cdot 10^{-6}$	$5.6124 \cdot 10^{-4}$	8012	9	0	0
$* - 7.6952 \cdot 10^{-9}$	$2.5684 \cdot 10^{-5}$	8918	11	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 1.19 \cdot 10^{-8}$		
-6.6974	$9.1029 \cdot 10^{-4}$	11743	11	0	0
$* - 3.1809 \cdot 10^{-8}$	$2.1430 \cdot 10^{-4}$	12493	12	0	0
$* - 1.1739 \cdot 10^{-9}$	$3.3945 \cdot 10^{-5}$	13002	13	0	0

Table A.8: Evaluation Perfect-AVGASA 4-Sphere

Name:DEGENLPA			Dim : 20		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.64 \cdot 10^{-5}$		
-161.9319	$9.6665 \cdot 10^{-3}$	6195	631	–	–
$* - 6.1371 \cdot 10^{-5}$	$9.9920 \cdot 10^{-4}$	12241	1479	–	–
$* - 4.3895 \cdot 10^{-6}$	$2.2244 \cdot 10^{-4}$	37137	5000	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.54 \cdot 10^{-5}$		
-161.9310	$9.9750 \cdot 10^{-3}$	6100	425	1	0
$* - 9.2148 \cdot 10^{-5}$	$9.9564 \cdot 10^{-4}$	13980	1576	1	0
$* - 4.3617 \cdot 10^{-6}$	$5.3198 \cdot 10^{-4}$	38081	5000	1	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 8.68 \cdot 10^{-7}$		
-161.9193	$8.9245 \cdot 10^{-3}$	2619	52	0	0
$* - 3.6382 \cdot 10^{-7}$	$9.6723 \cdot 10^{-4}$	2659	61	0	0
$* - 1.3066 \cdot 10^{-9}$	$8.0793 \cdot 10^{-5}$	2681	67	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 1.21 \cdot 10^{-6}$		
-161.9196	$5.7168 \cdot 10^{-3}$	2722	43	0	0
$* - 1.9036 \cdot 10^{-6}$	$9.4167 \cdot 10^{-4}$	2911	68	0	0
$* - 1.4470 \cdot 10^{-8}$	$1.5373 \cdot 10^{-5}$	3157	95	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.52 \cdot 10^{-7}$		
-161.9188	$9.8337 \cdot 10^{-3}$	56517	1611	–	–
$* - 3.3767 \cdot 10^{-5}$	$9.8746 \cdot 10^{-4}$	121748	4772	–	–
$* - 2.2853 \cdot 10^{-7}$	$1.0207 \cdot 10^{-3}$	126459	5000	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.69 \cdot 10^{-7}$		
-161.9187	$9.9877 \cdot 10^{-3}$	57405	1018	1	0
$* - 5.5953 \cdot 10^{-5}$	$9.8774 \cdot 10^{-4}$	128760	4172	1	0
$* - 9.4848 \cdot 10^{-7}$	$8.7829 \cdot 10^{-4}$	147913	5000	1	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.59 \cdot 10^{-8}$		
-161.9186	$5.4926 \cdot 10^{-3}$	24787	64	0	0
$* - 5.7039 \cdot 10^{-8}$	$6.2174 \cdot 10^{-4}$	24877	68	0	0
$* - 2.3033 \cdot 10^{-10}$	$4.3450 \cdot 10^{-5}$	24909	70	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 1.62 \cdot 10^{-8}$		
-161.9186	$7.2894 \cdot 10^{-3}$	26759	55	0	0
$* - 3.6747 \cdot 10^{-6}$	$2.7736 \cdot 10^{-4}$	28487	85	0	0
$* - 4.6827 \cdot 10^{-8}$	$8.7603 \cdot 10^{-5}$	30295	119	0	0

Table A.9: Evaluation Perfect-DEGE 4-Sphere

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.56 \cdot 10^{-3}$		
-679.3800	6.0860	16380	5000	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.54 \cdot 10^{-3}$		
-679.1762	6.0899	16326	5000	4939	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 5.33 \cdot 10^{-7}$		
-661.4928	$6.5159 \cdot 10^{-3}$	2007	64	1	0
$* - 4.8800 \cdot 10^{-7}$	$4.7777 \cdot 10^{-4}$	2285	79	2	0
$* - 8.2754 \cdot 10^{-6}$	$1.9764 \cdot 10^{-5}$	3850	169	4	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 5.73 \cdot 10^{-7}$		
-661.4923	$5.4001 \cdot 10^{-3}$	6678	18	0	0
$* - 1.0772 \cdot 10^{-6}$	$5.0623 \cdot 10^{-4}$	6745	22	0	0
$* - 1.6763 \cdot 10^{-7}$	$4.0129 \cdot 10^{-5}$	10394	57	1	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 3.94 \cdot 10^{-7}$		
-661.5038	$9.9633 \cdot 10^{-3}$	32949	3623	–	–
$* - 2.0417 \cdot 10^{-5}$	$5.1984 \cdot 10^{-3}$	40093	5000	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 5.24 \cdot 10^{-8}$		
-661.4998	$9.9522 \cdot 10^{-3}$	33814	3897	2	0
$* - 1.1826 \cdot 10^{-5}$	$3.1065 \cdot 10^{-3}$	39535	5000	2	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 5.27 \cdot 10^{-9}$		
-661.4991	$8.5130 \cdot 10^{-3}$	17411	28	0	0
$* - 1.9917 \cdot 10^{-5}$	$2.7149 \cdot 10^{-4}$	19074	64	0	0
$* - 3.8716 \cdot 10^{-8}$	$8.3053 \cdot 10^{-5}$	19471	85	2	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 5.24 \cdot 10^{-9}$		
-661.4991	$2.3995 \cdot 10^{-3}$	30661	29	0	0
$* - 6.6923 \cdot 10^{-7}$	$9.7160 \cdot 10^{-4}$	34297	39	0	0
$* - 1.5610 \cdot 10^{-7}$	$5.1292 \cdot 10^{-5}$	55665	55	0	0

Table A.10: Evaluation Perfect-NASH 4-Sphere

A.2.2 Armijo Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.37 \cdot 10^{-5}$		
-234.0798	$4.5264 \cdot 10^{-1}$	180	5000	$-$	$-$
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 7.67 \cdot 10^{-6}$		
-234.0833	$9.2850 \cdot 10^{-1}$	212	5000	20	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 2.45 \cdot 10^{-6}$		
-234.0986	$4.4662 \cdot 10^{-2}$	645	5000	5	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 1.14 \cdot 10^{-8}$		
-234.0964	$9.1679 \cdot 10^{-3}$	576	85	2	0
$* - 8.3354 \cdot 10^{-8}$	$8.6821 \cdot 10^{-4}$	720	112	2	0
$* - 5.1970 \cdot 10^{-9}$	$8.8255 \cdot 10^{-5}$	1015	167	2	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.25 \cdot 10^{-8}$		
-234.0963	$9.9219 \cdot 10^{-3}$	10035	1939	$-$	$-$
$* - 8.6490 \cdot 10^{-5}$	$3.6476 \cdot 10^{-3}$	46882	5000	$-$	$-$
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.98 \cdot 10^{-8}$		
-234.0963	$9.9680 \cdot 10^{-3}$	24150	3799	26	0
$* - 1.3530 \cdot 10^{-5}$	$8.0109 \cdot 10^{-3}$	31913	5000	26	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.47 \cdot 10^{-7}$		
-234.0965	$8.9713 \cdot 10^{-3}$	941	149	2	0
$* - 2.5464 \cdot 10^{-8}$	$9.5587 \cdot 10^{-4}$	1204	202	2	0
$* - 9.0250 \cdot 10^{-8}$	$5.9882 \cdot 10^{-4}$	31449	5000	2	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 5.74 \cdot 10^{-8}$		
-234.0964	$3.1631 \cdot 10^{-3}$	819	110	5	0
$* - 8.0775 \cdot 10^{-6}$	$9.7793 \cdot 10^{-4}$	1135	152	7	0
$* - 2.4221 \cdot 10^{-10}$	$6.9351 \cdot 10^{-5}$	1207	159	9	0

Table A.11: Evaluation Armijo-ACCOP30 4-Sphere

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.58 \cdot 10^{-3}$		
-679.5474	2.5192	32483	5000	$-$	$-$
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.58 \cdot 10^{-3}$		
-679.5474	2.5192	32483	5000	4999	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 7.97 \cdot 10^{-5}$		
-661.5706	$7.2049 \cdot 10^{-3}$	685	51	6	0
$* - 1.4039 \cdot 10^{-1}$	$6.0932 \cdot 10^{-4}$	6332	611	69	0
$* - 7.3484 \cdot 10^{-1}$	$4.1388 \cdot 10^{-5}$	29788	2921	380	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 4.98 \cdot 10^{-6}$		
-661.5585	$3.4750 \cdot 10^{-3}$	498	30	3	0
$* - 2.5846 \cdot 10^{-7}$	$8.7082 \cdot 10^{-4}$	631	44	3	0
$* - 1.0267 \cdot 10^{-8}$	$5.6060 \cdot 10^{-5}$	832	58	4	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 1.54 \cdot 10^{-5}$		
-661.6822	1.1532	33865	5000	$-$	$-$
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.54 \cdot 10^{-5}$		
-661.6822	1.1532	33865	5000	4999	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 3.62 \cdot 10^{-9}$		
-661.4991	$9.7806 \cdot 10^{-3}$	619	51	5	0
$* - 1.0795 \cdot 10^{-5}$	$4.2481 \cdot 10^{-4}$	764	62	7	0
$* - 1.8392 \cdot 10^{-7}$	$7.4577 \cdot 10^{-5}$	1074	83	9	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 5.04 \cdot 10^{-9}$		
-661.4991	$7.2125 \cdot 10^{-3}$	436	33	3	0
$* - 1.1456 \cdot 10^{-6}$	$9.9291 \cdot 10^{-4}$	702	54	3	0
$* - 1.5392 \cdot 10^{-8}$	$9.8141 \cdot 10^{-5}$	931	66	4	0

Table A.12: Evaluation Armijo-NASH 4-Sphere

A.3 Test Data Sphere Intersected

A.3.1 Perfect Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 8.32 \cdot 10^{-4}$		
-293.4697	1.2485	4695	5000	$-$	$-$
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.41 \cdot 10^{-4}$		
-293.4507	$1.3670 \cdot 10^{-1}$	2656	5000	10	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.53 \cdot 10^{-5}$		
-293.4472	$1.4346 \cdot 10^{-1}$	2117	5000	1	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 2.31 \cdot 10^{-5}$		
-293.4474	$2.7877 \cdot 10^{-3}$	2317	50	0	0
$* - 4.5518 \cdot 10^{-9}$	$5.9019 \cdot 10^{-4}$	2361	59	0	0
$* - 1.7684 \cdot 10^{-10}$	$9.1458 \cdot 10^{-5}$	2377	64	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 7.91 \cdot 10^{-7}$		
-293.4468	$9.9546 \cdot 10^{-3}$	23648	170	$-$	$-$
$* - 5.1693 \cdot 10^{-7}$	$9.9357 \cdot 10^{-4}$	24155	222	$-$	$-$
$* - 5.1640 \cdot 10^{-9}$	$9.3777 \cdot 10^{-5}$	24661	274	$-$	$-$
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.26 \cdot 10^{-5}$		
-293.4471	$6.2159 \cdot 10^{-3}$	38508	1263	14	0
$* - 9.0979 \cdot 10^{-8}$	$9.0467 \cdot 10^{-4}$	38810	1293	14	0
$* - 5.0615 \cdot 10^{-9}$	$9.0262 \cdot 10^{-5}$	39292	1339	14	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.69 \cdot 10^{-7}$		
-293.4468	$7.5212 \cdot 10^{-3}$	20594	22	0	0
$* - 2.9983 \cdot 10^{-8}$	$3.2624 \cdot 10^{-4}$	20640	26	0	0
$* - 9.9590 \cdot 10^{-11}$	$6.8426 \cdot 10^{-5}$	20708	29	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 2.08 \cdot 10^{-7}$		
-293.4468	$7.9622 \cdot 10^{-3}$	21321	29	0	0
$* - 3.8435 \cdot 10^{-8}$	$7.7354 \cdot 10^{-4}$	21371	33	0	0
$* - 2.2271 \cdot 10^{-10}$	$5.2842 \cdot 10^{-5}$	21437	38	0	0

Table A.13: Evaluation Perfect-ACCOP30 Sphere intersected

Name:AVGASA			Dim : 8		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 2.06 \cdot 10^{-6}$		
-6.8309	$4.1411 \cdot 10^{-3}$	672	4	–	–
$* - 7.4587 \cdot 10^{-7}$	$5.6274 \cdot 10^{-4}$	761	5	–	–
$* - 1.4336 \cdot 10^{-8}$	$8.1971 \cdot 10^{-5}$	854	6	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 3.05 \cdot 10^{-6}$		
-6.8309	$2.2449 \cdot 10^{-3}$	749	5	0	0
$* - 2.6046 \cdot 10^{-7}$	$2.9245 \cdot 10^{-4}$	854	6	0	0
$* - 4.1293 \cdot 10^{-9}$	$3.6623 \cdot 10^{-5}$	953	7	0	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 2.03 \cdot 10^{-6}$		
-6.8309	$2.7172 \cdot 10^{-3}$	1175	5	0	0
$* - 3.6761 \cdot 10^{-7}$	$3.3120 \cdot 10^{-4}$	1276	6	0	0
$* - 4.3977 \cdot 10^{-9}$	$3.9595 \cdot 10^{-5}$	1581	8	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 2.03 \cdot 10^{-6}$		
-6.8309	$2.5078 \cdot 10^{-3}$	1180	5	0	0
$* - 2.8346 \cdot 10^{-7}$	$8.2760 \cdot 10^{-4}$	1773	7	0	0
$* - 3.3810 \cdot 10^{-8}$	$4.9748 \cdot 10^{-5}$	3098	10	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.07 \cdot 10^{-8}$		
-6.8309	$4.0884 \cdot 10^{-3}$	6645	4	–	–
$* - 7.2135 \cdot 10^{-7}$	$5.4633 \cdot 10^{-4}$	7510	5	–	–
$* - 1.3635 \cdot 10^{-8}$	$7.9169 \cdot 10^{-5}$	8426	6	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 3.05 \cdot 10^{-8}$		
-6.8309	$2.8452 \cdot 10^{-3}$	7402	5	0	0
$* - 3.7124 \cdot 10^{-7}$	$5.0487 \cdot 10^{-4}$	8321	6	0	0
$* - 1.2000 \cdot 10^{-8}$	$9.7543 \cdot 10^{-5}$	9265	7	0	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 2.03 \cdot 10^{-8}$		
-6.8309	$2.0434 \cdot 10^{-3}$	11698	5	0	0
$* - 1.9493 \cdot 10^{-7}$	$7.7171 \cdot 10^{-5}$	14326	7	0	0
$* - 2.3951 \cdot 10^{-10}$	$7.0062 \cdot 10^{-6}$	15142	8	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 2.03 \cdot 10^{-8}$		
-6.8309	$2.1065 \cdot 10^{-3}$	11750	5	0	0
$* - 1.7224 \cdot 10^{-7}$	$8.4784 \cdot 10^{-4}$	16854	7	0	0
$* - 3.3769 \cdot 10^{-8}$	$1.0212 \cdot 10^{-5}$	32117	10	0	0

Table A.14: Evaluation Perfect-AVGASA Sphere Intersected

Name:DEGENLPA			Dim : 20		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 2.26 \cdot 10^{-6}$		
-212.0019	$4.3357 \cdot 10^{-4}$	2720	6	–	–
$* - 4.1220 \cdot 10^{-10}$	$1.5447 \cdot 10^{-5}$	2726	7	–	–
$* - 6.2528 \cdot 10^{-13}$	$4.2746 \cdot 10^{-6}$	2731	8	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 4.47 \cdot 10^{-6}$		
-212.0021	$4.6338 \cdot 10^{-3}$	2692	8	1	0
$* - 4.7063 \cdot 10^{-8}$	$1.6523 \cdot 10^{-4}$	2698	9	1	0
$* - 5.9913 \cdot 10^{-11}$	$5.8903 \cdot 10^{-6}$	2704	10	1	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 3.20 \cdot 10^{-6}$		
-212.0020	$3.6866 \cdot 10^{-3}$	3395	10	1	11
$* - 2.7677 \cdot 10^{-8}$	$9.4336 \cdot 10^{-4}$	3927	13	1	11
$* - 1.8506 \cdot 10^{-9}$	$1.6707 \cdot 10^{-5}$	3958	15	1	11
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 3.20 \cdot 10^{-6}$		
-212.0020	$4.1194 \cdot 10^{-3}$	3395	10	1	11
$* - 3.5241 \cdot 10^{-8}$	$1.5160 \cdot 10^{-5}$	5085	12	1	11
$* - 1.7053 \cdot 10^{-13}$	$1.4991 \cdot 10^{-5}$	5088	13	1	11
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 4.01 \cdot 10^{-8}$		
-212.0016	$4.8098 \cdot 10^{-4}$	44779	6	–	–
$* - 4.8041 \cdot 10^{-10}$	$5.5396 \cdot 10^{-6}$	44822	7	–	–
$* - 1.5007 \cdot 10^{-11}$	$1.3826 \cdot 10^{-5}$	44969	8	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.40 \cdot 10^{-7}$		
-212.0016	$9.6824 \cdot 10^{-3}$	45967	15	3	0
$* - 1.9451 \cdot 10^{-7}$	$1.1152 \cdot 10^{-4}$	46010	16	4	0
$* - 2.6859 \cdot 10^{-11}$	$8.3120 \cdot 10^{-6}$	46057	17	4	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 5.15 \cdot 10^{-8}$		
-212.0043	$8.2233 \cdot 10^{-3}$	32161	10	0	0
$* - 1.3928 \cdot 10^{-7}$	$7.0832 \cdot 10^{-4}$	32204	11	0	0
$* - 1.0243 \cdot 10^{-9}$	$9.4631 \cdot 10^{-5}$	32290	13	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 5.07 \cdot 10^{-8}$		
-212.0043	$4.5519 \cdot 10^{-3}$	39953	11	0	0
$* - 4.3296 \cdot 10^{-8}$	$4.7900 \cdot 10^{-4}$	53655	14	0	0
$* - 4.6830 \cdot 10^{-10}$	$6.2134 \cdot 10^{-5}$	53699	15	0	0

Table A.15: Evaluation Perfect-DEGE Sphere Intersected

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 3.78 \cdot 10^{-6}$		
-1475.6212	$7.8654 \cdot 10^{-3}$	2238	12	–	–
$* - 8.5215 \cdot 10^{-8}$	$6.0911 \cdot 10^{-4}$	2250	16	–	–
$* - 4.9909 \cdot 10^{-10}$	$8.9423 \cdot 10^{-5}$	2259	19	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 2.04 \cdot 10^{-6}$		
-1475.4697	$2.6812 \cdot 10^{-1}$	35879	5000	5	803
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 7.80 \cdot 10^{-6}$		
-1475.6243	$8.3599 \cdot 10^{-3}$	3718	36	0	0
$* - 9.5286 \cdot 10^{-8}$	$7.6196 \cdot 10^{-4}$	3763	51	0	0
$* - 7.9035 \cdot 10^{-10}$	$9.0416 \cdot 10^{-5}$	3802	64	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 4.12 \cdot 10^{-6}$		
-1475.6215	$5.9431 \cdot 10^{-3}$	3828	18	0	0
$* - 1.1461 \cdot 10^{-8}$	$6.8764 \cdot 10^{-4}$	3878	20	0	0
$* - 6.2005 \cdot 10^{-10}$	$7.2118 \cdot 10^{-5}$	3917	33	1	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 3.17 \cdot 10^{-8}$		
-1475.6184	$3.6570 \cdot 10^{-3}$	25826	6	–	–
$* - 4.4547 \cdot 10^{-9}$	$2.5792 \cdot 10^{-4}$	25835	7	–	–
$* - 2.3192 \cdot 10^{-11}$	$1.8196 \cdot 10^{-5}$	25844	8	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 4.25 \cdot 10^{-8}$		
-1475.6184	$1.0576 \cdot 10^{-3}$	25764	8	2	0
$* - 3.7153 \cdot 10^{-10}$	$8.7729 \cdot 10^{-5}$	25772	9	2	0
$* - 5.4570 \cdot 10^{-12}$	$4.6357 \cdot 10^{-5}$	25784	10	2	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.00 \cdot 10^{-7}$		
-1475.6185	$8.5384 \cdot 10^{-3}$	19394	15	0	0
$* - 2.4150 \cdot 10^{-8}$	$8.8917 \cdot 10^{-4}$	19421	18	0	0
$* - 2.5830 \cdot 10^{-10}$	$7.8702 \cdot 10^{-5}$	19453	22	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 6.20 \cdot 10^{-8}$		
-1475.6184	$1.7825 \cdot 10^{-3}$	21486	20	0	0
$* - 8.4628 \cdot 10^{-10}$	$7.9781 \cdot 10^{-4}$	21599	24	0	0
$* - 2.1350 \cdot 10^{-10}$	$2.8404 \cdot 10^{-5}$	21637	30	1	0

Table A.16: Evaluation Perfect-NASH Sphere Intersected

A.3.2 Armijo Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.00 \cdot 10^{-4}$		
-293.4493	$6.6011 \cdot 10^{-1}$	781	5000	$-$	$-$
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.00 \cdot 10^{-4}$		
-293.4492	$9.8686 \cdot 10^{-1}$	791	5000	100	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 3.98 \cdot 10^{-5}$		
-293.4479	$3.2249 \cdot 10^{-3}$	742	100	2	0
$* - 5.8460 \cdot 10^{-9}$	$9.4551 \cdot 10^{-4}$	793	110	2	0
$* - 5.8446 \cdot 10^{-10}$	$7.3924 \cdot 10^{-5}$	885	126	2	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 3.74 \cdot 10^{-5}$		
-293.4478	$9.2773 \cdot 10^{-3}$	766	111	2	0
$* - 1.9288 \cdot 10^{-7}$	$6.8111 \cdot 10^{-4}$	1059	167	2	0
$* - 2.3567 \cdot 10^{-10}$	$7.8797 \cdot 10^{-5}$	1114	174	3	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 1.10 \cdot 10^{-6}$		
-293.4468	$9.9331 \cdot 10^{-3}$	1063	161	$-$	$-$
$* - 8.5131 \cdot 10^{-7}$	$6.1449 \cdot 10^{-4}$	1202	168	$-$	$-$
$* - 3.3176 \cdot 10^{-9}$	$9.1365 \cdot 10^{-5}$	1300	173	$-$	$-$
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.11 \cdot 10^{-6}$		
-293.4468	$9.9691 \cdot 10^{-3}$	1193	170	109	0
$* - 1.8050 \cdot 10^{-7}$	$8.7295 \cdot 10^{-4}$	1563	226	109	0
$* - 1.2443 \cdot 10^{-9}$	$9.3973 \cdot 10^{-5}$	1887	275	109	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 4.41 \cdot 10^{-7}$		
-293.4468	$7.9275 \cdot 10^{-3}$	826	119	4	0
$* - 2.5479 \cdot 10^{-8}$	$9.4192 \cdot 10^{-4}$	904	135	4	0
$* - 4.1535 \cdot 10^{-10}$	$8.1618 \cdot 10^{-5}$	1017	159	4	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 5.11 \cdot 10^{-7}$		
-293.4468	$8.3522 \cdot 10^{-3}$	1122	166	4	0
$* - 4.6076 \cdot 10^{-8}$	$9.5835 \cdot 10^{-4}$	1485	240	4	0
$* - 1.4975 \cdot 10^{-9}$	$9.2954 \cdot 10^{-5}$	1928	326	4	0

Table A.17: Evaluation Armijo-ACCOP30 Sphere intersected

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 2.34 \cdot 10^{-5}$		
-1475.6363	$6.0282 \cdot 10^{-3}$	249	30	–	–
$* - 4.9292 \cdot 10^{-8}$	$8.8814 \cdot 10^{-4}$	273	33	–	–
$* - 1.0898 \cdot 10^{-9}$	$6.9128 \cdot 10^{-5}$	305	37	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 2.33 \cdot 10^{-5}$		
-1475.6362	$9.4906 \cdot 10^{-3}$	237	29	23	0
$* - 1.2399 \cdot 10^{-7}$	$7.3770 \cdot 10^{-4}$	265	33	23	0
$* - 7.5238 \cdot 10^{-10}$	$5.7470 \cdot 10^{-5}$	293	37	23	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 5.00 \cdot 10^{-3}$		
-1479.4233	4.7428	20260	5000	3	22
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 4.99 \cdot 10^{-3}$		
-1479.4240	3.6634	30182	5000	3	22
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 2.18 \cdot 10^{-7}$		
-1475.6185	$6.8741 \cdot 10^{-3}$	209	27	–	–
$* - 1.5714 \cdot 10^{-8}$	$5.7158 \cdot 10^{-4}$	223	28	–	–
$* - 1.0914 \cdot 10^{-10}$	$3.9810 \cdot 10^{-5}$	234	29	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 2.21 \cdot 10^{-7}$		
-1475.6185	$9.9486 \cdot 10^{-3}$	962	154	22	0
$* - 6.2234 \cdot 10^{-8}$	$9.9770 \cdot 10^{-4}$	2084	341	22	0
$* - 4.8180 \cdot 10^{-10}$	$5.3696 \cdot 10^{-5}$	2304	377	22	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 9.44 \cdot 10^{-7}$		
-1475.6191	$9.9321 \cdot 10^{-3}$	6841	1161	0	0
$* - 5.3999 \cdot 10^{-8}$	$9.5161 \cdot 10^{-4}$	9163	1565	0	0
$* - 4.2519 \cdot 10^{-10}$	$3.1270 \cdot 10^{-5}$	9428	1608	1	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 7.34 \cdot 10^{-7}$		
-1475.6189	$9.9549 \cdot 10^{-3}$	10477	1360	0	0
$* - 5.0932 \cdot 10^{-8}$	$8.5190 \cdot 10^{-5}$	14712	1914	1	0
$* - 1.1369 \cdot 10^{-12}$	$8.5170 \cdot 10^{-5}$	14727	1915	1	0

Table A.18: Evaluation Armijo-NASH Sphere Intersected

A.4 Test Data Stiefel manifold

A.4.1 Perfect Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 2.40 \cdot 10^{-6}$		
-141.6424	$3.9339 \cdot 10^{-3}$	4635	15	–	–
$* - 7.4440 \cdot 10^{-9}$	$8.1555 \cdot 10^{-4}$	4641	17	–	–
$* - 9.3371 \cdot 10^{-10}$	$7.4456 \cdot 10^{-5}$	4652	20	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 2.06 \cdot 10^{-5}$		
-141.6410	$8.4468 \cdot 10^{-3}$	5042	57	2	0
$* - 7.8368 \cdot 10^{-8}$	$5.4845 \cdot 10^{-4}$	5060	62	2	0
$* - 1.3290 \cdot 10^{-10}$	$8.4911 \cdot 10^{-5}$	5066	64	2	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.95 \cdot 10^{-6}$		
-141.6426	$8.1509 \cdot 10^{-3}$	4896	16	1	0
$* - 2.4391 \cdot 10^{-8}$	$9.6976 \cdot 10^{-4}$	5044	24	1	0
$* - 7.3214 \cdot 10^{-10}$	$9.9179 \cdot 10^{-5}$	5103	27	1	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 1.16 \cdot 10^{-6}$		
-141.6427	$3.8533 \cdot 10^{-3}$	6494	7	0	0
$* - 7.3944 \cdot 10^{-9}$	$2.7390 \cdot 10^{-5}$	7331	9	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 5.70 \cdot 10^{-8}$		
-141.6429	$7.3661 \cdot 10^{-3}$	59419	21	–	–
$* - 7.3946 \cdot 10^{-8}$	$8.5567 \cdot 10^{-4}$	59484	25	–	–
$* - 9.8441 \cdot 10^{-10}$	$6.5565 \cdot 10^{-5}$	59559	30	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.19 \cdot 10^{-7}$		
-141.6429	$7.8189 \cdot 10^{-3}$	46489	34	2	0
$* - 5.9799 \cdot 10^{-8}$	$7.0475 \cdot 10^{-4}$	46563	39	2	0
$* - 5.7514 \cdot 10^{-10}$	$6.6489 \cdot 10^{-5}$	46638	44	2	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 4.71 \cdot 10^{-8}$		
-141.6429	$9.0785 \cdot 10^{-3}$	76157	16	0	0
$* - 9.2710 \cdot 10^{-8}$	$2.8224 \cdot 10^{-4}$	76567	20	0	0
$* - 8.3787 \cdot 10^{-11}$	$4.0849 \cdot 10^{-5}$	76613	23	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 6.12 \cdot 10^{-8}$		
-141.6429	$2.3221 \cdot 10^{-3}$	77988	25	0	0
$* - 2.3546 \cdot 10^{-9}$	$7.3252 \cdot 10^{-5}$	78252	27	0	0

Table A.19: Evaluation Perfect-ACCOP30 Stiefel

Name:AVGASA			Dim : 8		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 4.00 \cdot 10^{-6}$		
-7.0053	$9.0910 \cdot 10^{-3}$	4835	15	–	–
$* - 9.1683 \cdot 10^{-6}$	$8.0840 \cdot 10^{-4}$	5495	20	–	–
$* - 5.8389 \cdot 10^{-8}$	$6.0436 \cdot 10^{-5}$	6126	25	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 8.34 \cdot 10^{-6}$		
-7.0053	$6.7016 \cdot 10^{-3}$	6416	17	0	0
$* - 4.3952 \cdot 10^{-6}$	$9.1207 \cdot 10^{-4}$	6924	21	0	0
$* - 8.4311 \cdot 10^{-8}$	$7.9283 \cdot 10^{-5}$	7564	26	0	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 4.90 \cdot 10^{-6}$		
-7.0053	$1.3455 \cdot 10^{-3}$	5429	12	0	0
$* - 1.2274 \cdot 10^{-7}$	$5.1827 \cdot 10^{-4}$	5564	13	0	0
$* - 1.3733 \cdot 10^{-8}$	$5.7760 \cdot 10^{-5}$	5669	14	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 4.42 \cdot 10^{-6}$		
-7.0053	$1.8415 \cdot 10^{-3}$	5595	10	0	0
$* - 1.4207 \cdot 10^{-7}$	$9.8218 \cdot 10^{-4}$	6063	11	0	0
$* - 5.6611 \cdot 10^{-8}$	$4.1824 \cdot 10^{-7}$	7063	12	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 4.00 \cdot 10^{-8}$		
-7.0053	$9.3064 \cdot 10^{-3}$	48046	15	–	–
$* - 9.5454 \cdot 10^{-6}$	$8.3207 \cdot 10^{-4}$	54551	20	–	–
$* - 6.1385 \cdot 10^{-8}$	$6.0206 \cdot 10^{-5}$	60765	25	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 8.69 \cdot 10^{-8}$		
-7.0053	$6.8649 \cdot 10^{-3}$	62734	16	0	0
$* - 3.5649 \cdot 10^{-6}$	$6.7194 \cdot 10^{-4}$	67680	20	0	0
$* - 3.9071 \cdot 10^{-8}$	$8.7893 \cdot 10^{-5}$	72749	24	0	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 4.94 \cdot 10^{-8}$		
-7.0053	$8.2977 \cdot 10^{-3}$	51252	11	0	0
$* - 6.6028 \cdot 10^{-6}$	$4.6501 \cdot 10^{-4}$	54840	13	0	0
$* - 9.6950 \cdot 10^{-9}$	$8.1581 \cdot 10^{-5}$	55739	14	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 4.57 \cdot 10^{-8}$		
-7.0053	$1.2961 \cdot 10^{-3}$	62030	11	0	0
$* - 3.7326 \cdot 10^{-8}$	$8.2427 \cdot 10^{-4}$	64822	12	0	0
$* - 5.8411 \cdot 10^{-8}$	$2.6649 \cdot 10^{-5}$	66919	13	0	0

Table A.20: Evaluation Perfect-AVGASA Stiefel

Name:DEGENLPA			Dim : 20		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 7.24 \cdot 10^{-6}$		
-286.0475	$9.2995 \cdot 10^{-3}$	4691	26	–	–
$* - 5.7899 \cdot 10^{-7}$	$6.9894 \cdot 10^{-4}$	4761	33	–	–
$* - 4.3468 \cdot 10^{-9}$	$8.6877 \cdot 10^{-5}$	4822	39	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 2.02 \cdot 10^{-5}$		
-286.0484	$8.5111 \cdot 10^{-3}$	6079	53	1	0
$* - 6.2142 \cdot 10^{-7}$	$8.0893 \cdot 10^{-4}$	6149	60	1	0
$* - 5.5453 \cdot 10^{-9}$	$7.6916 \cdot 10^{-5}$	6219	67	1	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 4.02 \cdot 10^{-6}$		
-286.0470	$2.5207 \cdot 10^{-3}$	4930	14	0	0
$* - 4.6287 \cdot 10^{-8}$	$3.4623 \cdot 10^{-4}$	4961	17	0	0
$* - 7.5278 \cdot 10^{-10}$	$2.2837 \cdot 10^{-5}$	5007	21	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 4.74 \cdot 10^{-6}$		
-286.0470	$8.0378 \cdot 10^{-3}$	5407	17	0	0
$* - 3.2249 \cdot 10^{-7}$	$7.3314 \cdot 10^{-4}$	5443	20	0	0
$* - 1.8868 \cdot 10^{-9}$	$3.3577 \cdot 10^{-5}$	5493	24	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 6.04 \cdot 10^{-8}$		
-286.0466	$9.0805 \cdot 10^{-3}$	39792	24	–	–
$* - 6.5086 \cdot 10^{-7}$	$8.3700 \cdot 10^{-4}$	40364	31	–	–
$* - 5.7963 \cdot 10^{-9}$	$8.0583 \cdot 10^{-5}$	40941	38	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.83 \cdot 10^{-7}$		
-286.0466	$9.4000 \cdot 10^{-3}$	60213	49	1	0
$* - 7.4531 \cdot 10^{-7}$	$9.1687 \cdot 10^{-4}$	60791	56	1	0
$* - 6.5222 \cdot 10^{-9}$	$8.5257 \cdot 10^{-5}$	61364	63	1	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 3.14 \cdot 10^{-8}$		
-286.0466	$6.1020 \cdot 10^{-3}$	39720	11	0	0
$* - 1.5572 \cdot 10^{-7}$	$9.2494 \cdot 10^{-4}$	39954	13	0	0
$* - 5.8578 \cdot 10^{-9}$	$5.9893 \cdot 10^{-5}$	40223	15	0	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 3.50 \cdot 10^{-8}$		
-286.0466	$3.2562 \cdot 10^{-3}$	40937	14	0	0
$* - 5.9680 \cdot 10^{-8}$	$2.8212 \cdot 10^{-4}$	41583	19	0	0
$* - 5.6821 \cdot 10^{-10}$	$4.2019 \cdot 10^{-5}$	41731	20	0	0

Table A.21: Evaluation Perfect-DEGE Stiefel

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 4.34 \cdot 10^{-5}$		
-1501.0188	$9.8633 \cdot 10^{-3}$	8962	1347	–	–
$* - 4.3408 \cdot 10^{-5}$	$9.2862 \cdot 10^{-4}$	12952	2145	–	–
$* - 3.5659 \cdot 10^{-7}$	$9.2905 \cdot 10^{-5}$	16731	2900	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 6.77 \cdot 10^{-6}$		
-1501.0005	$9.8843 \cdot 10^{-3}$	4744	320	1	0
$* - 4.3605 \cdot 10^{-5}$	$9.9764 \cdot 10^{-4}$	8687	1108	1	0
$* - 3.8028 \cdot 10^{-7}$	$9.8851 \cdot 10^{-5}$	12423	1854	1	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 5.99 \cdot 10^{-7}$		
-1501.0004	$7.7674 \cdot 10^{-3}$	3677	4	0	0
$* - 3.9106 \cdot 10^{-8}$	$3.5463 \cdot 10^{-4}$	4161	5	0	0
$* - 1.5111 \cdot 10^{-9}$	$7.5264 \cdot 10^{-5}$	4419	8	0	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 5.99 \cdot 10^{-7}$		
-1501.0004	$7.7674 \cdot 10^{-3}$	3677	4	0	0
$* - 4.0231 \cdot 10^{-8}$	$1.9357 \cdot 10^{-4}$	4362	5	0	0
$* - 5.6457 \cdot 10^{-10}$	$5.3317 \cdot 10^{-5}$	4704	8	0	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 3.80 \cdot 10^{-7}$		
-1501.0001	$9.9736 \cdot 10^{-3}$	67509	1534	–	–
$* - 2.6377 \cdot 10^{-5}$	$9.9614 \cdot 10^{-4}$	90668	2320	–	–
$* - 3.5178 \cdot 10^{-7}$	$9.9561 \cdot 10^{-5}$	113500	3034	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 8.59 \cdot 10^{-8}$		
-1501.0000	$6.5309 \cdot 10^{-3}$	33873	20	2	0
$* - 1.6873 \cdot 10^{-5}$	$9.9496 \cdot 10^{-4}$	54907	719	2	0
$* - 3.3486 \cdot 10^{-7}$	$9.9112 \cdot 10^{-5}$	77413	1414	2	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.47 \cdot 10^{-8}$		
-1501.0000	$9.7055 \cdot 10^{-3}$	41230	8	0	0
$* - 1.3530 \cdot 10^{-7}$	$9.4162 \cdot 10^{-4}$	41363	12	0	0
$* - 8.2673 \cdot 10^{-10}$	$1.3869 \cdot 10^{-5}$	41398	15	1	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 1.46 \cdot 10^{-8}$		
-1501.0000	$9.1382 \cdot 10^{-3}$	42095	7	0	0
$* - 7.5052 \cdot 10^{-8}$	$4.9042 \cdot 10^{-4}$	53919	10	0	0
$* - 7.9194 \cdot 10^{-10}$	$1.9457 \cdot 10^{-5}$	54414	16	1	0

Table A.22: Evaluation Perfect-NASH Stiefel

A.4.2 Armijo Line Search

Name:ACOPP30			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D = 0$	$h = 10^{-3}$		$f_\epsilon = 1.28 \cdot 10^{-5}$		
-141.6428	$3.0991 \cdot 10^{-3}$	266	29	–	–
$* - 3.6143 \cdot 10^{-9}$	$8.1164 \cdot 10^{-4}$	273	30	–	–
$* - 2.6910 \cdot 10^{-10}$	$7.2299 \cdot 10^{-5}$	287	32	–	–
$D = 1$	$h = 10^{-3}$		$f_\epsilon = 1.31 \cdot 10^{-5}$		
-141.6427	$3.0885 \cdot 10^{-3}$	287	32	27	0
$* - 3.5851 \cdot 10^{-9}$	$8.0338 \cdot 10^{-4}$	294	33	27	0
$* - 2.5943 \cdot 10^{-10}$	$5.4494 \cdot 10^{-5}$	308	35	27	0
$D = 3$	$h = 10^{-3}$		$f_\epsilon = 1.13 \cdot 10^{-5}$		
-141.6424	$5.3219 \cdot 10^{-3}$	472	61	3	0
$* - 2.7330 \cdot 10^{-8}$	$8.4894 \cdot 10^{-4}$	526	72	3	0
$* - 6.3824 \cdot 10^{-10}$	$7.3890 \cdot 10^{-5}$	629	96	3	0
$D = 5$	$h = 10^{-3}$		$f_\epsilon = 2.01 \cdot 10^{-5}$		
-296.4327	$4.5139 \cdot 10^{-3}$	1272	141	12	0
$* - 2.6316 \cdot 10^{-8}$	$6.4566 \cdot 10^{-4}$	1407	169	12	0
$* - 5.9640 \cdot 10^{-10}$	$9.8511 \cdot 10^{-5}$	1538	199	12	0
$D = 0$	$h = 10^{-4}$		$f_\epsilon = 1.23 \cdot 10^{-7}$		
-141.6429	$8.7225 \cdot 10^{-5}$	247	33	–	–
$* - 2.5011 \cdot 10^{-12}$	$8.1944 \cdot 10^{-6}$	261	34	–	–
$* - 6.1391 \cdot 10^{-12}$	$3.4664 \cdot 10^{-6}$	275	35	–	–
$D = 1$	$h = 10^{-4}$		$f_\epsilon = 1.04 \cdot 10^{-7}$		
-141.6429	$9.6207 \cdot 10^{-3}$	372	43	20	0
$* - 3.6389 \cdot 10^{-8}$	$9.4412 \cdot 10^{-4}$	512	63	20	0
$* - 3.5118 \cdot 10^{-10}$	$9.2856 \cdot 10^{-5}$	652	83	20	0
$D = 3$	$h = 10^{-4}$		$f_\epsilon = 1.00 \cdot 10^{-7}$		
-141.6429	$9.6235 \cdot 10^{-3}$	696	98	4	0
$* - 1.1782 \cdot 10^{-7}$	$9.6248 \cdot 10^{-4}$	897	150	4	0
$* - 6.9178 \cdot 10^{-10}$	$2.9209 \cdot 10^{-5}$	1040	175	6	0
$D = 5$	$h = 10^{-4}$		$f_\epsilon = 2.16 \cdot 10^{-7}$		
-141.6429	$8.4999 \cdot 10^{-3}$	692	69	4	0
$* - 1.5479 \cdot 10^{2}$	$6.4455 \cdot 10^{-4}$	1599	187	9	0
$* - 4.8607 \cdot 10^{-10}$	$8.3198 \cdot 10^{-5}$	1748	203	10	0

Table A.23: Evaluation Armijo-ACCOP30 Stiefel

Name:NASH			Dim : 72		
L	∇L	f	It	$Rest$	$LogCor$
$D=0$	$h=10^{-3}$		$f_\epsilon = 2.49 \cdot 10^{-3}$		
-1502.8693	1.8744	30085	5000	$-$	$-$
$D=1$	$h=10^{-3}$		$f_\epsilon = 2.49 \cdot 10^{-3}$		
-1502.8693	1.8744	30085	5000	4999	0
$D=3$	$h=10^{-3}$		$f_\epsilon = 2.31 \cdot 10^{-5}$		
-1501.0048	$9.4322 \cdot 10^{-3}$	777	114	0	0
$*-2.2979 \cdot 10^{-6}$	$8.7532 \cdot 10^{-4}$	1634	293	0	0
$*-1.0170 \cdot 10^{-9}$	$6.8275 \cdot 10^{-5}$	1743	329	0	0
$D=5$	$h=10^{-3}$		$f_\epsilon = 1.16 \cdot 10^{-5}$		
-1501.0038	$9.4873 \cdot 10^{-3}$	1161	136	5	0
$*-2.4728 \cdot 10^{-7}$	$3.9793 \cdot 10^{-4}$	1284	153	8	0
$*-2.0893 \cdot 10^{-8}$	$9.3113 \cdot 10^{-5}$	1564	179	13	0
$D=0$	$h=10^{-4}$		$f_\epsilon = 3.20 \cdot 10^{-6}$		
-1501.0023	$9.8623 \cdot 10^{-3}$	8767	1435	$-$	$-$
$*-4.8156 \cdot 10^{-5}$	$9.7290 \cdot 10^{-4}$	21894	2250	$-$	$-$
$*-3.3324 \cdot 10^{-7}$	$9.5491 \cdot 10^{-5}$	33840	2964	$-$	$-$
$D=1$	$h=10^{-4}$		$f_\epsilon = 3.22 \cdot 10^{-6}$		
-1501.0024	$9.8757 \cdot 10^{-3}$	11128	1586	621	0
$*-1.0210 \cdot 10^{-5}$	$9.8641 \cdot 10^{-4}$	17897	2444	621	0
$*-1.0576 \cdot 10^{-7}$	$6.8302 \cdot 10^{-5}$	25321	3385	621	0
$D=3$	$h=10^{-4}$		$f_\epsilon = 2.12 \cdot 10^{-7}$		
-1501.0000	$4.7272 \cdot 10^{-3}$	653	73	6	0
$*-1.7184 \cdot 10^{-6}$	$9.9523 \cdot 10^{-4}$	798	85	7	0
$*-2.5515 \cdot 10^{-7}$	$2.0627 \cdot 10^{-5}$	2130	277	8	0
$D=5$	$h=10^{-4}$		$f_\epsilon = 1.47 \cdot 10^{-7}$		
-1501.0000	$8.5899 \cdot 10^{-3}$	1105	109	7	0
$*-2.9336 \cdot 10^{-8}$	$9.1855 \cdot 10^{-4}$	1353	146	7	0
$*-1.3517 \cdot 10^{-9}$	$7.9720 \cdot 10^{-5}$	1870	241	8	0

Table A.24: Evaluation Armijo-NASH Stiefel

List of Algorithms

List of Figures

List of Tables

Bibliography

[AMS09] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. "Optimization algorithms on matrix manifolds". In: *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[Arm66] Larry Armijo. "Minimization of functions having Lipschitz continuous first partial derivatives". In: *Pacific Journal of mathematics* 16.1 (1966), pp. 1–3.

[Bou14] Nicolas Boumal. *Optimization and estimation on manifolds*. 2014.

[BS21] Martin Buhmann and Dirk Siegel. "Implementing and modifying Broyden class updates for large scale optimization". In: *Computational Optimization and Applications* 78 (2021), pp. 181–203.

[CLO13] David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.

[Dur23] Tobias Durchholz. *My Research Software*. Version 1.0.0. Sept. 2023. URL: `https://github.com/shaiyjan/OptimisationOnManifolds.git`.

[EM12] Jennifer B Erway and Roummel F Marcia. "Limited-memory BFGS systems with diagonal updates". In: *Linear algebra and its applications* 437.1 (2012), pp. 333–344.

[For08a] O. Forster. *Analysis 1*. Vieweg, 2008.

[For08b] O. Forster. *Analysis 2*. Vieweg+Teubner Verlag, 2008.

[Gor21] Claudio Gorodski. "An Introduction to Riemannian Symmetric Spaces". In: *7th School and Workshop on Lie Theory*. 2021, pp. 8–15.

[GOT15] Nicholas IM Gould, Dominique Orban, and Philippe L Toint. "CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization". In: *Computational optimization and applications* 60 (2015), pp. 545–557.

[Ha01] Sung N. Ha. "A nonlinear shooting method for two-point boundary value problems". In: *Computers and Mathematics with Applications* 42.10 (2001), pp. 1411–1420. ISSN: 0898-1221. DOI: `https://doi.org/10.1016/S0898-1221(01)00250-4`. URL: `https://www.sciencedirect.com/science/article/pii/S0898122101002504`.

[Noc80] Jorge Nocedal. “Updating quasi-Newton matrices with limited storage”. In: *Mathematics of computation* 35.151 (1980), pp. 773–782.

[Pen18] Xavier Pennec. “Parallel transport with pole ladder: a third order scheme in affine connection spaces which is exact in affine symmetric spaces”. In: *arXiv preprint arXiv:1805.11436* (2018).

[Per+08] Daniel Perrin et al. *Algebraic geometry: an introduction*. Springer, 2008.

[Pow78] Michael JD Powell. “Algorithms for nonlinear constraints that use Lagrangian functions”. In: *Mathematical programming* 14 (1978), pp. 224–248.

[RS22] Joel W Robbin and Dietmar A Salamon. “Introduction to differential geometry”. In: *ETH, Lecture Notes, preliminary version* (2022).

[Sha13] Igor R. Shafarevich. *Basic Algebraic Geometry 1: Varieties in Projective Space*. 2013.

[Tow18] James Townsend. *Differentiating the qr decomposition*. 2018.

[Wol69] Philip Wolfe. “Convergence conditions for ascent methods”. In: *SIAM review* 11.2 (1969), pp. 226–235.

Declaration of Authorship

I declare that I have completed this dissertation single-handedly without the unauthorized help of a second party and only with the assistance acknowledged therein. I have appropriately acknowledged and cited all text passages that are derived verbatim from or are based on the content of published work of others, and all information relating to verbal communications. I consent to the use of an anti-plagiarism software to check my thesis. I have abided by the principles of good scientific conduct laid down in the charter of the Justus Liebig University Giessen „Satzung der Justus-Liebig-Universität Gießen zur Sicherung guter wissenschaftlicher Praxis“ in carrying out the investigations described in the dissertation.

Eigenständigkeitserklärung

Ich erkläre: Ich habe die vorgelegte Dissertation selbstständig und ohne unerlaubte fremde Hilfe und nur mit den Hilfen angefertigt, die ich in der Dissertation angegeben habe. Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen sind, und alle Angaben, die auf mündlichen Auskünften beruhen, sind als solche kenntlich gemacht. Ich stimme einer evtl. Überprüfung meiner Dissertation durch eine Antiplagiat-Software zu. Bei den von mir durchgeführten und in der Dissertation erwähnten Untersuchungen habe ich die Grundsätze guter wissenschaftlicher Praxis, wie sie in der „Satzung der Justus-Liebig-Universität Gießen zur Sicherung guter wissenschaftlicher Praxis“ niedergelegt sind, eingehalten.
Ich stimme zu, dass die vorliegende Arbeit mit einer Anti-Plagiatssoftware überprüft werden darf.

Gießen, den 11.09.2023

Tobias Durchholz