NTNU

Kunnskap for en bedre verden

Department of Mathematical Sciences

TMA4500 - Industrial Mathematics, Specialization Project

Riemannian Optimization using second order Information on the Symplectic Stiefel manifold

Author:
Ole Gunnar Røsholt Hovland

23rd December 2024

Abstract

This report explores optimization algorithms on Riemannian manifolds, which are problems where the cost function is defined on a smooth manifold. Riemannian optimization is used in many problems, for example in machine learning, robotics and scientific computing. The report reproduces some results from Jensen and Zimmermann [12] and Bendokat and Zimmermann [3]. In particular we investigate the Riemannian gradient descent method and the Riemannian trust-region method. To compare the algorithms we apply them to multiple optimization problems of symplectic nature, as well as evaluating the impact of different retractions on the algorithms performance. All of the code have been implemented in Julia using the package Manopt.jl [4]. Apart from mostly reaffirming the results in the aforementioned papers, we also find that the pseudo-Riemannian geodesic retraction, introduced by Bendokat and Zimmermann, performs competitively compared to the Cayley retraction.

Table of Contents

1 Introduction

In this report we will investigate algorithms that perform optimization on Riemannian manifolds. As noted in the preface of [6], because of the inherent structure of some problems, they can be described as problems on some smooth manifold. This could be because of the problem's natural geometry, latent data simplicity, symmetry and/or positivity. Formulating problems through the lens of smooth manifolds has proven successful in the fields of machine learning, computer vision and robotics, signal processing, and scientific computing [6]. Optimization on manifolds is a field that began around fifty years ago, and has in resent years garnered more attention [6]. Optimization on Riemannian manifolds, or simply Riemannian optimization, is a field of optimization where, in an effort to optimize some cost function, one restricts the search space to a Riemannian manifold $\mathcal{M}$. Specifically, instead of defining the problem in terms of restrictions on a Euclidean space, one defines the problem on an appropriate Riemannian manifold. The result of this is that we are, in a sense, encoding the restrictions into inherent properties of the search space. To this end one defines the cost function to be a smooth function inherently defined on the manifold, $f\colon \mathcal{M} \to \mathbb{R}$. This results in the following, now unconstrained, optimization problem:

$$\min_{p \in \mathcal{M}} f(p). \tag{1.1}$$

To solve this problem we employ optimization algorithms adapted to work inherently on the Riemannian manifold. Many of these algorithms can, among other things, exploit the information contained in the smoothness in the cost function. An algorithm is called a first order algorithm if it only utilizes the gradient of the cost function, or of second order if it also leverages the Hessian, as a means to find the critical point of (1.1) more efficiently. Despite this, in applications, the computational cost of computing the gradient and Hessian can be expensive. This means that, for algorithms using the gradient or Hessian in every iteration, the time to compute each iteration can be higher than that of an algorithm not relying on smoothness information. However, as each iteration could be a better guess, in the sense that the iteration moves us closer to the critical point, the total number of iterations are usually lower for higher order algorithms [6, p. 119]. For some problems the number of iterations is so low as to compensate for the higher computational cost per iteration, while for other problems the less accurate but faster lower order algorithms are the optimal choice.

The primary goal of this project report is to reproduce some of the findings, programmed in the programming language MATLAB, of Jensen and Zimmermann [12] while programming the experiments in the programming language Julia. To this end we investigate the performance of the first order algorithm "gradient descent" and the second order algorithm called the "trust-region method". We apply these algorithms to problems defined on the symplectic Stiefel manifold, where we solve the problem of finding the nearest symplectic matrix to a given non-symplectic matrix. We also investigate the symplectic decomposition problem. The secondary goal of this report is to investigate the performance of different retractions on algorithm performance, investigated by Bendokat and Zimmermann [3]. For completeness we provide a partial summary of the theory outlined in both of these papers. Regarding original works, some proofs have been worked out independently by the author of this report. Also, regarding the retraction called pseudo-Riemannian geodesic, investigated in [3], we investigate it further than what was done in the original paper. It will be stated explicitly when something is original work.

The structure of the report is as follows: In Section 2, foundational definitions are described, as well as necessary introductions to the symplectic group and the symplectic Stiefel manifold. The foundational definitions is the only part of the report formatted as a reference work, which means that this section consists of disjoint definitions coining the notation used in the report, as well as serving as a refresher to the underlying theory of the rest of the report. Section 3 outlines the theory which is the main contribution of [3], namely arriving at explicit geodesic equations through defining an appropriate right-invariant metric. We then use these results to define the Riemannian gradient and Hessian before finishing the section with some particular results bridging some gaps from the theory to application. After the theory has been outlined, Section 4 introduces the Riemannian gradient descent and the Riemannian trust region method. Lastly the section comments on how the algorithms for the experiments were implemented. Section 5 contains the description and results of the experiments retraction comparison through the nearest symplectic matrix problem, comparing the performance of the Riemannian gradient descent- and Riemannian trust-region algorithm through the nearest symplectic matrix problem, and finally comparing the aforementioned algorithms through the problem of performing a proper symplectic decomposition. Lastly Section 6 contains the discussion and conclusion of the report, where we summarize our findings, discuss the results, and suggest directions of further work.

This project report was written in LaTeX, based on the template made by Jon Arnt Kårstad [13].

2 Introductory Theory

In this section we will first cover some foundational definitions. We will then proceed with defining the symplectic group, and lastly the symplectic Stiefel manifold. Though the goal of this report is to perform experiments on the symplectic Stiefel manifold, this manifold is interlinked with the symplectic group. Namely it is a special kind of submanifold of the symplectic group called a quotient manifold. After defining necessary concepts on the symplectic group, we will leverage this property to define a right-invariant framework on the symplectic Stiefel manifold. This will enable us to define geodesics, the Riemannian gradient, and ultimately the Riemannian Hessian on the Symplectic Stiefel manifold.

The theory presented in this report is based on the works of Bendokat and Zimmermann [3], and Jensen and Zimmermann [12], with the goal of reproducing some of their findings. Other references will be mentioned as needed, and we will clearly indicate when a section is based on original work.

2.1 Foundational definitions

This section is designed to be a reference work to set notation, and to make the rest of the report more readable in the sense that we do not take detours to define basic properties. Note that even though the elements of the symplectic group and symplectic stiefel manifold are matrices, the word "point" will be used to refer to a specific matrix, as the specific matrix is a point on the matrix manifold.

Definition 1 (Orthogonal group)**.** *The* real Orthogonal group *is defined as the set of all orthogonal matrices in $\mathbb{R}^{n\times n}$, denoted by*

$$O(n) := \{p \in \mathbb{R}^{n\times n} \mid p^{\mathrm{T}}p = I_k\},$$

where $I_n \in \mathbb{R}^{n\times n}$ is the identity. [6, p. 10]

Definition 2 (Stiefel manifold)**.** *As in defined in [6, p. 9], the* Stiefel manifold *is the set of all matrices in $\mathbb{R}^{n\times k}$ with k orthonormal columns, denoted by*

$$\mathrm{St}(n,k) = \left\{p \in \mathbb{R}^{n\times k} \;\middle|\; p^{\mathrm{T}}p = I_k\right\}.$$

Definition 3 (Quotient manifold)**.** *We define the quotient manifold as in [1, p. 27]. Let $\mathcal{M}$ be a manifold,as defined in [1, p. 20], equipped with the operation $\sim$ called the* equivalence relation. *The equivalence relation has the following properties:*

1. *(reflexive) $p \sim p$ for all $p \in \mathcal{M}$,*
2. *(symmetric) $p \sim q$ if and only if $q \sim p$ for all $q, p \in \mathcal{M}$, and*
3. *(transitive) given $p \sim q$ and $q \sim r$ this implies that $p \sim r$ for all $p, q, r \in \mathcal{M}$.*

Given the set $[p] := \{q \in \mathcal{M} \colon q \sim p\}$ called the equivalence class *of all points equivalent to p, the set*

$$\mathcal{M}/\sim \; := \{[p] \mid p \in \mathcal{M}\}$$

is called the quotient of $\mathcal{M}$ by $\sim$. *It is the set of all equivalence classes of $\sim$ in $\mathcal{M}$. The mapping $\pi \colon \mathcal{M} \to \mathcal{M}/\sim,\ p \mapsto \pi(p) = [p]$ is called the* natural- *or* canonical projection. *Since $\pi(x) = \pi(y)$ if and only if $x \sim y$, $[p] = \pi^{-1}(\pi(p))$.*

Definition 4 (Tangent Space)**.** *Following [6, Def. 8.33], for a point p on a smooth manifold $\mathcal{M}$, defined as in [15, p. 13], denote the set of smooth curves [6, Def. 8.5] passing through p at $t = 0$ as C_p. This means that $\alpha(0) = p$ for all $\alpha \in C_p$. We say that to curves $\alpha, \beta \in C_p$ are equivalent if*

$$(\phi \circ \alpha)'(0) = (\phi \circ \beta)'(0),$$

meaning their derivatives match in a coordinate chart *(defined as in [15, p. 4]) if their derivatives in the coordinate chart at zero are equal. Denote this equivalence relation as $\alpha \sim \beta$. It has analogous properties to the equivalence relation in Definition 3. The equivalence class is defined as $[\alpha] = \{\beta \in C_p \mid \alpha \sim \beta\}$. Every equivalence class is called a* tangent vector *to $\mathcal{M}$ at p. The* tangent space *at p is the quotient set*

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

We denote the tangent bundle *as $T\mathcal{M} = \bigsqcup_{p\in\mathcal{M}} T_p\mathcal{M}$, where $\bigsqcup$ denotes the disjoint union.*

Definition 5 (Vector field on a smooth manifold)**.** *Following the Appendix of [17, A.3], a vector field $\mathcal{X} \colon \mathcal{M} \to T\mathcal{M},\ p \mapsto \mathcal{X}(p) \in T_p\mathcal{M}$ assigns a vector to every point on a smooth manifold $\mathcal{M}$, as defined in [15, p. 13]. We define the vectors point-wise for every p in a chart $(U, p^1, \dots, p^n) \subseteq \mathcal{M}$, as defined in [17, A.1]. Since the coordinate vectors $\partial_i \coloneqq \partial/\partial x^i|_p$ form a basis of $T_p\mathcal{M}$, the vector at a point p is defined as*

$$\mathcal{X}(p) = \sum_{i=1}^{n} \alpha_i \partial_i \eqqcolon \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\partial},$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$. We assume that the vector field is smooth, as defined in [17, A.3].

Definition 6 (Riemannian manifold)**.** *As defined in [5, def 2.6, p. 179]: a smooth manifold $\mathcal{M}$ is a* Riemannian manifold *if we can define a field of symmetric, positive definite, bilinear forms g, called the* Riemannian metric*. By field we mean that g_p is defined on the tangent space $T_p\mathcal{M}$ at each point $p \in \mathcal{M}$ [5, def 2.1, p. 178]. We will assume that g is smooth, as defined in [5, Def. 3.1, p 65].*

Definition 7 (Whitney sum)**.** *Restricting the definition in [7, p.114] to our scope, for a manifold $\mathcal{M}$ define subsets of the tangent bundle (see Definition 4) $\mathcal{X}, \mathcal{Y} \subseteq T\mathcal{M}$. The Whitney sum is defined as*

$$\mathcal{X} \oplus \mathcal{Y} \coloneqq \bigsqcup_{p\in\mathcal{M}} \mathcal{X}_p \times \mathcal{Y}_p,$$

where $\bigsqcup$ denotes the disjoint union.

Definition 8 (Horizontal & Vertical Space)**.** *Using Definition 3, given a Riemannian manifold $\overline{\mathcal{M}}$ with Riemannian metric $\overline{g}$, denote a quotient manifold of $\overline{\mathcal{M}}$ as $\mathcal{M} = \overline{\mathcal{M}}/\sim$. Following the definitions in Absil et al. [1, p. 43], for a point $p \in \mathcal{M}$, the equivalence class $[p] = \pi^{-1}(p)$ induces an embedded submanifold of $\overline{\mathcal{M}}$ (see Definition 3), hence it admits a tangent space,*

$$\mathcal{V}_{\overline{p}} = T_{\overline{p}}(\pi^{-1}(p)) = \ker(\mathrm{D}\,\pi(p))$$

[12, p. 4] named the vertical space *at $\overline{p}$. Canonically chosen as the orthogonal complement of $\mathcal{V}_{\overline{p}}$ in $T_{\overline{p}}\overline{\mathcal{M}}$, the* horizontal space *[1, p. 48] is defined as*

$$\mathcal{H}_{\overline{p}} \coloneqq \mathcal{V}_{\overline{p}}^{\perp} = \{Y_{\overline{p}} \in T_{\overline{p}}\overline{\mathcal{M}} \mid \overline{g}(Y_{\overline{p}}, Z_{\overline{p}}) = 0 \quad \forall \quad Z_{\overline{p}} \in \mathcal{V}_{\overline{p}}\}.$$

The horizontal lift *at $\overline{p} \in \pi^{-1}(p)$ of a tangent vector $X_p \in T_p\mathcal{M}$ is the unique tangent vector $X_{\overline{p}} \in \mathcal{H}_{\overline{p}}$ that satisfies $\mathrm{D}\pi(\overline{p})[X_{\overline{p}}] = X_p$. Note that given $\mathcal{H}_{\overline{p}}, \mathcal{V}_{\overline{p}} \in T_{\overline{p}}\overline{\mathcal{M}}$, $\mathcal{H}_{\overline{p}} \oplus \mathcal{V}_{\overline{p}} = T_{\overline{p}}\overline{\mathcal{M}}$, where $\oplus$ denotes the Whitney sum as in Definition 7.*

Definition 9 (Riemannian connection)**.** *The* Riemanian connection*, also known as the* Levi-Civita connection*, is the unique affine connection which is torsion free, and metric compatible [17, Def. 6.4]. In Appendix A of [12], denoting $\mathfrak{X}(\mathcal{M})$ as the space of smooth vector fields on $\mathcal{M}$, it is defined as the unique $\mathbb{R}$-bilinear smooth map on $\mathcal{M}$ with Riemannian metric $\langle\cdot,\cdot\rangle_p$*

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

such that the following properties hold. Given $X, Y, Z \in \mathfrak{X}(\mathcal{M})$, and $f \in \mathcal{C}^\infty(M)$, $\nabla_X Y$has the following properties:

1. *(first argument linearity)* $\nabla_{fX}Y = f\nabla_X Y$,
2. *(Leibnitz)* $\nabla_X(fY) = (Xf)Y + f\nabla_X Y$,
3. *(torsion free)* $\nabla_X Y - \nabla_Y X = [X, Y]$, *where* $[\cdot,\cdot]$ *is the Lie bracket [15, p. 186], and*
4. *(metric compatibility)* $Z\langle X, Y\rangle = \langle \nabla_Z X, Y\rangle + \langle X, \nabla_Z Y\rangle$.

Definition 10 (Geodesics)**.** *To be able to define what it means to travel a (local) path of minimal length on any manifold, we need to define geodesics. They are the generalization of straight lines in Euclidean space, and since most optimization algorithms travel along paths, they are essential in transferring Euclidean optimization algorithms to the Riemannian domain. Defining geodesics rigorously as in [17, Def. 14.1], assume we have a Riemannian manifold* $\mathcal{M}$ *equipped with an affine connection* ∇ *(as in Definition 9). A curve* $c\colon I \to \mathcal{M}$*, where* I *is an open, closed or half-open interval of* $\mathbb{R}$*, is a* geodesic *if the covariant derivative (defined in [12, Appendix A]) of* $c'(t)$ *is zero for all* t*.*

Definition 11 (Riemannian gradient)**.** *As defined in [6, Def. 3.58], for the Riemannian manifold* $\mathcal{M}$*, let* $f\colon \mathcal{M} \to \mathbb{R}$ *be a smooth function. The* Riemannian gradient *of* f *is the vector field* $\operatorname{grad} f$ *on* $\mathcal{M}$ *defined uniquely by for all* $(p, X) \in T\mathcal{M}$*,*

$$\mathrm{D}\, f(p)[X] = \langle X, \operatorname{grad} f(p)\rangle_p,$$

where D *denotes the differential of* f *at* p *meaning* $(f \circ c)'(0)$ *for som curve* $c(t)$ *like in Definition 4 [6, Def. 3.34]. The* $\langle\cdot,\cdot\rangle_p$ *denotes the Riemannian metric at the point* p*.*

Definition 12 (Christoffel symbols)**.** *The method we will employ to completely describe a connection (as defined in Definition 9) locally is to describe them through* Christoffel symbols*. Following the definition in [17, p. 100], let* ∇ *be an affine connection on* $\mathcal{M}$*. Denote a coordinate vector field on the coordinate open set* $(U, p^1, \dots, p^n) \subseteq \mathcal{M}$ *by* $\partial_i := \partial/\partial p^i$ *as in [17, Appendix A.3]. In this coordinate frame there exist the numbers called Christoffel symbols,* Γ_{ij}^k*, defined through the following*

$$\nabla_{\partial_i}\partial_j = \sum_{k=1}^{n} \Gamma_{ij}^k \partial_k =: \mathbf{\Gamma}_{ij}^{\mathrm{T}} \boldsymbol{\partial}.$$

Definition 13 (Retraction)**.** *Following [6, Def. 3.47], a retraction on a smooth manifold* $\mathcal{M}$ *is a smooth map,*

$$\mathcal{R}\colon T\mathcal{M} \to \mathcal{M}, \quad (p, X) \mapsto \mathcal{R}_p(X)$$

such that every curve generated from $c(t) = \mathcal{R}_p(tX)$ *satisfies* $c(0) = p$ *and* $\dot{c}(0) = X$*. Equivalently the conditions can be stated as in [6, p. 40] without the use of curves. For all* $p \in \mathcal{M}$*,* $\mathcal{R}_p(0) = p$*, and* $\mathrm{D}\,\mathcal{R}_p(0)\colon T_p\mathcal{M} \to T_p\mathcal{M}$*,* $\mathrm{D}\,\mathcal{R}_p(0)[X] = X$ *is the identify map.*

Now that we have defined som foundational definitions to this report, we will proceed with defining the symplectic group and the symplectic Stiefel manifold, the objects of interest in this report. For the rest of this project report we denote $\mathcal{M}$ as being a Riemannian manifold unless stated otherwise.

2.2 The Real Symplectic group

The *real symplectic group* is a space containing the symplectic Stiefel manifold. In this section we will therefore define the necessary concepts for this group in preparation for the symplectic Stiefel manifold. Following [3, p. 3], to be able to define the symplectic group we first need some preliminary definitions. Define the *symplectic identity* as the following block matrix,

$$J_{2n} := \begin{bmatrix} 0_n & I_n \\ -I_n & 0_n \end{bmatrix},$$

where I_n denotes the $n \times n$ identity matrix. Note that throughout this report, if it is clear from the context, the subscript for I and J will be omitted for the sake of notational clarity. J_{2n} has some properties we will take advantage of frequently:

$$J_{2n}^{\mathrm{T}} = -J_{2n} = J_{2n}^{-1}. \tag{2.1}$$

The symplectic group is defined as the set of matrices which satisfy the symplectic condition in the following sense. We define the real symplectic group as

$$\mathrm{Sp}(2n) \coloneqq \{p \in \mathbb{R}^{2n\times 2n} \mid p^{+}p = I_{2n}\}, \tag{2.2}$$

where $\cdot^{+}$ is defined as the *symplectic inverse* of any matrix $q \in \mathbb{R}^{2n\times 2k}$ such that

$$q^{+} \coloneqq J_{2k}^{\mathrm{T}} q^{\mathrm{T}} J_{2n}. \tag{2.3}$$

Finding the tangent space of $\mathrm{Sp}(2n)$ follows intuitively from the definition, which we will verify ourselves in the following derivation. We begin by defining the tangent space at the identity. Define the curve $c_I(t) \in \mathrm{Sp}(2n)$, such that $c_I(0) = I$ and $\dot{c}_I(0) \coloneqq \frac{d}{dt}c_I(t)|_{t=0} = X$. Since $c_I(t)$ is a curve on $\mathrm{Sp}(2n)$, by (2.2) it must satisfy the following condition:

$$c_I(t)^{\mathrm{T}} J_{2n} c_I(t) = J_{2n}. \tag{2.4}$$

Taking the derivative of (2.4) with respect to t at $t = 0$, (2.4) becomes

$$\dot{c}_I(t)^{\mathrm{T}} J c_I(0) + c_I(0)^{\mathrm{T}} J \dot{c}_I(0) = 0. \tag{2.5}$$

Recalling that $c_I(0) = I$, defining $\Omega \coloneqq \dot{c}_I(0)$, and leveraging the symplectic properties (2.1), (2.5) becomes

$$\Omega^{+} = -\Omega.$$

We notice that this definition of the tangent space at the identity is equivalent to the definition of the Lie algebra of $\mathrm{Sp}(2n)$ [3, p. 3]. It is given by

$$\mathfrak{sp}(2n) \coloneqq T_I\mathrm{Sp}(2n) = \{\Omega \in \mathbb{R}^{2n\times 2n} \mid \Omega^{+} = -\Omega\}, \tag{2.6}$$

where Ω is called the Hamiltonian matrix.

Now, for a general $p \in \mathrm{Sp}(2n)$, $c(t) = pc_I(t) \in \mathrm{Sp}(2n)$, where $c(0) = p$. Therefore, by translation, we can define the tangent space of $\mathrm{Sp}(2n)$ at any point p as

$$\begin{aligned} T_p\mathrm{Sp}(2n) &= \{\Omega p \in \mathbb{R}^{2n\times 2n} \mid \Omega \in \mathfrak{sp}(2n)\}, \\ &= \{p\Omega \in \mathbb{R}^{2n\times 2n} \mid \Omega \in \mathfrak{sp}(2n)\}. \end{aligned} \tag{2.7}$$

We will verify that this makes sense by inserting $c \coloneqq c(t)$ in the condition of (2.2) to get

$$c^{\mathrm{T}} p^{\mathrm{T}} J p c = J.$$

We again take the derivative of to get

$$\Omega^{\mathrm{T}} p^{\mathrm{T}} J p + p^{\mathrm{T}} J p \Omega = 0, \tag{2.8}$$

where we used that $c(0) = I$, and the definition of Ω. Now, multiplying (2.8) by $J^{\mathrm{T}}p$ from the left, and simplifying we again end up with $\Omega^{+} = -\Omega$, verifying our claim.

Now that we have defined the symplectic group, and its components that we will use throughout this report, we will move on to defining the symplectic Stiefel manifold.

2.3 The Symplectic Stiefel manifold

Having defined the real symplectic group, we introduce the manifold of interest, the real symplectic Stiefel manifold. It is defined as

$$\mathrm{SpSt}(2n, 2k) \coloneqq \{p \in \mathbb{R}^{2n\times 2k} \mid p^{+}p = I_{2k}\}, \tag{2.9}$$

where p^{+} is defined as in (2.3). Following [3, Prop. 3.1], it is explicitly connected to the symplectic group in the sense that $\mathrm{SpSt}(2n, 2k)$ is diffeomorphic to the following quotient manifold of $\mathrm{Sp}(2n)$:

$$\mathrm{SpSt}(2n, 2k) \cong \mathrm{Sp}(2n)/\mathrm{Sp}(2(n-k)),$$

where the notion of a quotient manifold is defined as in Definition 3. It has dimension $\dim(\mathrm{SpSt}(2n,2k)) = (2n-2k+1)k$. The following piece of insight regarding the Stiefel manifold, defined in Definition 2, can give some further intuition on what the symplectic Stiefel manifold is. We note that the Stiefel manifold is a quotient space, as defined in Definition 3, of the orthogonal group, as defined in Definition 1, such that $\mathrm{St}(2n,2k) = \mathrm{O}(n)/\mathrm{O}(n-k)$.

In a similar manner as for the symplectic group, the expression for the tangent space follows straightforwardly from the definition of $\mathrm{SpSt}(2n,2k)$. Assume we have a curve, $c(t) \in \mathrm{SpSt}(2n,2k)$, such that $c(0) = p$ and $\dot{c}(0) := \frac{d}{dt}c(t)|_{t=0} = X$. Since $c(t)$ is a curve in $\mathrm{SpSt}(2n,2k)$, by (2.9) it must satisfy the following condition:

$$c(t)^{\mathrm{T}} J_{2n} c(t) = J_{2k}. \tag{2.10}$$

Taking the derivative of (2.10) with respect to t at $t=0$ we get

$$\dot{c}^{\mathrm{T}}(t) J_{2n} c(t) + c^{\mathrm{T}}(t) J_{2n} \dot{c}\big|_{t=0} = X^{\mathrm{T}} J_{2n} p + p^{\mathrm{T}} J_{2n} X = 0_{2k}.$$

After moving the first term over to the left hand side, and multiplying with J_{2n} from the left, the expression simplifies to

$$p^{+} X = -X^{+} p.$$

We recognize this condition as $p^{+}X \in \mathfrak{sp}(2k)$ as defined in (2.6). This means that for a point p,

$$T_p\mathrm{SpSt}(2n,2k) = \left\{ X \in \mathbb{R}^{2n\times 2k} \;\middle|\; p^{+}X \in \mathfrak{sp}(2k) \right\}. \tag{2.11}$$

Now, after defining the symplectic Stiefel manifold and its tangent space, we will proceed to define the right-invariant metric on $\mathrm{SpSt}(2n,2k)$ in preparation for defining an explicit expression of the geodesics on the manifold.

3 Right-Invariant Framework on the Symplectic Stiefel

One of the key insights of Bendokat and Zimmermann [3, p. 11] is that using a right invariant framework one is able to construct geodesics on $\mathrm{SpSt}(2n,2k)$. Following in their footsteps, we will first define a right-invariant metric on $\mathrm{Sp}(2n)$ and its corresponding geodesics, then transport this metric to $\mathrm{SpSt}(2n,2k)$. This will allow us to define geodesics, the Riemannian gradient, and other tools to define the Riemannian gradient descent algorithm on $\mathrm{SpSt}(2n,2k)$. Finally, we will follow Jensen and Zimmermann [12, p. 10] who, through this right-invariant framework, defines the Riemannian Hessian, and other tools necessary for the implementation of the trust-region method on the $\mathrm{SpSt}(2n,2k)$.

3.1 Right-invariant metric

In this section we will define a right-invariant metric on $\mathrm{Sp}(2n)$, which will allow us to define a right-invariant metric on $\mathrm{SpSt}(2n,2k)$. We begin by defining the point-wise right-invariant metric on $\mathrm{Sp}(2n)$ as the mapping

$$\begin{aligned} g_p^{\mathrm{Sp}}&\colon T_p\mathrm{Sp}(2n) \times T_p\mathrm{Sp}(2n) \to \mathbb{R}, \\ g_p^{\mathrm{Sp}}(X_1,X_2) &:= \frac{1}{2}\operatorname{tr}((X_1p^{+})^T X_2 p^{+}), \quad X_1, X_2 \in T_p\mathrm{Sp}(2n). \end{aligned} \tag{3.1}$$

It is right-invariant in the sense that $g_{pq}^{\mathrm{Sp}}(X_1q, X_2q) = \frac{1}{2}\operatorname{tr}((X_1qq^{+}p^{+})^T X_2qq^{+}p^{+}) = g_p^{\mathrm{Sp}}(X_1,X_2)$ for all $p \in \mathrm{Sp}(2n)$. Now that we have defined g_p^{Sp}, we can define a metric $\mathrm{SpSt}(2n,2k)$ in a way that preserves the right-invariance. To achieve this we will use a *horizontal lift* to define a metric on $\mathrm{SpSt}(2n,2k)$ through (3.1). Split $T_p\mathrm{Sp}(2n)$ into to parts: the horizontal- and vertical part, with respect to g_p^{Sp} and π:

$$T_p\mathrm{Sp}(2n) = \mathrm{Ver}_p\,\mathrm{Sp}(2n) \oplus \mathrm{Hor}_p\,\mathrm{Sp}(2n). \tag{3.2}$$

Noting that $\mathfrak{sp}(2n) = T_I\mathrm{Sp}(2n)$, we can express these spaces through $\mathfrak{sp}(2n)$ as

$$\begin{aligned}\mathrm{Ver}_p\,\mathrm{Sp}(2n) &\coloneqq \{\Omega p \mid \Omega \in \mathrm{Ver}\,\mathfrak{sp}(2n)\}\,,\\ \mathrm{Hor}_p\,\mathrm{Sp}(2n) &\coloneqq \{\Omega p \mid \Omega \in \mathrm{Hor}\,\mathfrak{sp}(2n)\}\,,\end{aligned}$$

where horizontal- and vertical space is defined as in Definition 8. Explicit expressions of $\mathrm{Ver}\,\mathfrak{sp}(2n)$ and $\mathrm{Hor}\,\mathfrak{sp}(2n)$ is given in [3, p. 11]. We will only need the horizontal space to define our desired metric, so we will only explicitly state $\mathrm{Hor}\,\mathrm{Sp}(2n)$. It is expressed as

$$\mathrm{Hor}_p\,\mathrm{Sp}(2n) = \left\{\overline{\Omega}p \mid \Omega r + r\Omega - r\Omega r = \overline{\Omega} \in \mathrm{Hor}\,\mathfrak{sp}(2n)\right\},$$

where $r = J_{2n}^{\mathrm{T}}\pi(p)\pi(p)^{+}J_{2n}$. From this quotient perspective, for $p = \pi(q)$ where $q \in \mathrm{Sp}(2n)$, we can express vectors from $T_p\mathrm{SpSt}(2n, 2k)$ through $\mathrm{Hor}_p\,\mathrm{Sp}(2n)$. Following [12, p. 5] any $X \in T_p\mathrm{SpSt}(2n, 2k)$ and a specific $p = \pi(q)$, there exists a unique horizontal lift (see Definition 8)

$$\mathfrak{h}_q(X) = \overline{\Omega}(X)q, \tag{3.3}$$

where

$$\overline{\Omega}(X) = X(p^{\mathrm{T}}p)^{-1}p^{\mathrm{T}} + J_{2n}p(p^{\mathrm{T}}p)^{-1}X^{\mathrm{T}}\big(I_{2n} - J_{2n}^{\mathrm{T}}p(p^{\mathrm{T}}p)^{-1}p^{\mathrm{T}}J_{2n}\big)J_{2n}. \tag{3.4}$$

By [14, Thm. 2.28], we see that $\mathrm{SpSt}(2n, 2k)$ has a unique smooth manifold structure and a unique Riemannian metric such that π is a Riemannian submersion. The point-wise right-invariant Riemannian metric on $\mathrm{SpSt}(2n, 2k)$ can then be defined using (3.3) as the mapping

$$\begin{aligned}g_p\colon T_p\mathrm{SpSt}(2n, 2k) \times T_p\mathrm{SpSt}(2n, 2k) &\to \mathbb{R},\\ g_p(X_1, X_2) &\coloneqq g_p^{\mathrm{Sp}}(\mathfrak{h}_q(X_1), \mathfrak{h}_q(X_2)).\end{aligned}$$

Expressing the mapping explicitly, we get the following expression for the right-invariant metric on $\mathrm{SpSt}(2n, 2k)$:

$$g_p(X_1, X_2) = \mathrm{tr}\left(X_1^T\left(I_{2n} - \frac{1}{2}J_{2n}^T p(p^Tp)^{-1}p^TJ_{2n}\right)X_2(p^Tp)^{-1}\right), \tag{3.5}$$

for $X_1, X_2 \in T_p\mathrm{SpSt}(2n, 2k)$.

In the next section we will define an explicit expression for the geodesics on $\mathrm{SpSt}(2n, 2k)$ using the right-invariant metric we defined in this section.

3.2 Geodesics

In this section we will begin by defining geodesics, generally defined in Definition 10, on $\mathrm{Sp}(2n)$ using the right-invariant metric (3.1). Through the lens of quotient manifolds, and our right invariant framework, we will define geodesics on $\mathrm{SpSt}(2n, 2k)$ from the geodesics on $\mathrm{Sp}(2n)$.

Following [3, Prop. 2.1], given $p \in \mathrm{Sp}(2n)$, $X \in T_p\mathrm{Sp}(2n)$ and the right-invariant Riemannian metric (3.1), the respective geodesic $\gamma(t)$ is defined as

$$\gamma(t) \coloneqq \exp\big(t(Xp^{+} - (Xp^{+})^{\mathrm{T}})\big)\exp\big(t(Xp^{+})^{\mathrm{T}}\big)p,$$

where $\gamma(0) = p$, $\dot\gamma(0) = X$ and $\cdot^{+}$ is the symplectic inverse (as in (2.3)). Here, exp denotes the matrix exponential.

To be able define geodesics from the right invariant metric on $\mathrm{SpSt}(2n, 2k)$ we need the following result. According to [16, Cor. 7.46], if g_p has a horizontal tangent vector at every point, it projects to a Riemannian geodesic on $\mathrm{SpSt}(2n, 2k)$.

The last preliminary definition we need is the following. In [3, Lem. 3.11] Bendokat and Zimmermann prove that if we, for a point $p \in \mathrm{Sp}(2n)$, define a geodesic $\gamma(t)$ through a horizontal tangent vector $X \in \mathrm{Hor}_p\,\mathrm{Sp}(2n)$, then $\dot\gamma(t) \in \mathrm{Hor}_{\gamma(t)}\,\mathrm{Sp}(2n)$. We call such a geodesic a *horizontal geodesic.*

Following [3, Prop. 3.12], we can now define a geodesic on $\mathrm{SpSt}(2n,2k)$ through a horizontal geodesic $\gamma(t) \subseteq \mathrm{Sp}(2n)$. Let $q \in \mathrm{SpSt}(2n,2k)$, $Y \in T_q\mathrm{SpSt}(2n,2k)$, and $p \in \pi^{-1}(q) \subset \mathrm{Sp}(2n)$. Then the geodesic from p in direction Y is

$$\varphi(t) = \pi(\gamma(t)) = \exp\big(t(\overline{\Omega}(Y) - \overline{\Omega}(Y)^{\mathrm{T}})\big)\exp\big(t\overline{\Omega}(Y)^{\mathrm{T}}\big)p. \tag{3.6}$$

Now that we have an explicit expression for the geodesics on $\mathrm{SpSt}(2n,2k)$, we can use them to define the Riemannian gradient and Riemannian Hessian on $\mathrm{SpSt}(2n,2k)$.

3.3 The Riemannian gradient

One component to many optimization algorithms is the gradient. In the Euclidean setting the gradient of the cost function is a vector that points in the direction of the steepest ascent. Analogously, the Riemannian gradient points in the direction of the steepest ascent on the manifold. We define the notion of a Riemannian gradient rigorously in Definition 11. In this section we will state the Riemannian gradient explicitly, and prove that it satisfies necessary properties.

Proposition 1. *Given a function $f\colon \mathrm{SpSt}(2n,2k) \to \mathbb{R}$, the Riemannian gradient (as in Definition 11) with respect to g_p is given by*

$$\operatorname{grad} f(p) = \nabla f(p)p^Tp + J_{2n}p(\nabla f(p))^TJ_{2n}p, \tag{3.7}$$

where $\nabla f(p)$ is the Euclidean gradient of a smooth extension around $p \in \mathrm{SpSt}(2n,2k)$ in $\mathbb{R}^{2n\times 2k}$ at p.

Proof. We can see that this is the Riemannian gradient by the following two observations stated in [3, p. 12], which we verify ourselves below.

Firstly, the gradient must be in $T_p\mathrm{SpSt}(2n,2k)$, which means by (2.11) that $0 = p^+\operatorname{grad} f(p) + (\operatorname{grad} f(p))^+p$. Computing this we get

$$p^{\mathrm{T}}J\nabla f(p)p^{\mathrm{T}}p + p^{\mathrm{T}}JJp(\nabla f(p))^{\mathrm{T}}Jp + p^{\mathrm{T}}p(\nabla f(p))^{\mathrm{T}}Jp + p^{\mathrm{T}}J^{\mathrm{T}}\nabla f(p)p^{\mathrm{T}}J^{\mathrm{T}}Jp = 0,$$

where we have used $JJ = -J^{\mathrm{T}}J = -I_{2n}$ and (2.1).

Secondly, the gradient also has to satisfy $g_p(\operatorname{grad} f(p), X) = \mathrm{d}\,f_p(X) = \mathrm{tr}((\nabla f(p))^{\mathrm{T}}X)$ for all $X \in T_p\mathrm{SpSt}(2n,2k)$:

$$g_p(\operatorname{grad} f(p), X) = \mathrm{tr}\big((p^{\mathrm{T}}p(\nabla f(p))^{\mathrm{T}} + p^{\mathrm{T}}J^{\mathrm{T}}\nabla f(p)p^{\mathrm{T}}J^{\mathrm{T}})(I_{2n} - \tfrac{1}{2}G)X(p^{\mathrm{T}}p)^{-1}\big),$$

where $G := J^{\mathrm{T}}p(p^{\mathrm{T}}p)^{-1}p^{\mathrm{T}}J$. Expanding this expression we obtain

$$\begin{aligned} &= \mathrm{tr}\big(p^{\mathrm{T}}p(\nabla f(p))^{\mathrm{T}}X(p^{\mathrm{T}}p)^{-1}\big) - \tfrac{1}{2}\mathrm{tr}\big(p^{\mathrm{T}}p(\nabla f(p))^{\mathrm{T}}GX(p^{\mathrm{T}}p)^{-1}\big)\\ &+ \mathrm{tr}\big(p^{\mathrm{T}}J^{\mathrm{T}}\nabla f(p)p^{\mathrm{T}}J^{\mathrm{T}}X(p^{\mathrm{T}}p)^{-1}\big) - \tfrac{1}{2}\mathrm{tr}\big(p^{\mathrm{T}}J^{\mathrm{T}}\nabla f(p)p^{\mathrm{T}}J^{\mathrm{T}}GX(p^{\mathrm{T}}p)^{-1}\big), \end{aligned}$$

where the cancellations used the fact that the trace is invariant under circular shifts. Noting that the first term is by definition $\mathrm{d}\,f_p(X)$, and inserting the definition of G, the expression becomes

$$\begin{aligned} &= \mathrm{d}\,f_p(X) - \tfrac{1}{2}\mathrm{tr}\big((\nabla f(p))^{\mathrm{T}}J^{\mathrm{T}}p(p^{\mathrm{T}}p)^{-1}p^{\mathrm{T}}JX\big)\\ &+ \mathrm{tr}\big(p^{\mathrm{T}}J^{\mathrm{T}}\nabla f(p)p^{\mathrm{T}}J^{\mathrm{T}}X(p^{\mathrm{T}}p)^{-1}\big)\\ &- \tfrac{1}{2}\mathrm{tr}\big(p^{\mathrm{T}}J^{\mathrm{T}}\nabla f(p)p^{\mathrm{T}}J^{\mathrm{T}}J^{\mathrm{T}}p(p^{\mathrm{T}}p)^{-1}p^{\mathrm{T}}JX(p^{\mathrm{T}}p)^{-1}\big). \end{aligned}$$

After using $J^{\mathrm{T}}J^{\mathrm{T}} = -I_{2n}$ and (2.1) on the last term, we notice that we can cancel $p^{\mathrm{T}}p(p^{\mathrm{T}})p^{-1}$, making it equal to the second to last term. Now focusing on the second term: for the first equality we use the fact that for any matrix, A, $\mathrm{tr}(A) = \mathrm{tr}(A^{\mathrm{T}})$, and for the second equality we utilize the cyclic property of the trace, and (2.1),

$$\begin{aligned} \tfrac{1}{2}\mathrm{tr}\big((\nabla f(p))^{\mathrm{T}}J^{\mathrm{T}}p(p^{\mathrm{T}}p)^{-1}p^{\mathrm{T}}JX\big) &= \tfrac{1}{2}\mathrm{tr}\big(X^{\mathrm{T}}J^{\mathrm{T}}p(p^{\mathrm{T}}p)^{-1}p^{\mathrm{T}}J\nabla f(p)\big)\\ &= -\tfrac{1}{2}\mathrm{tr}\big(p^{\mathrm{T}}J^{\mathrm{T}}\nabla f(p)X^{\mathrm{T}}J^{\mathrm{T}}p(p^{\mathrm{T}}p)^{-1}\big). \end{aligned} \tag{3.8}$$

Inserting (3.8) into our expression we end up with:

$$g_p(\operatorname{grad} f(p), X) = \mathrm{d}\, f_p(X) + \tfrac{1}{2}\mathrm{tr}\big(p^{\mathrm{T}} J \nabla f(p) X^{\mathrm{T}} J^{\mathrm{T}} p (p^{\mathrm{T}} p)^{-1}\big) + \tfrac{1}{2}\mathrm{tr}\big(p^{\mathrm{T}} J^{\mathrm{T}} \nabla f(p) p^{\mathrm{T}} J^{\mathrm{T}} X (p^{\mathrm{T}} p)^{-1}\big),$$

where the last two terms cancel after applying (2.1), and the tangent space condition (2.11), $p^{\mathrm{T}} J X = -X^{\mathrm{T}} J p$. □

Now that the expression for the Riemannian gradient is established, we have almost all the tools needed to be able to define the Riemannian Hessian on $\mathrm{SpSt}(2n, 2k)$. In the next section we will define the remaining concepts, before providing an analytical expression for the Riemannian Hessian.

3.4 The Riemannian Hessian

If the cost function is smooth enough, the Hessian of the cost function can add information to an algorithm that lower order algorithms do not have access to. The hope would be that this information would make the algorithm more efficient. In the Euclidean setting, the Hessian gives us curvature information about the cost function. Generalizing the Hessian to the Riemannian setting it gives us something similar, and we can generalize many popular algorithms. In this section we will therefore define the Riemannian Hessian of the symplectic manifold. We begin by defining what it means to take a second derivative on a manifold. After defining the necessary tools, we will give an analytical expression for the Riemannian Hessian.

Loosely following [12, Appendix A], for two smooth vector fields, $\mathcal{X}(p) = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\partial}$ and $\mathcal{Y}(p) = \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\partial}$ defined as in Definition 5, the covariant derivative (defined through the Riemannian connection by Definition 9) written in local coordinates is

$$\nabla_{\mathcal{X}}\mathcal{Y} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \partial_i(\beta_j)\partial_j + \alpha_i\beta_j \sum_{k=1}^{n} \Gamma_{ij}^{k}\partial_k.$$

In preparation for the Hessian, we include the description of the covariant derivative from [17, p. 96] to restrict the covariant derivative further. We want to define the covariant derivative of a vector field along a curve $c(t)$. $c(t)$ is the smooth curve, $c\colon I \to \mathcal{M},\ t \mapsto (\gamma_1(t), \dots, \gamma_n(t))$, where $I := [a, b] \subseteq \mathbb{R}$. Since we have an affine connection on $\mathcal{M}$, the following unique map exists:

$$\frac{D}{\mathrm{d}t}\colon \Gamma(T\mathcal{M}|_{c(t)}) \to \Gamma(T\mathcal{M}|_{c(t)}),$$

where $\Gamma(T\mathcal{M}|_{c(t)})$ denotes the the vector space of all smooth vector fields along $c(t)$. If $V \in \Gamma(T\mathcal{M}|_{c(t)})$ is induced by $\mathcal{X}$, meaning $V(t) = \mathcal{X}|_{c(t)}$, then

$$\frac{DV}{\mathrm{d}t}(t) = \nabla_{\dot{c}(t)}\mathcal{X} = \dot{\alpha}(t) + \Gamma(\alpha(t), \dot{\gamma}(t)), \quad \Gamma(u, v) = \begin{bmatrix} u^{\mathrm{T}}\Gamma^1 v \\ \vdots \\ u^{\mathrm{T}}\Gamma^n v \end{bmatrix}.$$

$\Gamma(u, v)$ is called the *Christoffel function.* If $\dot{c}(t)$ is a geodesic, the expression above reduces to

$$\ddot{\gamma}(t) = -\Gamma(\dot{\gamma}(t), \dot{\gamma}(t)), \tag{3.9}$$

since by definition geodesics must satisfy $\frac{D}{\mathrm{D}t}\dot{\gamma}(t) = 0$ and $\dot{\alpha}(t) = \dot{\gamma}(t)$. Importantly, once we have found the Christoffel symbols through (3.9), we can still use them for curves that are not geodesics. This is because the Christoffel symbols only depend on the Riemannian metric, and the local coordinates. To do this, we recover the Christoffel function for two different inputs through polarization [9, p. 312]

$$\Gamma(X, Y) = \frac{1}{4}\big(\Gamma(X + Y, X + Y) - \Gamma(X - Y, X - Y)\big),$$

where $X, Y \in \Gamma(T\mathcal{M}|_{c(t)})$.

Following [12, p. 10], to find the Christoffel symbols for $\mathrm{SpSt}(2n, 2k)$ with respect to the right invariant metric g defined in (3.5), we differentiate the geodesic formula from (3.6), and use (3.9) to achieve the following formula,

$$\Gamma(X,X) = -\ddot{\gamma}(0) = -(\overline{\Omega}(X) - \overline{\Omega}(X)^{\mathrm{T}})(X + \overline{\Omega}(X)^{\mathrm{T}}p) - (\overline{\Omega}(X)^{\mathrm{T}})^2 p.$$

Here $X = \dot{\varphi}(0) \in T_p\mathrm{SpSt}(2n, 2k)$, $p \in \mathrm{SpSt}(2n, 2k)$, and $\overline{\Omega}(X)$ is as in (3.4). With our metric g, the Hessian at p of a smooth function $f\colon \mathrm{SpSt}(2n, 2k) \to \mathbb{R}$ is the endomorphism

$$\begin{aligned} \operatorname{Hess} f(p)\colon T_p\mathrm{SpSt}(2n,2k) &\to T_p\mathrm{SpSt}(2n,2k), \\ \operatorname{Hess} f(p)[X] &= \frac{\mathrm{d}}{\mathrm{d}t} \operatorname{grad} f(c(t))\Big|_{t=0} + \Gamma(\operatorname{grad} f(p), X), \end{aligned} \tag{3.10}$$

where $\operatorname{grad} f(\cdot)$ is as in (3.7). $c(t) \in \mathrm{SpSt}(2n, 2k)$ is an arbitrary curve such that $c(0) = p$ and $c'(0) = X$. Although powerful, the Hessian can become cumbersome to compute. In the next section we will give an alternative way: to compute an approximate Hessian that has the potential to be more computationally efficient.

3.5 Moving from Theory to Application

In this section, we will discuss various ways to improve the computational efficiency of the theoretical results presented above. We will first introduce the Cayley retraction, and the pseudo-Riemannian geodesic, which both approximate how we compute geodesics. Secondly we will introduce an approximate Hessian. The formulas to be optimized share a common computational challenge: they both involve computations of matrix exponentials, which are computationally expensive.

From [12, p. 7] define the *Cayley transformation* as the first-order expansion of the matrix exponential,

$$\exp(2p) \approx (I - p)(p - I)^{-1} =: \mathrm{Cay}(p).$$

Though it is only an approximation, it has the property that $\mathrm{Cay}\colon \mathfrak{sp}(2n) \to \mathrm{Sp}(2n)$. Inserting this into (3.6) gives us the *Cayley retraction* given by

$$\hat{\mathcal{R}}_p(X) = \mathrm{Cay}\left(\frac{1}{2}(\overline{\Omega}(X) - \overline{\Omega}(X)^{\mathrm{T}})\right) \mathrm{Cay}\left(\frac{1}{2}\overline{\Omega}(X)^{\mathrm{T}}\right) p,$$

where we define the notion of retraction in Definition 13. However, we will proceed with the even simpler, yet shown to be sufficient, retraction presented in [3, p. 20]:

$$\begin{aligned} \mathcal{R}_p(tX) &:= \mathrm{Cay}\left(\frac{t}{2}\tilde{\Omega}(p, X)\right) p \\ &= -p + (tq + 2p)\left(\frac{t^2}{4}q^+q - \frac{t}{2}r + I\right)^{-1}, \end{aligned} \tag{3.11}$$

where $\tilde{\Omega}(p, X) := \left(I - \frac{1}{2}pp^+\right)Xp^+ - pX^+\left(I - \frac{1}{2}pp^+\right)$, $r := p^+X$, and $q := X - pr$. In (3.11), the last equality comes from [3, Prop. 5.2].

Functioning as a numerical middle ground between the geodesic (3.6) on $\mathrm{SpSt}(2n, 2k)$ and the Cayley retraction (3.11), we now define the pseudo-Riemannian geodesic described in [3, p. 10]. While invoking [3] to explain the underlying theory, the pseudo-Riemannian geodesic is defined for $X \in \mathrm{SpSt}(2n, 2k)$ as

$$\phi(t) = \begin{bmatrix} p & \frac{1}{2}pr + qz \end{bmatrix} \exp\left(t\begin{bmatrix} \frac{1}{2}r & \frac{1}{2}r^2 - q^+q \\ I_{2n} & \frac{1}{2}r \end{bmatrix}\right)\begin{bmatrix} I_{2k} \\ 0 \end{bmatrix},$$

where q and r are as above.

To approximate the Riemannian Hessian defined in (3.10) we include the following approximation from [6, Corr. 5.16]. Since $\mathrm{SpSt}(2n, 2k)$ is a submanifold of a Euclidean space, if one uses the Euclidean metric, the Hessian can be expressed as

$$\operatorname{Hess} f(p)[X] = \mathrm{Proj}_p(\mathrm{D}\,\overline{\operatorname{grad}} f(p)[X]), \tag{3.12}$$

where $\overline{\text{grad}} f(p)$ is a smooth extension of grad $f(p)$, and $\text{Proj}_p(\cdot)$ is the projection onto the tangent space of p. Since we are using the metric g_p defined in (3.5), (3.12) is only an approximation of the Hessian defined in (3.10) [12, p. 11]. The projection $\text{Proj}_p(\cdot)$ is defined explicitly in [12, Lem. 2.3], but we will solve it numerically through the following optimization problem. For $q \in \mathbb{R}^{2n\times 2k}$,

$$\text{Proj}_p(A) \approx \min_{r\in\mathbb{R}^{2n\times 2k}} \frac{1}{2}||r-q||^2, \quad \text{subject to} \quad r^{\mathrm{T}}Jp + p^{\mathrm{T}}Jr = 0. \tag{3.13}$$

Having introduced the Cayley retraction, the pseudo-Riemannian geodesic, and the approximate Hessian, we have increased our toolbox in applying the theoretical results to optimization problems. The last thing on our agenda before we can conduct our experiments is to introduce the optimization algorithms we will use.

4 Algorithms

Now that we have defined the necessary theory, we can describe the algorithms of interest in this project. These algorithms use the right-invariant framework on $\text{SpSt}(2n, 2k)$ to perform Riemannian optimization. By this we mean that, either through utilizing the geodesics or the Riemannian Hessian, the algorithms leverage this information to optimize some function defined on $\text{SpSt}(2n, 2k)$. The algorithms we investigate are the Riemannian gradient descent algorithm, and the Riemannian trust-region method. They are used by Jensen and Zimmermann for optimization on $\text{SpSt}(2n, 2k)$. One of our goals is to study the feasibility of these algorithms by attempting to reproduce some of the findings of Jensen and Zimmermann [12] as well as Bendokat and Zimmermann [3]. After introducing the two algorithms, we will outline how we implemented them, and how we conducted our experiments.

4.1 Riemannian gradient descent

The first algorithm we will define is Riemannian gradient descent (GD). Although seemingly simple, the Euclidean gradient descent (E-GD) is a powerful algorithm because of its simplicity. THrough the use of only a few assumptions, the algorithm applicable to a wide range of problems, and its simplicity also makes it computationally efficient. Initially following [6, p. 56], the E-GD is intuitively transferred to the Riemannian framework. For the sequence $\{p_k\}$ where $k \in \mathbb{N}$, we have the point $p_k \in \text{SpSt}(2n, 2k)$, where the next point in the sequence is computed as

$$p_{k+1} = \mathcal{R}_{p_k}(-t_k \,\text{grad}\, f(p_k)).$$

Here $t_k > 0$ is some step-size to be determined.

To ensure that reach step sufficiently decreases the cost function, we will use the Riemannian version of the Armijo condition to compute t_k. In [10, p. 17] the Armijo condition for each iteration k given for $\beta \in (0, 1)$, and a search direction X_k as

$$f\big(\mathcal{R}_{p_k}(t_k X_k)\big) \leq f(p_k) + \beta t_k \langle \text{grad}\, f(p_k), X_k\rangle_{p_k}. \tag{4.1}$$

The step-size t_k is calculated as $t_k = \gamma_k \delta^h$, where $\gamma \in (0, 1)$ is the backtracking parameter and h is the smallest integer such that (4.1) is satisfied.

Algorithm 1 Riemannian gradient descent

Input: Initial point $p_0 \in \mathrm{SpSt}(2n, 2k)$, objective function $f\colon \mathrm{SpSt}(2n, 2k) \to \mathbb{R}$, retraction $\mathcal{R}$, parameters $\beta, \gamma \in (0, 1)$, step length range $0 < \gamma_{\min} < \gamma_{\max}$, initial step size $\gamma_0 = f(p_0)$, maximum number of iterations $N \in \mathbb{N}$, step parameters $h_{\min} < h_{\max} \in \mathbb{Z}$, tolerance parameters $\epsilon, \epsilon_p > 0$, Riemannian metric $\langle \cdot, \cdot \rangle_p$, with Riemannian gradient $\operatorname{grad} f(p)$, and $||\cdot, \cdot||_{\mathrm{F}}$ denotes the Frobenius norm.

```
1: for $0 \le k \le N$ do
2:     $X_k = -\operatorname{grad} f(p_k)$
3:     $\gamma_k = \max(\gamma_{\min}, (\gamma_k, \gamma_{\max}))$
4:     Find $t_k$ through solving (4.1)
5:     $p_{k+1} = \mathcal{R}_{p_k}(t_k X_k)$
6:     if $||\operatorname{grad} f(p_k)||_{\mathrm{F}} < \epsilon$ or $\frac{||p_k - p_{k+1}||_{\mathrm{F}}}{\sqrt{2n}} < \epsilon_p$ then
7:         Break
8:     end if
9: end for
```

Output: Iterates $\{p_k\}$

For robustness it is important to know if this algorithm behaves properly, in the sense that we want it to reliably converge to critical points of f. It is shown in [10, Cor. 5.8] that for $\mathrm{SpSt}(2n, 2k)$ Algorithm 1 generates an infinite sequence of iterates $\{p_k\}$[10, Prop. 5.6]. It can be shown that every accumulation point p^* of $\{p_k\}$ is a critical point of f, meaning $||\operatorname{grad} f(p^*)|| = 0$.

Having outlined the first order method Riemannian gradient descent, the next section will introduce the second order method Riemannian trust-region.

4.2 Riemannian trust-region method

Since the gradient descent method only uses first order information, a natural question would be to investigate how a method that utilizes second order information would perform. The criterion for this second order method to be considered better, for a specific problem, than GD would be that it, despite the (expected) increased computing time per step, would converge with few enough steps as to still beat GD.

In choosing a second order method, the simplest choice would be a Riemannian version of Newton's method (NM). Despite it's simple design, NM is known to be quite unstable, and need to be sufficiently close to a local minima to guarantee convergence [6, p. 122]. The trust-region method (TR) addresses several of the problems with NM, while still having comparatively fast convergence properties as NM locally.

Following [6, p. 131], the goal of each step k of TR is for a point p_k, is to approximate the pullback $f \circ \mathcal{R}_{p_k}(X)$ through an approximation of $T_{p_k}\mathrm{SpSt}(2n, 2k)$,

$$f(\mathcal{R}_{p_k}(X)) \approx m_{p_k}(X) = f(p_k) + \langle \operatorname{grad} f(p_k), X \rangle_{p_k} + \frac{1}{2} \langle H_{p_k}(X), X \rangle_{p_k}.$$

H_{p_k} can be any self-adjoint linear map on $T_{p_k}\mathrm{SpSt}(2n, 2k)$, but in our experiments H_{p_k} will be either $\operatorname{Hess} f(p_k)[X]$ as in (3.10), or $\operatorname{Proj}_{p_k}(\mathrm{D}\,\overline{\operatorname{grad}} f(p_k)[X])$ as in (3.12). From [6, Prop. 5.44], H_{p_k} is essentially the same as $\operatorname{Hess} f(p)[X]$ when close to critical points.

To choose a step size for TR we demand that the step must reduce the value of $m_{p_k}(X)$. Since our model is an approximation in the tangent space, we construct a trust region which is the region around p_k where we assume that the error in our approximation is negligible. We solve the TR subproblem

$$\min_{X \in T_{p_k}\mathrm{SpSt}(2n,2k)} m_k(X) \quad \text{subject to} \quad ||X||_{p_k} \le \Delta_k \tag{4.2}$$

to find the candidate step X_k, where candidate for next iterate then is $\hat{p}_k = \mathcal{R}_{p_k}(X_k)$. Δ_k denotes the radius of the trust region at that iterate. Depending on how the new step performs (see line

5 of Algorithm 2) it is either accepted or rejected. Finally the trust region radius is evaluated to see if it needs to be modified. The procedure is codified in Algorithm 2, which is adapted from [6, Algorithm 3.3]. To solve the subproblem in line 2 of Algorithm 2 we employ, as Jensen and Zimmermann did, the *truncated conjugate gradients method*. We will not go into more detail in this report, but further reading can be found in [6, p. 131].

Algorithm 2 Riemannian Trust-region method

Input: Initial point $p_0 \in \mathrm{SpSt}(2n, 2k)$, objective function $f\colon \mathrm{SpSt}(2n,2k) \to \mathbb{R}$, retraction $\mathcal{R}$, maximum number of iterations $N \in \mathbb{N}$, maximal radius $\overline{\Delta} > 0$, initial radius $\Delta_0 \in (0, \overline{\Delta})$, ratio of model improvement threshold $\gamma_{\min} > 0$, tolerance parameter $\epsilon > 0$, Riemannian metric $\langle\cdot,\cdot\rangle_p$, with Riemannian gradient $\operatorname{grad} f(p)$, and $||\cdot,\cdot||_{\mathrm{F}}$ denotes the Frobenius norm.

```
1:  for $0 \leq k \leq N$ do
2:      Find $X_k$ through solving (4.2)
3:      $\hat{p} = \mathcal{R}_{p_k}(X_k)$
4:      $\gamma_k = \big(f(p_k) - f(\hat{p})\big)/\big(m_k(0) - m_k(X_k)\big)$
5:      Compute new iterate
            $$p_{k+1} = \begin{cases} \hat{p} & \text{if } \gamma_k > \gamma_{\min} \\ p_k & \text{otherwise} \end{cases}$$
6:      Compute new trust-region radius
            $$\Delta_{k+1} = \begin{cases} \frac{1}{4}\Delta_k & \text{if } \gamma_k < \frac{1}{4} \\ \min\Big\{2\Delta_k, \overline{\Delta}_k\Big\} & \text{if } \gamma > \frac{3}{4} \text{ and } ||X_k|| = \Delta_k \\ \Delta_k & \text{otherwise} \end{cases}$$
7:      if $||\operatorname{grad} f(p_k)||_{\mathrm{F}} < \epsilon$ then
8:          Break
9:      end if
10: end for
```

Output: Iterates $\{p_k\}$

Convergence and stability of TR is presented in-depth in [6, p. 147]. However, for consistency we note here that given sufficient assumptions (which are met in our examples), by [6, Cor. 6.24] for $\{p_k\}$ generated by TR,

$$\liminf_{k\to\infty} ||\operatorname{grad} f(p_k)||_{p_k} = 0.$$

In other words, this means that for all $\epsilon > 0$ and K there exists $k \geq K$ such that $||\operatorname{grad} f(p_k)||_{p_k} \leq \epsilon$.

Now that we have defined the Riemannian counterparts for GD and TR, we have all of the tools necessary to conduct our experiments. In the next section we will discuss some minor details around the implementation before moving on to the actual experiments.

4.3 Implementation

In this section we will outline how we will conduct our experiments. We will first define which retractions and algorithms we will test, as well as presenting the main packages we will use to implement the experiments. Finally we will cite where one can find the code used in the following experiments.

Before comparing algorithms, we will first look at the feasibility off the different retractions defined in Section 3.5. These are the Cayley retraction, the pseudo-Riemannian geodesic, and the Riemannian geodesic. The motivation for this experiment is to test how the different retractions perform in terms of speed and accuracy, with the goal of finding which retraction to use in the subsequent experiments.

We will use three algorithms for our experiments. The first is the gradient descent algorithm using the Armijo condition to find appropriate step sizes, denoted as GD. The second is the trust-region algorithm using the exact Riemannian Hessian $\operatorname{Hess} f(p_k)[X]$ as in (3.10), denoted as TR-1. The final algorithm, which we will denote as TR-2, will differ from TR-1 by using the approximate Hessian $\operatorname{Proj}_{p_k}(\mathrm{D}\,\overline{\operatorname{grad}} f(p_k)[X])$ defined in (3.12). The shorthand TR will be used when referring to both TR-1 and TR-2.

The code for the experiments are mainly using the two packages Manifolds.jl [2] and Manopt.jl [4]. Manifolds.jl provides us with a useful framework to be able to work with their implementation of the symplectic Stiefel manifold. Manopt.jl provides us with a high level framework to be able to implement the optimization algorithms. An effort has been made to ensure the parameters in functions and objects defined in the utilized libraries are consistent as to align with the theory presented in this project report.

To measure the performance of the algorithms in terms of computational time, the function @benchmark from the package BenchmarkTools.jl [8] will be used.

Although many of the necessary tools for the experiments are already implemented in Manifolds.jl and Manopt.jl, the following components were implemented manually from the theory presented in this project report. These are the pseudo-Riemannian geodesic, the Riemannian Hessian and its components, and the approximate Hessian. The following is already implemented in the two libraries, but were manually implemented for debugging purposes: Cayley retraction, and the projection of Euclidean gradient to the Riemannian gradient.

All files used to produce the data in this report are available in the following github repository [11].

5 Numerical Experiments

In this section we present the numerical experiments, and results, conducted in this report. We have conducted the following three experiments on $\mathrm{SpSt}(2n, 2k)$:

1. retraction comparison between the Cayley retraction, Riemannian geodesics, and pseudo-Riemannian geodesics through the nearest symplectic matrix problem using GD,
2. comparing the performance of GD, TR-1, and TR-2 through the nearest symplectic matrix problem, and
3. comparing the GD, TR-1, and TR-2 through the problem of performing a proper symplectic decomposition.

The problem of nearest symplectic matrix is used in both Bendokat and Zimmermann [3, p. 25] as well as Jensen and Zimmermann [12, p. 15], and the problem of proper symplectic decomposition is used in Jensen and Zimmermann [12, p. 17]. The first experiment is conducted in a similar manner to Bendokat and Zimmermann [3, p. 25], where we leverage the problem of nearest symplectic matrix to compare the performance of the Cayley retraction, the Riemannian geodesic, and the pseudo-Riemannian geodesic through the use of GD. The second and third experiments are conducted similarly to the analogous experiments in Jensen and Zimmermann [12, p. 15 & 17] where the purpose of these two experiments is to compare the performance of the GD, TR-1, and TR-2 algorithms. The reason for conducting both of these experiments is to make the conclusions we derive more robust than if only one experiment was conducted.

The experiments are performed using Julia version 1.10.2 on the Laptop ACER Swift SF314-43 processor AMD Ryzen 7 5700U with Radeon Graphics 1.80GHz and 16GB of RAM. To ensure reproducibility, all of the random processes are seeded using the seed 42. For readability, the code is written in the Pluto.jl notebook format, a format that allows for interactive code execution and visualization in a similar manner to the Python equivalent ”Jupyter notebook”.

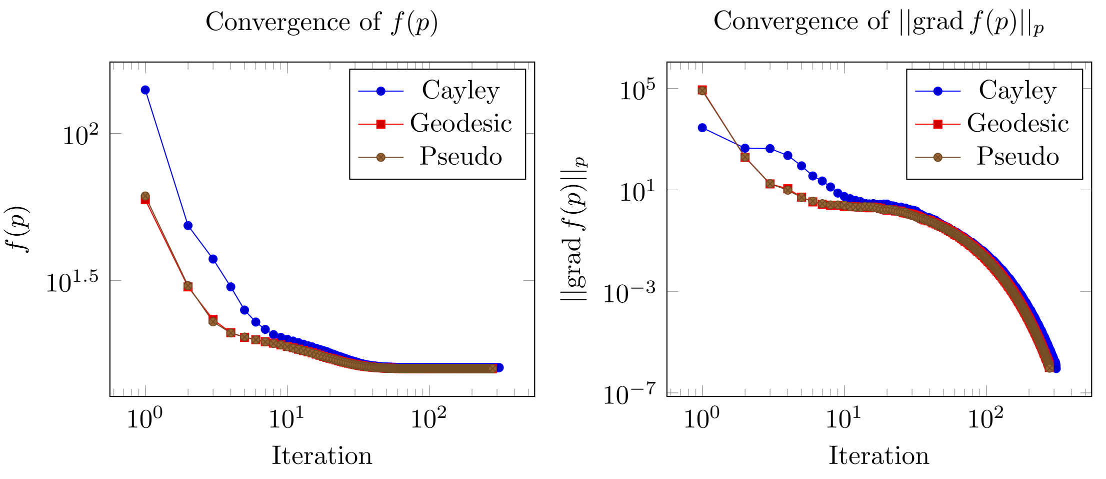

Figure 1: Nearest symplectic matrix problem solved by GD for different retractions. The figure displays the value of $f(p)$ and $||\text{grad}\, f(p)||_p$ on $\text{SpSt}(2n, 2k)$ as a function of iteration using Cayley retraction, Riemannian geodesics, and pseudo-Riemannian geodesics, with $n = 1000$ for $k = 20$.

Table 1: Nearest symplectic matrix problem solved by GD for different retractions. The table summarizes time to converge using Cayley retraction, Riemannian geodesics, and pseudo-Riemannian geodesics on $\text{SpSt}(2n, 2k)$, with $n = 1000$ for $k = \{5, 10, 20\}$.

	Runtime (s)		
	Cayley	Geodesic	Pseudo
$k = 5$	0.52	0.73	0.49
$k = 10$	1.6	2.8	1.7
$k = 20$	5.4	7.9	5.0

5.1 Nearest Symplectic Matrix Problem - Retraction comparison

Before we compare GD and TR, we will verify the performance of the Cayley retraction, Riemannian geodesics, and pseudo-Riemannian geodesics on $\text{SpSt}(2n, 2k)$. To test and compare the feasibility of our retractions we will, similarly to [3, p. 25], try to solve the following problem called the *nearest symplectic matrix problem.* For a matrix $q \in \mathbb{R}^{2n\times 2k}$ we want to find the closest symplectic matrix $p \in \text{SpSt}(2n, 2k)$. We formalize this in as the following optimization problem,

$$\min_{p\in\text{SpSt}(2n,2k)} f(p), \quad \text{where} \quad f(p) := \tfrac{1}{2}||q - p||_{\text{F}}^2. \tag{5.1}$$

For a point $p \in \text{SpSt}(2n, 2k)$ and $X \in T_p\text{SpSt}(2n, 2k)$, Euclidean gradient and Hessian of $f(p)$ are, respectively,

$$\nabla f(p) = p - q, \quad \nabla^2 f(p)[X] = X.$$

For the experiments we generate q randomly, and normalize it, $q \cdot ||q||_{\text{F}}^{-1}$. For the optimization runs we choose $n = 1000$ and $k = \{10, 50, 100\}$. The results is displayed in Table 1 and in Figure 1.

In Figure 1 we chose only to plot the optimization run for $n = 1000$, $k = 20$ since the other runs followed a similar pattern. We observe in the figure that the Riemannian Geodesic and the pseudo-Riemannian geodesic performed similarly in the sense that the pseudo-Riemannian geodesic did not drift away far from the Riemannian Geodesic in either the value for $f(p)$ nor $||\text{grad}\, f(p)||_p$. Cayley, on the other hand, is less accurate in the first ~ 10 iterations than the other two in both $f(p)$ and $||\text{grad}\, f(p)||_p$. Despite this, it quickly catches up to the others, and converges in a comparable amount of steps.

Regarding Table 1 we can see a clear pattern of all three using using more time to converge as we increase the dimension k. We observe that the Riemannian geodesic performs somewhat worse than

Table 2: Nearest symplectic matrix problem solved by GD, TR-1, and TR-2. The table summarizes time to converge for all the algorithms on $\mathrm{SpSt}(2n, 2k)$, with $n = 100$ for $k = 5, 10, 20$.

	Runtime (s)		
	GD	TR-1	TR-2
$k = 5$	0.054	0.21	1.7
$k = 10$	0.22	11	0.12
$k = 20$	0.99	31	0.27

the other two for all three runs. Interestingly, the Cayley retraction and the pseudo-Riemannian geodesic performed almost identically, in terms of wall-clock speed, in all three runs.

Despite the promising results of the pseudo-Riemannian geodesic, we will not use it in the following experiments, rather we will use the Cayley retraction. The reason for this is twofold. The first reason is that the Cayley retraction is already implemented in Manifolds.jl. This means that it has been tested and verified to work by numerous users, and developers, and is therefore probably more reliable. The second reason is that in an experiment in [3, p. 26] they found both the Riemannian geodesic and the pseudo-Riemannian geodesic to be exponentially more inaccurate for large step sizes. Because of these two reasons, in an effort to make the experiments more independent, we will proceed with using the Manifolds.jl implementation of the Cayley retraction.

5.2 Nearest Symplectic Matrix Problem - 2nd Order

As in the preceding section, we will conduct our experiments on the optimization problem called the nearest symplectic matrix problem, (5.1). Striving to replicate the conditions in [12, p. 15], for $\mathrm{SpSt}(2n, 2k)$ we select $n = 100$, and $k = \{5, 10, 20\}$. The algorithms tested are GD, TR-1, and TR-2. The results of these runs are summarized in Table 2 and Figure 2.

Figure 2 shows $f(p)$ and $||\mathrm{grad}\, f(p)||_p$ as a function of iteration for the three different runs with $k = \{5, 10, 20\}$, for the three algorithms GD, TR-1, and TR-2. We observe that for all the values of k, the algorithms behaved similarly in the sense that they all converged in a similar number of steps, and following a similar shape. We observe that TR-1 and TR-2 behave similarly, so we will denote both of them as TR. The only major pattern to be observed between k's is that the first ~ 15 iterations of TR in $k = 5$ have a steeper slope than GD. This difference is smaller for $k = 10$, and almost non-existent in $k = 20$. Looking at $||\mathrm{grad}\, f(p)||_p$ we observe that the major difference in convergence, as theoretically predicted, is visible after the first ~ 15 iterations. After this point, while the convergence rate of GD seem to slow down, the convergence rate of TR seems to increase. It should be noted that this "speeding up" and "slowing down" is an artifact of the log-log plotting, but it is clearly showing that the rate of convergence close to the critical point is fundamentally different between TR and GD. It should also be noted that all algorithms terminated as the result of hitting the condition $||\mathrm{grad}\, f(p)||_p \leq 10^{-6}$. This can give the perception that the final iteration of TR is more precise than GD. However, this is probably an artefact of the convergence rate of TR being so high that the final step before convergence is detected, significantly overshoots the convergence condition for $k = \{5, 10\}$.

Table 2 displays the time to converge for the runs displayed in Figure 2. We can see that GD has a significantly faster time to converge than TR-1 and TR-2 for $k = 5$. However, for $k = \{10, 20\}$, TR-2 takes the lead. We also observe that while TR-2 is faster than TR-1 for $k = \{10, 20\}$, it is not for $k = 5$. While GD and TR-1 have increasing runtimes as k increases, TR-2 has an initial dip from $k = 5$ to $k = 10$, before increasing again.

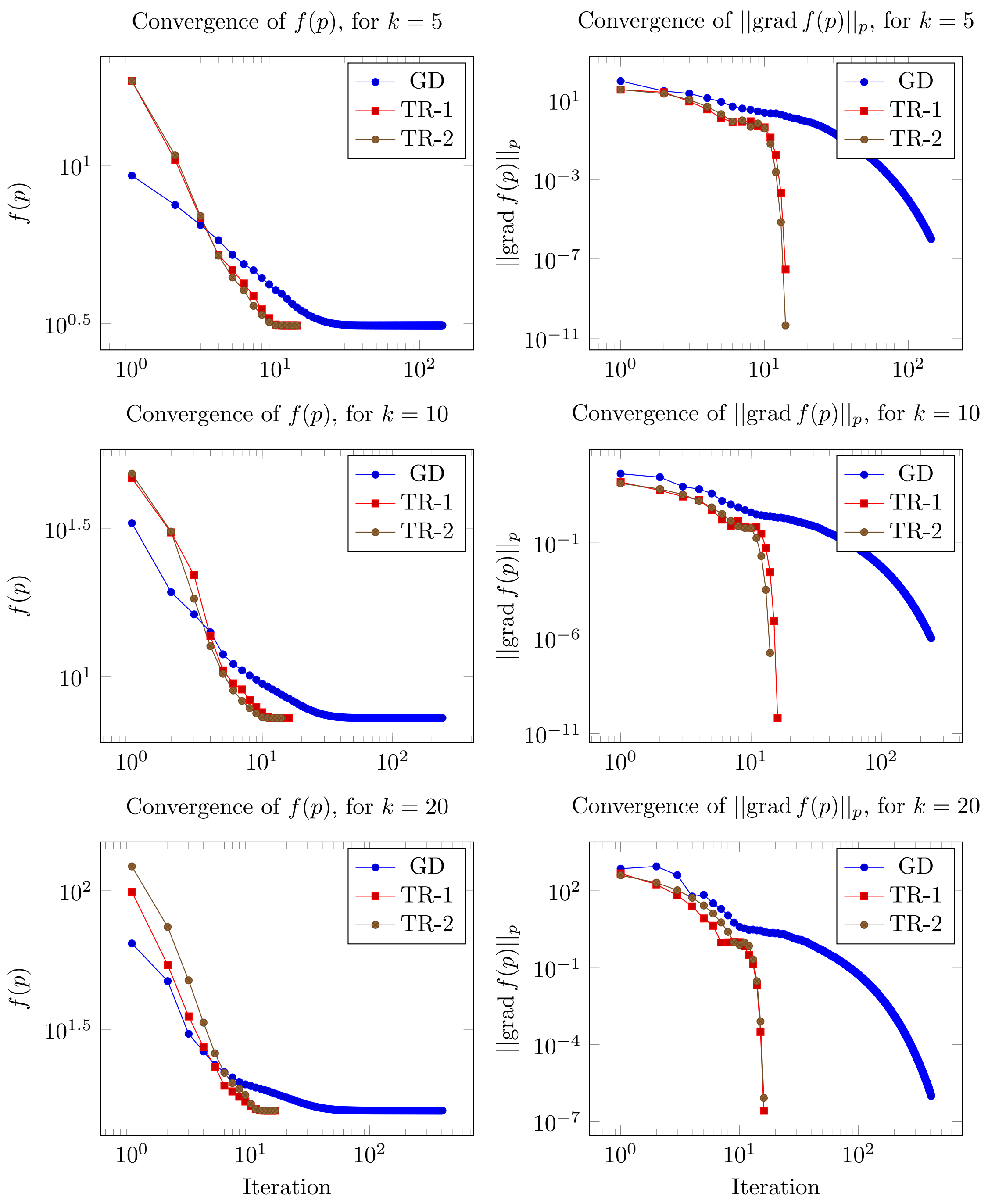

Figure 2: Nearest symplectic matrix problem solved by GD, TR-1, and TR-2. The figures show $f(p)$ and $\|\text{grad}\, f(p)\|_p$ on $\text{SpSt}(2n, 2k)$ as a function of iteration for all three algorithms, with $n = 100$ for $k = 5, 10, 20$.

Table 3: The proper symplectic decomposition problem attempted solved by GD, and solved by TR-1 and TR-2. The table summarizes time to converge for all the algorithms on $\mathrm{SpSt}(2n, 2k)$, with $n = 100$, $r = 4$, and $m = 10$ for $k = 1, 2, 3$.

	Runtime (s)		
	GD	TR-1	TR-2
$k = 1$	3.8	1.5	2.6
$k = 2$	3.7	11	4.7
$k = 3$	4.1	19	7.0

5.3 The Proper Symplectic Decomposition Problem

The final experiment we will conduct is to apply GD, TR-1, and TR-2 to the problem of computing the proper symplectic decomposition of a matrix $s \in \mathbb{R}^{2n\times 2m}$ in a similar manner to Jensen and Zimmermann [12, p. 18]. This problem is an important task in symplectic model order reduction of Hamiltonian systems. This problem can be formulated as the following optimization problem:

$$\min_{p\in\mathrm{SpSt}(2n,2k)} f(p), \quad \text{where} \quad f(p) \coloneqq ||s - pp^+s||_\mathrm{F}^2. \tag{5.2}$$

In most applications $k \ll n$. For a point $p \in \mathrm{SpSt}(2n, 2k)$ and $X \in T_p\mathrm{SpSt}(2n, 2k)$, Euclidean gradient and Hessian are, respectively,

$$\begin{aligned}\nabla f(p) &= -2\big(\eta(p)pJ - \eta(p)^\mathrm{T}pJ\big),\\ \nabla^2 f(p)[X] &= -2\big(\alpha(p,X)pJ + \eta(p)XJ - \alpha(p,X)^\mathrm{T}pJ - \eta(p)^\mathrm{T}XJ\big),\end{aligned}$$

where $\alpha(p, X) = \beta(p, X)ss^\mathrm{T}J^\mathrm{T}$, $\beta(p, X) = \mathrm{D}(I - pp^+)[X] = -Xp^+ + (-Xp^+)^+$, and $\eta(p) = (I - pp^+)ss^\mathrm{T}J^\mathrm{T}$. To construct s, we generate a matrix $u \in \mathrm{SpSt}(2n, 2r)$ as well as another matrix $v \in \mathbb{R}^{2r\times 2m}$, which we normalize; $v = v \cdot ||v||_\mathrm{F}^{-1}$. From these two matrices we generate $s = u \cdot v$. In an effort to replicate similar conditions as in [12, p. 18], for the optimization experiment we choose $n = 100$, $k = \{1, 2, 3\}$, $r = 4$, and $m = 10$.

Figure 3 shows $f(p)$ and $||\mathrm{grad}\, f(p)||_p$ as a function of iteration for the three different runs with $k = \{1, 2, 3\}$, for the three algorithms GD, TR-1, and TR-2. As in the nearest symplectic matrix problem we see that for the different values of k, the algorithms generally show a similar pattern of convergence to itself. The most striking pattern in the figures is that the convergence rate of GD slows down until it is not able to converge within iter = 1000, while both TR-1 and TR-2 do. Regarding TR-1 and TR-2, the actual pattern of convergence is similar to the pattern of convergence in the previous experiment. For all three values of k, TR clearly displays an elbow shape in the $||\mathrm{grad}\, f(p)||_p$ plot. After an initial period of a slower convergence rate, after it is close enough to the critical point the convergence rate increases dramatically. A less prominent pattern is that the relative convergence rate of TR-1 and TR-2, although initially similar, diverges as k increases. For $k = 3$ it TR-1 uses many more steps than TR-2 to converge.

In Table 3 the runtime for GD, TR-1, and TR-2 are displayed. As expected, for bigger systems the runtime for all three algorithms increase. Regarding GD, despite it not converging within 1000 iterations, the runtime is still comparably low. As GD did not converge for any of the values of k we will not comment on it further. For $k = 1$ TR-1 was the fastest, while for $k = \{2, 3\}$ TR-2 was significantly faster than TR-1.

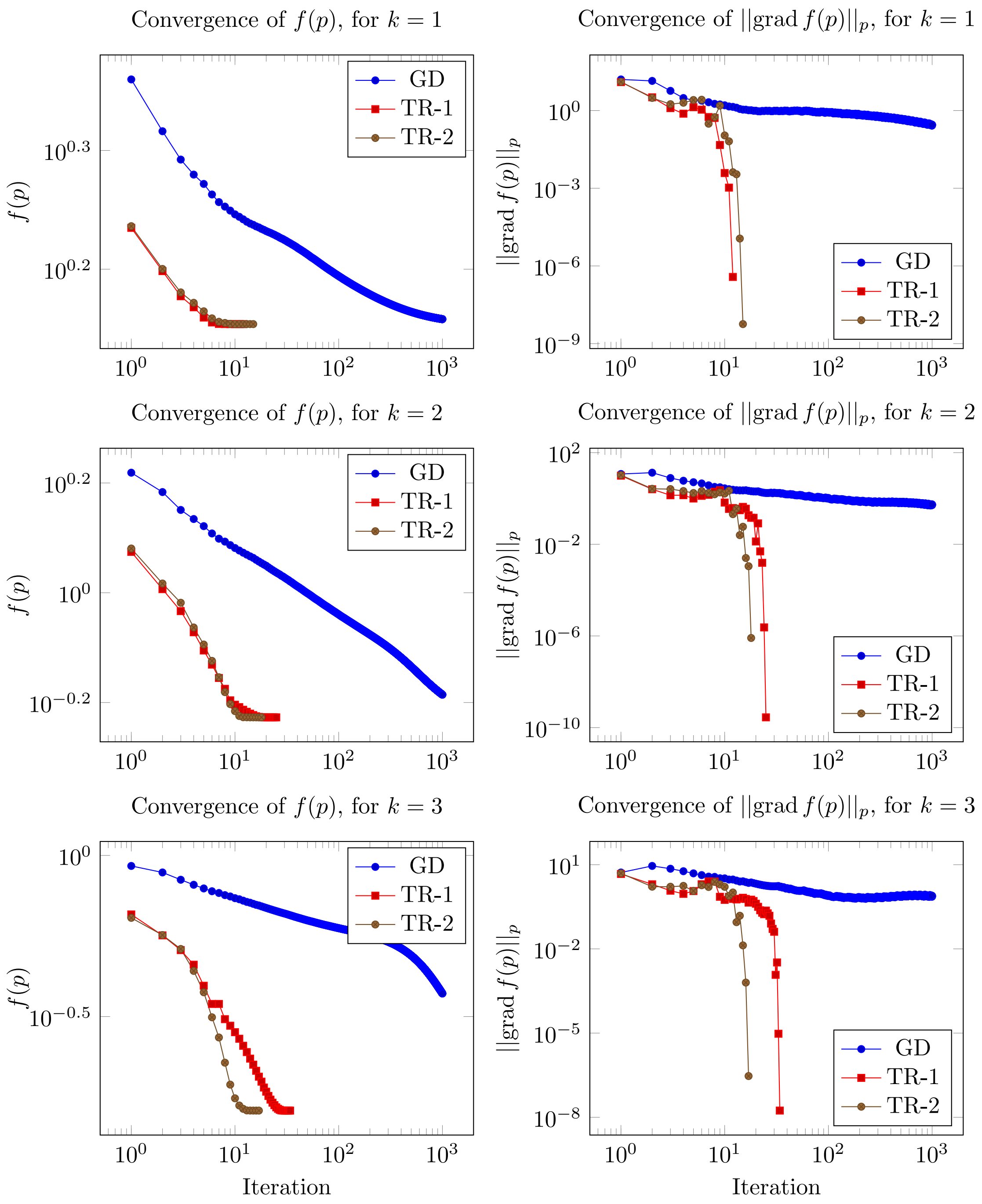

Figure 3: The proper symplectic decomposition problem attempted solved by GD, and solved by TR-1 and TR-2. The figures show $f(p)$ and $||\text{grad}\, f(p)||_p$ on $\text{SpSt}(2n, 2k)$ as a function of iteration for all three algorithms, with $n = 100$, $r = 4$, and $m = 10$ for $k = \{1, 2, 3\}$.

6 Conclusion

In this report we presented the right-invariant framework on the symplectic Stiefel manifold. Through this framework we presented the Riemannian gradient descent algorithm (GD) and the Riemannian trust-region method using the exact Riemannian Hessian (TR-1) and the approximate Riemannian Hessian (TR-2). We used GD to analyze the performance of the Cayley retraction, the pseudo-Riemannian geodesic retraction, and the Riemannian geodesic retraction on the problem of finding the nearest symplectic matrix. Lastly we used the same problem, as well as the proper symplectic decomposition problem, to compare the three algorithms.

The retraction experiment, in section 5.1, apart from reaffirming the dominance of the Cayley retraction over the Riemannian geodesic in terms of computational efficiency (as observed in both [12, p. 8] and [3, p. 26]), we also found that the pseudo-Riemannian geodesic performed comparably or better than the Cayley retraction. It is unexpected that the pseudo-Riemannian geodesic performed as well as it did, seeing as in [3, p. 28] it was excluded from a similar test to the one presented in this report. This was because it seemed to do rather unimpressively in an earlier test Bendokat and Zimmermann performed. Given that the pseudo-Riemannian geodesic was seemingly as accurate as the Riemannian geodesic, in addition to being as fast as the Cayley transformation, further investigation into the pseudo-Riemannian geodesic is warranted.

Regarding using the nearest symplectic matrix problem to compare GD, TR-1, and TR-2, the results for the experiments presented in Section 5.2 are mostly in line with the experiment done by Jensen and Zimmermann [12, Tbl. 4.1]. We observed on $\mathrm{SpSt}(2n, 2k)$ for $n = 100$, and $k = 5$ that GD was superior in terms of runtime, while on $k = \{10, 20\}$ TR-2 took the lead. Strangely this is mostly in line with what Jensen and Zimmermann observed. They observed that for $n = 1000$, and $k = \{10, 50, 100\}$, GD barely beat the others for $k = 10$, with TR-2 surpassing GD in terms of speed for $k = \{50, 100\}$. It is strange that the results are so similar, seeing as the size of the system is so different. One plausible explanation of the similarity in convergence behavior is that the relative size of n and k could play a significant role in the performance of the algorithms. Actually running the experiments for $n = 1000$, and $k = \{10, 50, 100\}$ through our implementation could provide more insight into this. It should be noted that Jensen and Zimmermann used a more costly retraction [12, Fig. 1] than what we used. Despite this the results for GD we observed are not in conflict with other experiments, as in [3, Tbl. 1]. It is also worth noting that around iteration 15 we see that for all three runs in Figure 2, GD starts to struggle to keep up with TR. Therefore, if speed is of more importance than accuracy, the number of cases where GD is superior to TR will probably be higher.

Another point worth noting is that the superior numerical accuracy by TR-1 and TR-2 pointed out by Jensen and Zimmermann, which they conclude is purely due to the fact that TR-1 and TR-2 make use of second order information, could be misinterpreted. As we commented on in Section 5.2, we can see in Figure 2 that TR-1 and TR-2 tend to overshoot the convergence condition of $||\mathrm{grad}\, f(p)||_p \leq 10^{-6}$, while GD does not. This effect has the potential to muddy the waters when comparing accuracies of TR-1 and TR-2 to GD. Despite this concern, we do observe that for larger systems, i.e. larger k, this effect became smaller. It is therefore possible that for the larger system Jensen and Zimmermann tested, the overshooting effect was negligible.

For the problem of performing a proper symplectic decomposition using GD, TR-1, and TR-2, as done in Section 5.3, the experiment yielded mixed results compared to the experiment done by Jensen and Zimmermann [12, Tbl. 4.4]. Regarding TR-1 and TR-2, they seem to share a comparable difference in runtime between each other as they did in Jensen and Zimmermann. The actual pattern of convergence for TR also aligns with the nearest symplectic element problem, which aligns with what is expected in theory. However, there are some discrepancies which we will now describe.

The biggest discrepancy between our results and the results of Jensen and Zimmermann [12, Tbl. 4.4] is the performance of GD. While GD had no issue converging within 1000 iterations in the experiment by Jensen and Zimmermann, even though their system was larger than ours, GD in our experiments failed to converge for all presented values of k. From Figure 3, it appears that the way GD failed to converge was through the step size becoming so small that the algorithm would not converge within a reasonable amount of iterations. This leads us to another discrepancy within our implementation of GD compared to Jensen and Zimmermann. In our implementation of GD the step size is determined by the Armijo condition, while Jensen and Zimmermann uses the Barzilai–Borwein (BB) method. Since the most likely way GD failed to converge was through the step size becoming too small too fast, the most likely way to fix this issue is to change the step size method to the BB method. However, as this explanation was inferred, further analysis as to why GD failed to converge is warranted.

An interesting discrepancy between our experiments and Jensen and Zimmermann, which appears in both nearest symplectic element- and the symplectic decomposition problem, is the inferior runtime performance of TR-2 for their respectively lowest value of k. This is in contrast to the results of Jensen and Zimmermann, where TR-2 was superior to TR-1 for all three values of k. These observations could mean that, for small systems, solving the subproblem (3.13) to find the projection is not worth the computational cost. It could also mean that this method of finding the projection is not as accurate for smaller systems. Implementing the analytical projection of the gradient, as defined in [12, Lem. 2.3], and comparing it to the optimization problem (3.13) could therefore be an interesting experiment.

It should be noted that since all of the experiments were performed on a laptop, there is a high degree of uncertainty in all of the timed runs. This is because, like most laptops, when they heat up above a certain temperature, the processor will throttle down to prevent overheating. This results in the operations taking longer time, and the time comparison between different runs becomes less reliable. Running the tests on a setup with a more stable temperature would likely produce more reliable results.

Bibliography

[1] P.-A. Absil, R Mahony and R Sepulchre. *Optimization Algorithms on Matrix Manifolds.* Princeton University Press, 2008. ISBN: 9780691132983.

[2] Seth D. Axen et al. 'Manifolds.Jl: An Extensible Julia Framework for Data Analysis on Manifolds'. In: *ACM Transactions on Mathematical Software* 49.4 (Dec. 2023). DOI: 10.1145/3618296.

[3] Thomas Bendokat and Ralf Zimmermann. *The real symplectic Stiefel and Grassmann manifolds: metrics, geodesics and applications.* 2021. arXiv: arXiv:2108.12447v1.

[4] Ronny Bergmann. 'Manopt.jl: Optimization on Manifolds in Julia'. In: *Journal of Open Source Software* 7.70 (2022), p. 3866. DOI: 10.21105/joss.03866.

[5] William M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry.* Vol. 63. Elsevier Science (USA), 1975. DOI: https://doi.org/10.1016/S0079-8169(08)61028-4.

[6] Nicolas Boumal. *An introduction to optimization on smooth manifolds.* Cambridge University Press, 2023. DOI: 10.1017/9781009166164. URL: https://www.nicolasboumal.net/book.

[7] Henri Bourles. *Fundamentals of Advanced Mathematics V3.* Elsevier Ltd., 2019. DOI: 10.1016/c2017-0-00728-0.

[8] Jiahao Chen and Jarrett Revels. *Robust benchmarking in noisy environments.* 2016. arXiv: arXiv:1608.04295.

[9] Alan Edelman, Tomás A. Arias and Steven T. Smith. 'The Geometry of Algorithms with Orthogonality Constraints'. In: *SIAM Journal on Matrix Analysis and Applications* 20 (1998). DOI: 10.1137/s0895479895290954. arXiv: arXiv:physics/9806030v1.

[10] Bin Gao et al. 'Riemannian optimization on the symplectic Stiefel manifold'. In: *SIAM Journal on Optimization* 31 (2021). DOI: https://doi.org/10.1137/20M1348522. arXiv: arXiv:2006.15226.

[11] Ole Gunnar Røsholt Hovland. *TMA4500-Project-Hovland-Symplectic-Stiefel.* 2024. URL: https://github.com/kellertuer/TMA4500-Project-Hovland-Symplectic-Stiefel.git.

[12] Rasmus Jensen and Ralf Zimmermann. *Riemannian optimization on the symplectic Stiefel manifold using second-order information.* 2024. arXiv: arXiv:2404.08463v2.

[13] Jon Arnt Kårstad. *Template Project NTNU.* URL: https://no.overleaf.com/latex/templates/template-project-ntnu/zjystqvqztpg (visited on 11th Sept. 2024).

[14] John M. Lee. *Introduction to Riemannian Manifolds.* 2nd ed. Springer Cham, 2018. DOI: 10.1007/978-3-319-91755-9.

[15] John M. Lee. *Introduction to Smooth Manifolds.* 2nd ed. Springer New York, NY, 2012. DOI: 10.1007/978-1-4419-9982-5.

[16] Barett O'Neill. *Semi-Riemannian Geometry With Applications to Relativity.* Vol. 103. Academic Press, 1983. ISBN: 9780125267403.

[17] Loring W. Tu. *Differential Geometry.* Springer International Publishing, 2017. DOI: 10.1007/978-3-319-55084-8.