Centro de Investigación en Matemáticas, A.C.

TRANSPORTLESS CONJUGATE GRADIENT FOR OPTIMIZATION ON STIEFEL MANIFOLDS

T E S I S

Que para obtener el grado de

Maestro en Ciencias

con Orientación en

Computación y Matemáticas Industriales

Presenta

Edgar Fuentes Figueroa

Director de Tesis:

Dr. Oscar Susano Dalmau Cedeño

Autorización de la versión final

Guanajuato, Gto., 19 de Diciembre de 2018

Abstract

In this work we develop a method for optimization over Stiefel manifold that allows us to use conjugate gradient methods without vector transport for tangent vectors.

Vector transport is a crucial concept in the performance of Riemannian conjugate gradient methods, we investigate the possibility of avoiding approximations and using parallel transport for tangent vectors via transforming problems over Stiefel manifold onto problems on Euclidean spaces.

However, our proposal relies in Riemannian concepts such as retraction. In particular, the Cayley retraction and the Polar Decomposition based retraction are used to parametrize Stiefel manifold. Taking advantage of said parametrization, we use a composite objective function defined over a vector space, the space of skew-symmetric matrices, therefore, no vector transport is needed.

Performance of our approach is compared with those of three state-of-the-art algorithms in a variety of numerical experiments. Our approach is seen to have fine performance but at higher computational cost.

Acknowledgements

The author is especially grateful to his parents and brother because of their unconditional support that has led him to the successful conclusion of this stage of his life. All efforts are rewarded and rewards taste better when shared with love ones.

Likewise, the author is grateful to the Center for Research in Mathematics, CIMAT, which granted him the opportunity of professional growth. Thanks to all researchers in the Computer Science department which make the Master's program the excellent program it is today.

Especially, thanks to Dr. Oscar Dalmau whose advice molded ideas and thoughts into this tangible work. Passion for scientific knowledge is the common denominator among CIMAT researchers, this passion is now acquired by the author thanks to the excellent classes of professors Dr. Johan Van Horebeek, Dr. Jean-Bernard Hayet, Dr. Sergio Ivaan Valdez Peña and Dr. Oscar S. Dalmau Cedeño.

Particularly, the author would like to thank Dr. César Octavio Monzón and M.S. Laura Puebla for their professional advice and support about his career.

The author wants to thank the National Council for Science and Technology, CONACYT, for the economical support provided. Personal preparation and conclusion of this work would not be possible without Council's support.

All problems seem small when there are friends around. Thanks to Ana Caren Hernández Soto, José Carlos Zamarripa Rodriguez, Cynthia Guadalupe De Loza Aguilar, Jamie Areli Toral Barrera and Alejandra Quintero García for being around in this journey.

Finally, thanks to Nayeli Ángel Pérez, Lourdes Merino, Patricia del Carmen, Guillermo Herrera and Gerardo Rodriguez, classmates that shared the challenges of this experience, classmates that at the end of the journey became friends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work concerns the solution of the problem

$$\min_{X\in\mathbb{R}^{n\times p}} \mathcal{F}(X), \quad \text{s.t.} \quad X^\top X = I_p, \tag{1.1}$$

where I_p is the $p-$by$-p$ identity matrix, $1 \leq p \leq n$ and $\mathcal{F}(X) : \mathbb{R}^{n\times p} \to \mathbb{R}$ is a differentiable function. The feasible set is the smooth manifold introduced in 1935 by Stiefel [46],

$$\mathrm{St}(n,p) := \{X \in \mathbb{R}^{n\times p} | X^\top X = I_p\}. \tag{1.2}$$

The existence of a solution to (1.1) is guaranteed since the domain, $\mathrm{St}(n,p)$, is a compact set [4]. Some particular examples of the Stiefel manifold are the sphere and the orthogonal group, when $p = 1$ and $p = n$, respectively.

However, finding a solution for (1.1) is difficult in general. Since $\mathrm{St}(n,p)$ is not a linear manifold nor a convex set, iterative schemes involve high computational cost is they are to maintain feasibility of iterates.

Nevertheless, wide applicability of problem (1.1) made it attractive to researchers.

1.1 Motivation

Many applications appear in the context of (1.1), e.g., the linear eigenvalue problem [22, 40], the orthogonal procrustes problem [23, 16, 44], the weighted orthogonal procrustes problem [19], the joint diagonalization problem [29, 47], heterogeneous quadratics minimization [7], the nearest low-rank correlation matrix problem [24, 38], singular value decomposition [41] and sparse principal component analysis [14, 30, 53].

Considering the wide applicability of optimization on the Stiefel manifold is natural that a lot of efforts have been made to develop capable algorithms for solving (1.1). However, the Euclidean geometry perspective could be non adequate to develop feasible optimization methods on $\mathrm{St}(n,p)$. A more suitable approach considers the Riemannian nature of the Stiefel manifold and the geometry yielded by it.

In 1973, Luebenberger [34] mentioned the idea of performing line search along geodesics, however, it is also mentioned that this would be a fine idea if it were computationally feasible. It was in 1982 [20] when Gabay actually developed a steepest descent along geodesics besides the Newton method and a Quasi-Newton method, also along geodesics. Later on, in 1994 Smith [45] added the Levi-Civita connection, the Riemmanian exponential and parallel traslation concepts to the Riemannian tool-box.

As good as the consideration of Riemannian geometry concepts is for optimization on Riemannian manifolds, there is always a compromise with computational cost. Some pragmatic simplifications have been made in order to reduce the computational effort. Manton [35] started not moving along geodesics and replaces the Riemmanian exponential mapping with a projection operator. Newton's method on Riemannian manifolds presented in [5] avoids the exponential update using the general notion of retraction. Also, the geodesic is replaced with any curve tangent to the search direction.

Good surveys on Riemannian optimization can be found in [4, 15], where the theory of Riemannian geometry is applied to develop the Riemannian versions of classical optimization methods such as gradient descent, Newton's method, trust-region methods and the conjugate gradient method. However, there are many references to algorithms developed from the Riemannian perspective including [2, 36, 35], for gradient descent, [52, 1, 42], for conjugate gradient, [3, 6, 50],for trust-region, and [43, 27], for quasi-Newton method. Riemannian optimization methods can be seen as generalizations of the corresponding classical methods.

The main feature of Riemannian optimization or optimization over manifolds is maintaining each iterate feasible. From the above it can be seen that the iterative schemes can be separated in two different kinds. Those that follow a curve, geodesic or not, and those that use a projection of some kind, namely, a retraction. However, other classification is presented in [28] where three general formulas comprehend most iterative schemes of literature.

Particularly, our work is inspired in the use of the Cayley Transform for Riemannian methods. Nishimori [36] proposed a quasi-geodesic scheme avoiding computing exponentials of matrices using the Cayley transform. However, the proposed Cayley transform scheme requires the inversion of a $n-$by$-n$ matrix which is also expensive. Fortunately, Wen and Yin [48] developed the Crank-Nicholson-like scheme which is equivalent to the Cayley transform scheme but has a very efficient implementation for low-rank-matrices. More recently, Zhu [52] developed a Riemannian conjugate gradient method considering the Cayley Retraction, Zhu's method had very competitive results when compared to other Riemannian conjugate gradient methods.

Riemannian conjugate gradient methods need a vector transport [4] and great performance of Zhu's method can be associated with the features of the vector transports developed in [52]. It has been shown that vector transport must satisfy the

Ring-Wirth nonexpansive condition in order to have global convergence [39] but not all vector transports satisfy this condition. Sato and Iwai introduced the notion of a scaled vector transport [42] in order to satisfy the nonexpansive condition but this kind of scaled vector transports violates the property of linearity. Generally, vector transport is associated to a retraction, that is, a vector transport can be obtained by the differentiated retraction [4]. Nonetheless, in [26] an intrinsic representation of vector transports on matrix manifolds was developed which does not have an associated retraction, however, they are isometric and cheap.

The vector transport that is used in a Riemannian conjugate gradient method is crucial to the performance of the algorithm as shown by the numerical experiments of [52]. Therefore, our work proposes to avoid the computation of a vector transport and, as consequence, avoid approximations on parallel transport since our optimization approach works over the Euclidean space.

Using the Cayley transform, we develop a feasible scheme that does not use Riemannian optimization directly. As far as we establish an equivalent problem over a vector space, we can use classical optimization methods and this is the main feature of this work.

1.2 Objectives

The investigation presented in this work has the following objectives.

1. To have a better comprehension of some optimization algorithms for Stiefel manifold that are part of the state-of-the-art literature.

2. To show that, under mild assumptions, problem (1.1) over Stiefel manifold can be solved by finding the solution of an equivalent problem over a vector space and, therefore, Riemannian optimization is not used directly.

1.3 Overview

We first review some Riemannian optimization algorithms in order to establish a difference with the framework we are proposing. Cayley Transform is going to be a common concept between Riemannian methods and our proposal. Inspired by Riemannian optimization methods, we will use the Cayley Transform to build an equivalent problem to (1.1) but defined over the space of skew-symmetric matrices, which is a vector space.

Since the equivalent problem can be solved with classical optimization methods, we propose to use nonlinear conjugate gradient and make a comparison with some state-of-the-art Riemannian methods over a selection of objective functions.

The remainder of the work is organized as follows.

Chapter 2 gives a quick review of mathematical analysis concepts, in particular, matrix analysis. Classical and Riemannian conjugate gradient methods are also presented in this chapter. A summary on Riemannian optimization is also given for better understanding of Riemannian conjugate gradient.

In Chapter 3, three state-of-the-art algorithms are presented. The first one is the feasible method of Wen and Yin [48] that introduces the Cayley Transform based retraction inspired by the Crank-Nicholson scheme for differential equations. The second algorithm, due to Dalmau and Oviedo [12], uses another differential equation scheme, the Adams-Moulton scheme; this algorithm uses a SVD in order to maintain feasibility. The third algorithm is a conjugate gradient method developed by Zhu [52] where vector transports are generated by differentiation of Cayley Retraction.

The development of our proposal is explained in Chapter 4 in which we establish an equivalent problem to (1.1) that can be solved over a vector space. The description of algorithms applied in this framework is also presented in this chapter.

Chapter 5 contains results of the numerical experiments and comparison of the performance of the different optimization methods.

A collection of ideas for possible future investigation is presented in Chapter 6. These ideas were motivated by the things we learned trough the development of this work but due to lack of time we were not able to develop them further.

Finally, Chapter 7 contains conclusions regarding the performance of our methods. Mentioned contents show that problem (1.1) can be solved without using of Riemannian optimization methods whilst maintaining a feasible scheme. Our proposal can be seen as a change of variable that allows us to use classical optimization methods.

Chapter 2

Background theory

In this chapter we present some definitions and theorems appearing in the theory of methods for solving (1.1). The main topics comprise matrix calculus, mathematical analysis and Riemannian geometry.

The elements presented in this chapter will give the reader sufficient knowledge to understand the ideas behind the optimization methods presented in this work.

2.1 Matrix Analysis

In this section we review a few basic definitions and concepts that will result useful in the development of this work. Further information on this topics can be found in [22].

Definition 2.1 (Symmetric matrix). *A square matrix $A \in \mathbb{R}^{n\times n}$ is said to be symmetric if $A^\top = A$.*

Definition 2.2 (Skew-symmetric matrix). *A square matrix $A \in \mathbb{R}^{n\times n}$ is said to be skew-symmetric if $A^\top = -A$.*

Definition 2.3 (Trace). *The trace of a square matrix is the sum of its diagonal entries,*

$$\mathrm{Tr}[A] = \sum_{i=1}^{n} a_{ii}, \quad A \in \mathbb{R}^{n\times n}.$$

Theorem 1 (Trace properties). *The trace has the following properties.*

1. $\mathrm{Tr}[kA] = k\mathrm{Tr}[A], \quad A \in \mathbb{R}^{n\times n}, k \in \mathbb{R}.$
2. $\mathrm{Tr}[A+B] = \mathrm{Tr}[A] + \mathrm{Tr}[B], \quad A, B \in \mathbb{R}^{n\times n}.$
3. $\mathrm{Tr}[A] = \mathrm{Tr}[A^\top], \quad A \in \mathbb{R}^{n\times n}.$
4. $\mathrm{Tr}[AB] = \mathrm{Tr}[BA], \quad A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times m}.$

5. $\mathrm{Tr}[A^\top A] = \mathrm{Tr}[AA^\top] = \sum_{i=1}^m \sum_{j=1}^n a_{ij}^2, \quad A \in \mathbb{R}^{m\times n}$.

Definition 2.4 (Euclidean inner product in $\mathbb{R}^{m\times n}$). *Let $A, B \in \mathbb{R}^{m\times n}$. The inner product between A and B is defined as*

$$\langle A, B\rangle := \sum_{i,j} a_{ij} b_{ij} = \mathrm{Tr}[A^\top B].$$

The inner product in $\mathbb{R}^{m\times n}$ induces the Frobenius norm.

Definition 2.5 (Frobenius norm). *The Frobenius norm of $A \in \mathbb{R}^{m\times n}$ is defined as*

$$\|A\|_F := \sqrt{\langle A, A\rangle} = \sqrt{\mathrm{Tr}[A^\top A]} = \sqrt{\sum_{i,j} a_{ij}^2}.$$

Definition 2.6 (Orthogonal complement). *Let $S \subseteq \mathbb{R}^m$ be a subspace of $\mathbb{R}^m$. The orthogonal complement of S is*

$$S^\perp := \{y \in \mathbb{R}^m | \langle y, x\rangle = 0, \forall x \in S\}.$$

Definition 2.7 (Orthogonal Matrix). *The matrix $Q \in \mathbb{R}^{n\times n}$ is orthogonal if its columns are an orthonormal basis of $\mathbb{R}^n$, i.e., Q is orthogonal if, and only if,*

$$Q^\top Q = QQ^\top = I,$$

where I is the n-by-n identity matrix.

Proposition 2.1 (Cayley Transform). *Let $W \in \mathbb{R}^{n\times n}$ be a skew-symmetric matrix, i.e., $W^\top = -W$. Then $I - W$ is nonsingular and the matrix $(I-W)^{-1}(I+W)$ is orthogonal. This is known as the Cayley Transform of W.*

Proof. See Appendix A $\square$

Theorem 2 (Sherman-Morrison-Woodbury Formula). *For $A \in \mathbb{R}^{n\times n}$ and $U, V \in \mathbb{R}^{n\times l}$ we have*

$$(A + UV^\top)^{-1} = A^{-1} - A^{-1}U(I_k + V^\top A^{-1}U)^{-1}V^\top A^{-1}, \tag{2.1}$$

where I_k is the size k identity matrix. Matrices A and $I_k + V^\top A^{-1}U$ are assumed to be nonsingular.

2.2 Mathematical Analysis

Definitions contained in this chapter appear in mathematical analysis and functional analysis. Since they are standard concepts they can be found in books like [8, 32]. These concepts will be useful to understand the convergence properties of the optimization algorithms to be presented later in this document.

Although some of the following definitions are given over $\mathbb{R}$ or $\mathbb{R}^n$ the have natural extensions over $\mathbb{R}^{m\times n}$.

Definition 2.8 (Convergent sequence). *A sequence $\{x_n\} \subset \mathbb{R}$ is said to converge to $x \in \mathbb{R}$, or x is said to be the limit of $\{x_n\}$, if for every $\epsilon > 0$ there exists a natural number $K(\epsilon)$ such that for all $n \geq K(\epsilon)$, the terms x_n satisfy $|x_n - x| \leq \epsilon$.*

If the sequence has a limit, we say that the sequence is convergent; if it has no limit, we say that the sequence is divergent.

Definition 2.9 (Bounded sequence). *A sequence $\{x_n\}$ of real numbers is said to be bounded if there exists a real number $M > 0$ such that $|x_n| \leq M$ for all $n \in \mathbb{N}$.*

Definition 2.10 (Monotone sequence). *Let $\{x_n\}$ a sequence of real numbers. We say that $\{x_n\}$ is increasing if it satisfies the inequelities*

$$x_1 \leq x_2 \leq \cdots \leq x_n \leq x_{n+1} \leq \cdots .$$

We say that $\{x_n\}$ is decreasing if it satisfies the inequalities

$$x_1 \geq x_2 \geq \cdots \geq x_n \geq x_{n+1} \cdots .$$

We say that $\{x_n\}$ is monotone if it is either increasing or decreasing.

Definition 2.11 (Bounded set). *The set $X \subset \mathbb{R}^n$ is a bounded set if there exists a real number $M > 0$ such that $\|x\|_2 \leq M$, for all $x \in X$.*

Definition 2.12 (Closure). *The closure of the set $X \subseteq \mathbb{R}^n$ is defined as*

$$\overline{X} := \{x \in \mathbb{R}^n | N_\epsilon(x) \cap X \neq \emptyset, \forall \epsilon > 0\},$$

where $N_\epsilon = \{s \in \mathbb{R}^n | \|x - s\|_2 < \epsilon\}$.

Definition 2.13 (Closed set). *The set $X \subset \mathbb{R}^n$ is closed if, and only if, $X = \overline{X}$.*

Definition 2.14 (Compact set). *The set $X \subset \mathbb{R}^n$ is compact if, and only if, X is closed and bounded.*

Definition 2.15 (Limit point). *Let $X \subseteq \mathbb{R}^n$ and $x \in X$. We say that x is a limit point of X if said point belongs to the closure of $X - \{x\}$. The set of all limit points of X will be denoted as X'.*

Definition 2.16 (Cauchy sequence, completeness). *A sequence $\{x_n\}$ in a metric space $X = (X, d)$ is said to be Cauchy (or fundamental) if for every $\epsilon > 0$ there is an $N = N(\epsilon)$ such that*

$$d(x_m, x_n) < \epsilon \quad \text{for every } m, n > N.$$

The space X is said to be complete if every Cauchy sequence in X converges (that is, has a limit which is an element of X).

Definition 2.17 (Hilbert Space). *A Hilbert space is a complete inner product space (complete in the metric defined by the inner product).*

Theorem 3 (Riez representation). *Let H be a Hilbert space and H^* its dual space (all the continuous functions from H to $\mathbb{R}$). For any $\phi \in H^*$ there exists a unique $x_0 \in H$ such that $\phi(x) = \langle x, x_0 \rangle$ for all $x \in H$. We say that x_0 represents ϕ.*

2.3 Derivatives

In this work, we consider first order methods for optimization, i.e., methods that take advantage of the information given by the gradient of the objective function $\mathcal{F}$. In this regard, let us present some concepts about differentiable functions. These and more concepts on differential calculus can be studied in mathematical analysis literature such as [21].

Definition 2.18 (Differential). *Let $\mathcal{F} : A \subset \mathbb{R}^{m\times n} \to \mathbb{R}$ a function defined on an open set $A \subset \mathbb{R}^{m\times n}$ and $X \in A'$. Then the function $\mathcal{F}$ is differentiable at X if there exists a matrix $G_X \in \mathbb{R}^{m\times n}$ such that, for all $H \in \mathbb{R}^{m\times n}$,*

$$\lim_{H\to 0} \frac{\mathcal{F}(X+H) - \mathcal{F}(X) - \langle G_X, H\rangle}{\|H\|} = 0. \tag{2.2}$$

Theorem 4. *Let $\mathcal{F} : \mathbb{R}^{m\times n} \to \mathbb{R}$ a differentiable function at $X \in \mathbb{R}^{m\times n}$ and $Z \in \mathbb{R}^{m\times n}$. Then*

$$\mathrm{D}\mathcal{F}(X)[Z] := \lim_{t\to 0} \frac{\mathcal{F}(X+tZ) - \mathcal{F}(X)}{t} = \langle G_X, Z\rangle,$$

where $t \in \mathbb{R}$ and $G_X = \left[\frac{\partial \mathcal{F}(X)}{\partial x_{ij}}\right]$.

2.4 Matrix calculus

In order to derive our optimization method, we will need to compute the derivative of matrix functions with respect to matrices. Theory is clear but actual computing of derivatives can be cumbersome. We next present some rules of matrix calculus that simplify these computations. Two important concepts are the *vec* operator and the Kronecker product that are presented next.

2.4.1 Kronecker product and *vec* operator

The *vec* operator vectorizes a matrix by stacking its columns,

$$vec : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}.$$

For example, if $A \in \mathbb{R}^{m\times n}$ and A_i denotes the i-th column of A, then

$$vec(A) = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}.$$

The Kronecker product of two matrices, $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, is defined as

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & & \ddots & \\ A_{m1}B & A_{m2}B & \cdots & A_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nq}. \tag{2.3}$$

For square matrices A and B of order m and n, respectively, the Kronecker sum is defined as

$$A \oplus B = (A \otimes I_n) + (I_m \otimes B). \tag{2.4}$$

From [17] we have a useful relationship:

$$\text{vec}(AXB) = (B^\top \otimes A)\text{vec}(X). \tag{2.5}$$

Consider again a matrix $A \in \mathbb{R}^{m\times n}$, there is a permutation matrix $T_{m,n} \in \mathbb{R}^{mn\times mn}$ such that

$$T_{m,n}\text{vec}(A) = \text{vec}(A^\top),$$

with the property $T_{m,n}^\top = T_{n,m}$ [17].

2.4.2 Notation for derivatives

Since Riemannian optimization methods consider both the Euclidean gradient and the Riemannian gradient, it is important to establish an agreement on notation in order to avoid any confusion. We also use a derivative with respect to a skew-symmetric matrix in our methods so the functions defined over this space will be denoted by $\mathcal{H}$ to help noticing that the gradient of these functions is also skew-symmetric.

Following notation is adopted in the remainder of this document.

Let us consider a differentiable function $\mathcal{F}$ that assigns to an n-by-p real matrix a real value, i.e.,

$$\mathcal{F} : \mathbb{R}^{n\times p} \to \mathbb{R}.$$

We use the notation of [48] and denote the Euclidean gradient of $\mathcal{F}$ with respect to X as $G := \text{D}\mathcal{F} := \left(\frac{\partial \mathcal{F}(X)}{\partial X_{ij}}\right)$. The Riemannian gradient will be denoted as $\nabla\mathcal{F}$. Remember that differences between these concepts will be clarified in Section 2.6.2.

In the remainder of this document, $W \in \mathbb{R}^{n\times n}$ will denote a n-by-n skew-symmetric matrix, i.e., $W^\top = -W$. We will denote as $\mathcal{H}$ the differentiable function that maps a skew-symmetric matrix to a real value, i.e.,

$$\mathcal{H} : \mathbb{R}^{n\times n} \to \mathbb{R}.$$

Since the set $\{W \in \mathbb{R}^{n\times n} | W^\top = -W\}$ is a vector space, there is no confusion in using $\nabla\mathcal{H}$ to denote the gradient of $\mathcal{H}$ for the Euclidean and Riemannian gradient of $\mathcal{H}$ are the same. Note that $\nabla\mathcal{H}$ is a n-by-n skew-symmetric matrix.

On the other hand, we also consider derivatives of matrices with respect to matrices which is accomplished by vectorizing the matrices. If $\mathcal{U} : \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q} : M \mapsto \mathcal{U}(M)$ then

$$\frac{\mathrm{d}vec(\mathcal{U}(M))}{\mathrm{d}vec(M)}$$

is a Jacobian matrix as we know it, i.e., is the derivative of $vec(\mathcal{U}) : \mathbb{R}^{mn} \to \mathbb{R}^{pq}$. The transpose of this Jacobian matrix will be the gradient of $vec(\mathcal{U})$ and will be denoted as

$$\overset{vec}{\nabla}\,\mathcal{U} := \left(\frac{\mathrm{d}vec(\mathcal{U}(M))}{\mathrm{d}vec(M)}\right)^{\top}.$$

2.4.3 Product rule

For functions of matrices, we will adopt the definition of product rule for derivatives as defined in [17]. Let $\mathcal{U} : \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q}$ and $\mathcal{V} : \mathbb{R}^{m\times n} \to \mathbb{R}^{q\times r}$. The product rule is

$$\frac{\mathrm{d}vec(\mathcal{U}(X)\mathcal{V}(X))}{\mathrm{d}vec(X)} = (\mathcal{V}(X)^{\top} \otimes I_p)\frac{\mathrm{d}vec(\mathcal{U}(X))}{\mathrm{d}vec(X)} + (I_r \otimes \mathcal{U}(X))\frac{\mathrm{d}vec(\mathcal{V}(X))}{\mathrm{d}vec(X)}, \quad (2.6)$$

where I_p and I_r are identity matrices of size p and r, respectively.

2.5 Conjugate Gradient method for unconstrained optimization

Conjugate Gradient method is a powerful tool developed for minimization of convex quadratic functions. However, extensions of the CG method have been developed for general nonlinear functions [37].

In the context of nonlinear conjugate gradient method, the first search direction is selected as the negative gradient and the following directions are linear combinations of the negative gradient and the preceding directions. The coefficient β in this combination is selected according to a conjugacy condition.

As an example, we present a nonlinear CG method due to Fletcher and Reeves [18]. Since optimization is made over the Euclidean space $\mathbb{R}^d$ we use $\nabla f(x_k)$ to denote the Euclidean gradient of the objective function f at x_k.

Algorithm 2.1 Fletcher-Reeves nonlinear CG method

```
Input: $x_0 \in \mathbb{R}^n$.
1: $p_0 \leftarrow -\nabla f(x_0)$, $k \leftarrow 0$.
2: while $\nabla f(x_k) \neq 0$ do
3:     Select $\alpha_k$.
4:     Set $x_{k+1} \leftarrow x_k + \alpha_k p_k$.
5:     Compute
           $\beta_{k+1}^{FR} \leftarrow \frac{\nabla f(x_{k+1})^\top \nabla f(x_{k+1})}{\nabla f(x_k)^\top \nabla f(x_k)}$,    (2.7)
           $p_{k+1} \leftarrow -\nabla f(x_{k+1}) + \beta_{k+1}^{FR} p_k$,
           $k \leftarrow k+1$
6: end while
```

Other selections for the coefficient β have led to different nonlinear CG methods. In subsection 3.3.2 of Chapter 3 a nonmonotone method due to Dai [13] will be presented.

2.6 Riemannian geometry for the Stiefel manifold

Stiefel manifold is a smooth manifold embedded on $\mathbb{R}^{n\times p}$ [4]. In this regard, we discuss in this section the concepts that allow us to apply Riemannian optimization on Stiefel manifold.

Stiefel manifold is defined as all the $n \times p$ matrices, $p \leq n$, on the set

$$\mathrm{St}(n,p) := \{X \in \mathbb{R}^{n\times p} : X^\top X = I_p\}.$$

The Proposition 3.3.3 in [4] shows that the dimension of $\mathrm{St}(n,p)$ is

$$\dim(\mathrm{St}(n,p)) = np - \frac{1}{2}p(p+1).$$

In [4], it is also shown that $\mathrm{St}(n,p)$ is closed and bounded, hence, it is compact. As some examples, when $p = 1$ the Stiefel manifold is the sphere S^{n-1}; for $p = n$ we have the orthogonal group O_n.

2.6.1 Tangent vectors, tangent space and differential of a map

In order to define a search direction in the context of an optimization algorithm, some generalizations have to be made. Similar to the case of regular surfaces, smooth manifolds have a tangent space in each point of the manifold conformed with all the tangent vectors on said point.

Let $\mathcal{M}$ be a smooth manifold. A smooth mapping

$$\begin{aligned}\gamma :&\mathbb{R} \to \mathrm{St}(n,p)\\ &t \mapsto \gamma(t),\end{aligned}$$

is called a curve in $\mathcal{M}$. If $\mathcal{F}$ is a smooth real-valued function on $\mathcal{M}$ then $\mathcal{F} \circ \gamma$ is a smooth function from $\mathbb{R}$ to $\mathbb{R}$ with well-defined classical derivative. Let $\gamma(0) = X \in \mathcal{M}$. Then, a tangent vector to γ at $t = 0$ is the mapping $\dot{\gamma}(0)$, from $\mathfrak{F}_X(\mathcal{M})$ to $\mathbb{R}$, satisfying

$$\dot{\gamma}(0)\mathcal{F} := \frac{\mathrm{d}(\mathcal{F}(\gamma(t)))}{\mathrm{d}t}\Big|_{t=0}, \quad \mathcal{F} \in \mathfrak{F}_X(\mathcal{M}),$$

where $\mathfrak{F}_X(\mathcal{M})$ is the set of smooth real-valued functions defined on a neighborhood of X [4].

The *tangent space* to $\mathcal{M}$ at X, denoted by $T_X\mathcal{M}$ is the set of all tangent vectors to $\mathcal{M}$ at X. The tangent space is indeed a vector space.

Let F be a smooth mapping between two manifolds $\mathcal{M}$ and $\mathcal{N}$. The mapping

$$\begin{aligned}\mathrm{D}F(X) : T_X\mathcal{M} &\to T_{F(X)}\mathcal{N}\\ \xi &\mapsto \mathrm{D}F(X)[\xi]\end{aligned}$$

is a linear mapping called the *differential* of F at X [4].

2.6.2 Riemannian manifolds and Riemannian gradient

A Riemannian manifold is a smooth manifold whose tangent spaces are endowed with a smooth inner product, called Riemannian metric. The Riemannian metric induces a norm on the tangent space and, therefore, we can characterize the steepest increase direction of a scalar field at a point X on the manifold.

Let $\mathcal{M}$ be a smooth manifold. Suppose that $T_X\mathcal{M}$ is endowed with an inner product $\langle\cdot,\cdot\rangle_X$ and a smooth scalar field $\mathcal{F}$ is defined on $\mathcal{M}$. The Riemannian gradient of $\mathcal{F}$ at X is the unique element of $T_X\mathrm{St}(n,p)$ that satisfies [4]

$$\langle\nabla\mathcal{F}(X), \xi\rangle_X = \mathrm{D}\mathcal{F}(X)[\xi], \quad \forall\xi \in T_X\mathrm{St}(n,p), \tag{2.8}$$

The Riemannian gradient is the direction of steepest ascent of $\mathcal{F}$ at X [4].

Suppose that $\mathcal{F}_N$ is a smooth function defined on a manifold $\mathcal{N}$. Let $\mathcal{M}$ be a manifold embedded in $\mathcal{N}$ and $\mathcal{F}_M$ be the restriction of $\mathcal{F}_N$ to $\mathcal{M}$. The gradient of $\mathcal{F}_M$ is equal to the projection of $\mathcal{F}_N$ onto $T_X\mathcal{M}$ [4]

$$\nabla\mathcal{F}_M(X) = P_X(\nabla\mathcal{F}_N(X)). \tag{2.9}$$

2.6.3 Stiefel manifold as a Riemannian manifold

As mentioned at the beginning of Section 2.6, Stiefel manifold is embedded in $\mathbb{R}^{n\times p}$. Moreover, it is a Riemannian manifold. Particular details on its geometry are given next.

For $\mathrm{St}(n,p)$, the tangent space at X is given by [4]

$$\begin{aligned} T_X\mathrm{St}(n,p) &= \{Z\in\mathbb{R}^{n\times p} : X^\top Z + Z^\top X = 0\} && (2.10)\\ &= \{X\Omega + X_\perp K : \Omega^\top = -\Omega, K\in\mathbb{R}^{(n-p)\times p}\}, && (2.11)\end{aligned}$$

where $X_\perp$ is any matrix such that $\mathrm{span}(X_\perp)$ is the orthogonal complement of $\mathrm{span}(X)$. For the tangent space $T_X\mathrm{St}(n,p)$ is natural to define the Riemannian metric as the metric inherited from $\mathbb{R}^{n\times p}$, i.e.,

$$\langle \xi,\eta\rangle_X := \mathrm{Tr}[\xi^\top \eta], \tag{2.12}$$

the so-called Frobenius inner product.

An important element of $T_X\mathrm{St}(n,p)$ is the Riemannian gradient whose computation is described next. From now on, we consider $\mathcal{F}$ to be a smooth real-valued function defined on $\mathbb{R}^{n\times p}$ and, therefore, also defined on $\mathrm{St}(n,p)$. In this regard, the Riemannian gradient of $\mathcal{F}$ can be computed considering Equation (2.9), i.e., projecting the Euclidean gradient, $G := \frac{\partial\mathcal{F}}{\partial X}$, onto $T_X\mathrm{St}(n,p)$:

$$\nabla\mathcal{F}(X) = P_X(G). \tag{2.13}$$

In order to find the projection operator P_X we solve

$$P_X(Z) = \underset{\eta\in T_X\mathrm{St}(n,p)}{\arg\min}\ \|Z-\eta\|_F^2, \tag{2.14}$$

where $X\in\mathrm{St}(n,p)$ and $Z\in\mathbb{R}^{n\times p}$. The Lagrangian can be written as

$$\mathcal{L}(\eta;\Lambda) = \mathrm{Tr}[(Z-\eta)^\top(Z-\eta)] - \mathrm{Tr}[\Lambda(\eta^\top X + X^\top\eta)],$$

the derivative equal to zero yields

$$-2Z + 2\eta - X(\Lambda^\top + \Lambda) = 0,$$

hence

$$\eta = Z + X\frac{\Lambda+\Lambda^\top}{2}.$$

Since $\eta\in T_X\mathrm{St}(n,p)$, we have

$$\left(Z + X\frac{\Lambda+\Lambda^\top}{2}\right)^\top X + X^\top\left(Z + X\frac{\Lambda+\Lambda^\top}{2}\right) = 0,$$

or

$$\Lambda + \Lambda^\top = -(X^\top Z + Z^\top X).$$

Finally, for any $Z \in \mathbb{R}^{n\times p}$ the projection to $T_X\mathrm{St}(n,p)$ is given by

$$P_X(Z) = Z - X\mathrm{sym}(X^\top Z) = (I - XX^\top)Z + X\mathrm{skew}(X^\top Z), \tag{2.15}$$

where $\mathrm{sym}(A) = (A + A^\top)/2$ and $\mathrm{skew}(A) = (A - A^\top)/2$.

Using (2.13) and (2.15), the Riemannian gradient of $\mathcal{F}$ at $X \in \mathrm{St}(n,p)$ is

$$\nabla\mathcal{F}(X) = G - X\mathrm{sym}(X^\top G).$$

2.6.4 Optimization on St(n,p)

Classical line search methods consider a search direction η_k and then decide how far to move along that direction from a given iterate X_k. Next iterate is generated choosing the step length α, a positive scalar, in the following function

$$\psi(\alpha) = X_k + \alpha\eta_k.$$

However, if $X_k \in \mathrm{St}(n,p)$ and $\eta \in T_{X_k}\mathrm{St}(n,p)$ then $\psi(\alpha) \in T_{X_k}\mathrm{St}(n,p)$, since $T_{X_k}\mathrm{St}(n,p)$ is a vector space. In such case, if $\alpha > 0$ then $\psi(\alpha) \notin \mathrm{St}(n,p)$. Therefore, if feasibility is to be maintained in each iterate, it is not adequate to select $X_{k+1} = \psi(\alpha_k)$, for some $\alpha_k > 0$.

In order to have a feasible iterative scheme, a mapping from $T_X\mathrm{St}(n,p)$ onto $\mathrm{St}(n,p)$ is used, namely, a retraction. The definition for general manifold is [4]

Definition 2.19. *A retraction on a manifold $\mathcal{M}$ is a smooth mapping R from the tangent bundle $T\mathcal{M}$ onto $\mathcal{M}$ with the following properties. Let R_X denote the restriction of R to $T_X\mathcal{M}$.*

(i) $R_X(0_X) = X$, where 0_X denotes the zero element of $T_X\mathcal{M}$.

(ii) With the canonical identification $T_{0_X}T_X\mathcal{M} \simeq T_X\mathcal{M}$, R_X satisfies

$$\mathrm{D}R_X(0_X) = \mathrm{id}_{T_X\mathcal{M}},$$

where $\mathrm{id}_{T_X\mathcal{M}}$ denotes the identity mapping on $T_X\mathcal{M}$.

Therefore, an adequate iterative scheme is

$$X_{k+1} = R_{X_k}(\alpha_k\eta_k), \tag{2.16}$$

where $\alpha_k > 0$ and $\eta_k \in T_{X_k}\mathrm{St}(n,p)$.

Example 2.1 (Retraction base on the polar decomposition). *The polar decomposition of $M \in \mathbb{R}^{n\times p}$ is*

$$M = QR,$$

where $Q = M(M^\top M)^{-1/2}$ and $R = (M^\top M)^{1/2}$. We have $Q \in \mathrm{St}(n,p)$ since $Q^\top Q = I_p$.

The retraction based on the polar decomposition is [4]

$$R_X(\eta) = (X+\eta)(I_p + \eta^\top \eta)^{-1/2}, \quad \eta \in T_X\mathrm{St}(n,p). \tag{2.17}$$

Property (i) in definition 2.19 is readily checked since

$$R_X(0_X) = (X + 0_X)(I_p + 0_X^\top 0_X)^{-1/2} = X.$$

Property (ii) is harder to show. Let us begin by computing $\frac{\mathrm{d}vec(M^{1/2})}{\mathrm{d}vec(M)}$, for $M \in \mathbb{R}^{n\times n}$ such that $M^\top = M$. Since $\frac{\mathrm{d}vec(M)}{\mathrm{d}vec(M)} = I_{n^2}$, we have

$$\begin{aligned} I_{n^2} &= \frac{\mathrm{d}vec(M^{1/2}M^{1/2})}{\mathrm{d}vec(M)} \\ &= (M^{1/2}\otimes I_n)\frac{\mathrm{d}vec(M^{1/2})}{\mathrm{d}vec(M)} + (I_n \otimes M^{1/2})\frac{\mathrm{d}vec(M^{1/2})}{\mathrm{d}vec(M)} \\ &= (M^{1/2}\oplus M^{1/2})\frac{\mathrm{d}vec(M^{1/2})}{\mathrm{d}vec(M)}, \end{aligned}$$

hence,

$$\frac{\mathrm{d}vec(M^{1/2})}{\mathrm{d}vec(M)} = (M^{1/2}\oplus M^{1/2})^{-1}. \tag{2.18}$$

Now,

$$\begin{aligned} \frac{\mathrm{d}vec((I_p+\eta^\top\eta)^{1/2})}{\mathrm{d}vec(\eta)} &= \frac{\mathrm{d}vec((I_p+\eta^\top\eta)^{1/2})}{\mathrm{d}vec((I_p+\eta^\top\eta))}\frac{\mathrm{d}vec((I_p+\eta^\top\eta))}{\mathrm{d}vec(\eta)} \\ &= [(\eta^\top\eta)^{1/2}\oplus(\eta^\top\eta)^{1/2}]^{-1}[(\eta^\top\otimes I_p)T_{n,p} + (I_p\otimes\eta^\top)]. \end{aligned} \tag{2.19}$$

We are actually interested in computing $\frac{\mathrm{d}vec((I_p+\eta^\top\eta)^{-1/2})}{\mathrm{d}vec(\eta)}$. Using the product rule and (2.19), we have

$$\begin{aligned} 0_{p^2\times np} &= \frac{\mathrm{d}vec((I_p+\eta^\top\eta)^{1/2}(I_p+\eta^\top\eta)^{-1/2})}{\mathrm{d}vec(\eta)} \\ &= ((I_p+\eta^\top\eta)^{-1/2}\otimes I_p)[(\eta^\top\eta)^{1/2}\oplus(\eta^\top\eta)^{1/2}]^{-1}[(\eta^\top\otimes I_p)T_{n,p}+(I_p\otimes\eta^\top)] \\ &\quad + (I_p\otimes(I_p+\eta^\top\eta)^{1/2})\frac{\mathrm{d}vec((I_p+\eta^\top\eta)^{-1/2})}{\mathrm{d}vec(\eta)}, \end{aligned}$$

therefore,

$$\frac{\mathrm{d}vec((I_p+\eta^\top\eta)^{-1/2})}{\mathrm{d}vec(\eta)} = -[(I_p+\eta^\top\eta)^{-1/2}\otimes(I_p+\eta^\top\eta)^{-1/2}] \\ [(\eta^\top\eta)^{1/2}\oplus(\eta^\top\eta)^{1/2}]^{-1}[(\eta^\top\otimes I_p)T_{n,p}+(I_p\otimes\eta^\top)]. \quad (2.20)$$

Finally, we have

$$\frac{\mathrm{d}vec(R_X(\eta))}{\mathrm{d}vec(\eta)} = ((I_p+\eta^\top\eta)^{1/2}\otimes I_n) - [(I_p+\eta^\top\eta)^{-1/2}\otimes(X+\eta)(I_p+\eta^\top\eta)^{-1/2}] \\ [(\eta^\top\eta)^{1/2}\oplus(\eta^\top\eta)^{1/2}]^{-1}[(\eta^\top\otimes I_p)T_{n,p}+(I_p\otimes\eta^\top)], \quad (2.21)$$

hence,

$$\left.\frac{\mathrm{d}vec(R_X(\eta))}{\mathrm{d}vec(\eta)}\right|_{\eta=0} = (I_p\otimes I_n). \quad (2.22)$$

For a tangent vector $\xi\in T_X\mathrm{St}(n,p)$, we have

$$(I_p\otimes I_n)vec(\xi) = vec(I_n\xi I_p) = vec(\xi),$$

since ξ was arbitrary, this is actually the required identity mapping.

Example 2.2 (Retraction based on the Cayley Transform). *Wen and Yin introduced the Cayley Transform based retraction deduced from the Crank-Nicholson scheme [48], details will be presented in Section 3.1. For now, consider $\eta\in T_X\mathrm{St}(n,p)$ in order to build a skew-symmetric matrix*

$$W := W(\eta) := P\eta X^\top - X\eta^\top P,$$

where $P = I_n - \frac{1}{2}XX^\top$. The retraction based on the Cayley Transform is [48]

$$R_X(\eta) = \left(I_n - \frac{1}{2}W\right)^{-1}\left(I_n + \frac{1}{2}W\right)X.$$

Since $W(0)=0$ we have $R_X(0)=X$, i.e., property (i) in Definition 2.19 is satisfied. Once again, property (ii) takes more effort to be proven.

Let us begin by taking the derivative of W w.r.t. η.

$$\begin{aligned}\frac{\mathrm{d}vec(W)}{\mathrm{d}vec(\eta)} &= \frac{\mathrm{d}vec(P\eta X^\top - X\eta^\top P)}{\mathrm{d}vec(\eta)} \\ &= (I_n\otimes P)(X^\top\otimes I_n) - (I_n\otimes X)(P\otimes I_p)T_{n,p} \\ &= (X\otimes P) - (P\otimes X)T_{n,p}. \end{aligned} \quad (2.23)$$

In order to find the derivative of the inverse matrix we write

$$0 = \frac{\mathrm{d}vec\left(\left(I_n-\frac{1}{2}W\right)^{-1}\left(I_n+\frac{1}{2}W\right)\right)}{\mathrm{d}vec(\eta)}$$

$$
\begin{aligned}
&= \left(\left(I_n - \frac{1}{2}W\right)^{\top} \otimes I_n\right) \frac{\mathrm{d}vec\left(I_n - \frac{1}{2}W\right)^{-1}}{\mathrm{d}vec(\eta)} \\
&\quad - \frac{1}{2}\left(I_n \otimes \left(I_n - \frac{1}{2}W\right)^{-1}\right)[(X \otimes P) - (P \otimes X)T_{n,p}], \qquad (2.24)
\end{aligned}
$$

hence

$$
\begin{aligned}
\frac{\mathrm{d}vec\left(I_n - \frac{1}{2}W\right)^{-1}}{\mathrm{d}vec(\eta)} = \frac{1}{2}\Bigg[&\left(\left(I_n - \frac{1}{2}W\right)^{-\top} X \otimes \left(I_n - \frac{1}{2}W\right)^{-1} P\right) \\
&- \left(\left(I_n - \frac{1}{2}W\right)^{-\top} P \otimes \left(I_n - \frac{1}{2}W\right)^{-1} X\right) T_{n,p}\Bigg]. \qquad (2.25)
\end{aligned}
$$

Now, the product rule yields

$$
\begin{aligned}
\frac{\mathrm{d}vec(R_X(\eta))}{\mathrm{d}vec(\eta)} = \frac{1}{2}\Bigg[&\left(X^{\top}\left(I_n + \frac{1}{2}W\right)^{\top}\left(I_n - \frac{1}{2}W\right)^{-\top} X \otimes \left(I_n - \frac{1}{2}W\right)^{-1} P\right) \\
&- \left(X^{\top}\left(I_n + \frac{1}{2}W\right)^{\top}\left(I_n - \frac{1}{2}W\right)^{-\top} P \otimes \left(I_n - \frac{1}{2}W\right)^{-1} X\right) T_{n,p} \\
&+ \left(I_p \otimes \left(I_n - \frac{1}{2}W\right)^{-1} P\right) - \left(X^{\top} P \otimes \left(I_n - \frac{1}{2}W\right)^{-1} X\right) T_{n,p}\Bigg],
\end{aligned}
$$

so that,

$$
\frac{\mathrm{d}vec(R_X(0))}{\mathrm{d}vec(\eta)} = (I_p \otimes P) - (X^{\top} P \otimes X)T_{n,p}. \qquad (2.26)
$$

For $\xi \in T_X\mathrm{St}(n,p)$ we have

$$
\begin{aligned}
[(I_p \otimes P) - (X^{\top} P \otimes X)T_{n,p}]vec(\xi) &= vec(P\xi I_p - X\xi PX) \\
&= vec\left(\left(I_n - \frac{1}{2}XX^{\top}\right)\xi - X\xi^{\top}\left(I_n - \frac{1}{2}XX^{\top}\right)X\right) \\
&= vec\left(\xi - X\frac{X^{\top}\xi + \xi^{\top}X}{2}\right) \\
&= vec(\xi),
\end{aligned}
$$

where the last equality follows from the fact that ξ was already a tangent vector, and, therefore, $X^{\top}\xi + \xi^{\top}X = 0$. Since ξ is arbitrary, $\frac{\mathrm{d}vec(R_X(0))}{\mathrm{d}vec(\eta)}$ is indeed the identity mapping.

We have seen that proving a mapping to be a retraction can be difficult using only the definition. A simpler way to do that is described next.

Since $\mathrm{St}(n,p)$ is an embedded manifold of a vector space $\mathbb{R}^{n\times p}$, we can construct a retraction by looking for a manifold $\mathcal{N}$ such that $\dim \mathrm{St}(n,p) + \dim \mathcal{N} = \dim \mathbb{R}^{n\times p}$ and a diffeomorphism

$$\begin{aligned} \phi : \mathrm{St}(n,p) \times \mathcal{N} &\to \mathbb{R}^{n\times p}_{*} \\ (M,N) &\mapsto \phi(M,N), \end{aligned}$$

where $\mathbb{R}^{n\times p}_{*}$ is an open subset of $\mathbb{R}^{n\times p}$, with a neutral element $I \in \mathcal{N}$ satisfying

$$\phi(M,I) = M, \quad \forall M \in \mathrm{St}(n,p).$$

Then, apply the following proposition.

Proposition 2.2. *With the notation above, the mapping*

$$R_X(\eta) := \pi_1(\phi^{-1}(X+\eta)),$$

where $\pi_1 : St(n,p) \times \mathcal{N} \to St(n,p)$, $\pi_1(M,N) = M$, is the projection onto the first component, defines a retraction on $St(n,p)$.

Proof. See proposition 4.1.2 in [4] □

According to Proposition 2.2, the following example is aretraction on the Stiefel manifold. Compute the QR decomposition of $X+\eta$ and use the Q factor as the retracted tangent vector, i.e.,

$$R_X(\eta) = \mathrm{qf}(X+\eta).$$

2.6.5 Riemannian conjugate gradient and vector transport

Considering the geometry of $\mathrm{St}(n,p)$, the conjugate gradient method on $\mathbb{R}^n$ has to be adapted in order to be properly defined on $\mathrm{St}(n,p)$.

The first important detail is to notice that the conjugate gradient equation

$$\eta_{k+1} = -\nabla \mathcal{F}(X_{k+1}) + \beta_{k+1}\eta_k \tag{2.27}$$

does not guarantee that η_{k+1} is in $T_{X_{k+1}}\mathrm{St}(n,p)$ in general. Actually, it only happens for $\beta_{k+1} = 0$. In order to assure that η_{k+1} is in the desired tangent space, both terms in the right side of (2.27) should be in $T_{X_{k+1}}\mathrm{St}(n,p)$.

In [15] parallel translation is used to transport η_k from $T_{X_k}\mathrm{St}(n,p)$ to $T_{X_{k+1}}\mathrm{St}(n,p)$ and generate a conjugate gradient algorithm, considering the equation of a geodesic at a point X with direction D. However, parallel translation can be computationally demanding.

In order to reduce the computational cost other mappings between tangent spaces are used. We consider a transport vector as defined in [4],

Definition 2.20. *A vector transport on a manifold $\mathcal{M}$ is a smooth mapping*

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

satisfying the following properties for all $X \in \mathcal{M}$, and $\eta, \xi \in T_X\mathcal{M}$:

(i) (Associated retraction) There exists a retraction R, called the retraction associated with $\mathcal{T}$, such that the following diagram commutes

$$\begin{tikzcd}
(\eta,\xi) \arrow[r, "\mathcal{T}"] \arrow[d] & \mathcal{T}_\eta(\xi) \arrow[d, "\phi"] \\
\eta \arrow[r, "R"] & \phi(\mathcal{T}_\eta(\xi)) = R_X(\eta)
\end{tikzcd}$$

where $\phi(\mathcal{T}_\eta(\xi))$ denotes the foot of the tangent vector $\mathcal{T}_\eta(\xi)$.

(ii) (Consistency) $\mathcal{T}_{0_X}\xi = \xi$ for all $\xi \in T_X\mathcal{M}$.

(iii) (Linearity) $\mathcal{T}_\eta(a\xi + b\zeta) = a\mathcal{T}_\eta(\xi) + b\mathcal{T}_\eta(\zeta)$.

Given a retraction R on $\mathrm{St}(n,p)$ there exists a vector transport associated with the differential of R. According to [4], we have

$$\mathcal{T}_\eta(\xi) := \mathrm{D}R_X(\eta)[\xi] = \frac{\mathrm{d}}{\mathrm{d}t} R_X(\eta + t\xi)\big|_{t=0}, \tag{2.28}$$

where $\eta, \xi \in T_X\mathrm{St}(n,p)$.

Finally, the Riemannian conjugate gradient equation is

$$\eta_{k+1} = -\nabla\mathcal{F}(X_{k+1}) + \beta_{k+1}\mathcal{T}_{\alpha_k\eta_k}(\eta_k). \tag{2.29}$$

Remark. Remeber that for a smooth, mapping, F, between two manifolds $\mathcal{M}$ and $\mathcal{N}$ we have

$$\begin{aligned} \mathrm{D}F(X) :& T_X\mathcal{M} \to T_{F(X)}\mathcal{N} \\ & \xi \mapsto \mathrm{D}F(X)[\xi]. \end{aligned}$$

It is important to notice that in the case of a retraction on Stiefel manifold we have

$$\begin{aligned} R_X(\eta) : T_X\mathrm{St}(n,p) &\to \mathrm{St}(n,p) \\ \eta &\mapsto R_X(\eta), \end{aligned}$$

hence,

$$\begin{aligned} \mathrm{D}R_X(\eta) : T_\eta(T_X\mathrm{St}(n,p)) &\to T_{R_X(\eta)}\mathrm{St}(n,p) \\ \xi &\mapsto \mathrm{D}R_X(\eta)[\xi]. \end{aligned}$$

However, since $T_X\mathrm{St}(n,p)$ is a vector space we have $T_\eta(T_X\mathrm{St}(n,p)) \simeq T_X\mathrm{St}(n,p)$, so that, $\mathrm{D}R_X(\eta)[\xi]$ is indeed the desired transported vector.

Chapter 3

State-of-the-Art

We now present the main ideas on the literature that led us to develop this work.

3.1 A Feasible Method for Optimization with Orthogonality Constraints

In this article, Wen and Yin proposed an iterative scheme based on the Crank-Nicolson method for partial differential equations. Given $X \in \mathrm{St}(n,p)$ a new iterate is generated by

$$Y(t) = X - \frac{t}{2}W(X + Y(t)),$$

so that,

$$Y(t) = \left(I + \frac{t}{2}W\right)^{-1}\left(I - \frac{t}{2}W\right)X. \tag{3.1}$$

If $W = (P_X G)X^\top - X(P_X G)^\top$, where $P_X = (I - \frac{1}{2}XX^\top)$, or $W = GX^\top - XG^\top$ and $G := \mathrm{D}\mathcal{F}(X) = (\frac{\partial \mathcal{F}(X)}{\partial X_{i,j}})$, then $W^\top = -W$ and $(I + \frac{t}{2}W)^{-1}(I - \frac{t}{2}W)$ is orthonormal for any t (this follows from Proposition 2.1).

Let us comment about the two possible choices for W. Definition (2.8) of the Riemannian gradient depends on the definition of the metric. Euclidean metric is an option but, because of the properties of tangent vectors, the canonical metric for Stiefel manifold is sometimes more adequate. For $\xi, \eta \in T_X\mathrm{St}(n,p)$ the canonical inner product is defined as [15]

$$\langle \eta, \xi \rangle_c = \mathrm{Tr}\left[\eta^\top \left(I_n - \frac{1}{2}XX^\top\right)\xi\right]. \tag{3.2}$$

In this regard, the Riemannian gradient of $\mathcal{F}(X)$ under the canonical metric is $\nabla_c \mathcal{F}(X) = G - XG^\top X$ [15]. We have the relation [52]

$$\nabla_c \mathcal{F}(X) = (I_n + XX^\top)\nabla \mathcal{F}(X). \tag{3.3}$$

On the other hand, differentiating both sides of

$$\left(I_n + \frac{t}{2}W\right) Y(t) = \left(I_n - \frac{t}{2}W\right) X$$

with respect to t, we have

$$\frac{1}{2}WY + \left(I_n + \frac{t}{2}W\right) \dot{Y}(t) = -\frac{1}{2}WX,$$

therefore,

$$\dot{Y}(0) = -WX \tag{3.4}$$

Then $Y(t)$ is a smooth curve $Y(t)$ on $\mathrm{St}(n,p)$ with the properties

- $Y(0) = X$.
- $Y(t)^\top Y(t) = X^\top X = I_p$.
- $\dot{Y}(0) = \begin{cases} -\nabla \mathcal{F}(X) & \text{if } W = GX^\top - XG^\top \\ -\nabla_c \mathcal{F}(X) & \text{if } W = (P_X G)X^\top - X(P_X G)^\top \end{cases}.$

In order to state first-order optimality conditions, consider the Lagrangian of problem (1.1),

$$\mathcal{L}(X, \Lambda) = \mathcal{F}(X) - \frac{1}{2}\mathrm{tr}(\Lambda(X^\top X - I)).$$

Lemma 3.1. *Suppose X is a local minimizer of problem (1.1). Then X satisfies the first-order optimality conditions $D_X\mathcal{L}(X,\Lambda) = G - XG^\top X = 0$ and $X^\top X = I$ with the associated Lagrangian multiplier $\Lambda = G^\top X$. Define*

$$\nabla_c \mathcal{F} := G - XG^\top X \text{ and } W := GX^\top - XG^\top.$$

Then $\nabla_c \mathcal{F} = WX$. Moreover, $\nabla_c \mathcal{F} = 0$ if and only if $W = 0$.

Proof. See [48]. □

On the other hand, in Lemma 4, Wen and Yin, gave an efficient way to compute $Y(t)$ when $p \ll n$. They consider the decomposition $W = UV^\top$, where $U = \begin{bmatrix} G & -X \end{bmatrix}$ and $V = \begin{bmatrix} X & G \end{bmatrix}$ are n-by-$2p$ matrices, so that

$$Y(t) = X - tU\left(I + \frac{t}{2}V^\top U\right)^{-1} V^\top X, \tag{3.5}$$

requires only the inversion of a $2p$-by-$2p$ matrix. Last equation is obtained via the Sherman-Morrison-Woodbury formula. However, to avoid lost of feasibility, after a certain number of iterations a modified Gram-Schmidt process is performed since the SMW formula is numerically unstable.

Choosing a good stepsize t_k at iteration k could accelerate the convergence of the algorithm. However, choosing a stepsize by minimizing $\mathcal{F}(Y_k(t))$ along the curve $Y_k(t)$ with respect to t could be computationally expensive and one is usually satisfied with an approximate minimizer such as t_k satisfying the Armijo-Wolfe conditions:

$$\mathcal{F}(Y_k(t_k)) \leq \mathcal{F}(Y_k(0)) + \rho_1 t_k \mathcal{F}'_t(Y_k(0)) \tag{3.6}$$

$$\mathcal{F}'_t(Y_k(t_k)) \geq \rho_2 \mathcal{F}'_t(Y_k(0)), \tag{3.7}$$

where $0 < \rho_1 < \rho_2 < 1$.

The initial guess for the step size is computed according to the Barzilai-Borwein stepsize [9],

$$t_k = \frac{\mathrm{Tr}[S_k^\top S_k]}{|\mathrm{Tr}[S_k^\top Y_k]|},$$

where $S_k = X_{k+1} - X_k$ and $Y_k = \nabla\mathcal{F}(X_{k+1}) - \nabla\mathcal{F}(X_k)$. Note that the elements of two different tangent spaces are being summed without using of a vector transport.

Wen and Yin used the former elements to define the following algorithm.

Algorithm 3.1 A gradient descent method with curvilinear search

```
Input: $X_0 \in \mathrm{St}(n,p)$, $\epsilon \geq 0$ and $0 < \rho_1 < \rho_2 < 1$.
1: while true do
2:     Set $W = GX^\top - XG^\top$
3:     Choose the step size $t_k$. Call line search along the path of $Y_k(t)$ to obtain a
       step size $t_k$ that satisfies the Riemannian Armijo-Wolfe conditions.
4:     $X_{k+1} \leftarrow Y(t_k)$.
5:     if $\|\nabla\mathcal{F}_{k+1}\| \leq \epsilon$ then STOP
6:     else   $k \leftarrow k+1$.
7:     end if
8: end while
```

3.2 Projected nonmonotone search methods for optimization with orthogonality constraints

Similar to Wen and Yin [48], Dalmau and Oviedo [12] developed an optimization approach for Stiefel manifold based on numerical schemes for solving differential equations. They selected the Adams-Moulton scheme and used a projection operator to maintain feasibility. Furthermore, they proposed another method based on a linear combination of descent directions.

First, consider $X \in \mathbb{R}^{n\times p}$ with rank equal to p. The nearest matrix to X on the Stiefel manifold, with respect to the Frobenius norm, is given by

$$\pi(X) = \arg\min_{Q\in\mathrm{St}(n,p)} \|X - Q\|_F. \tag{3.8}$$

We restate here Proposition 1 in [12] regarding the explicit formula for $\pi(X)$. In the next proposition, $I_{n,p}$ denotes a truncated identity matrix.

Proposition 3.1. *Let $X \in \mathbb{R}^{n\times p}$ be a rank p matrix. Then, $\pi(X)$ is well defined. Moreover, if the SVD of X is $X = U\Sigma V^\top$, then $\pi(X) = UI_{n,p}V^\top$.*

Proof. See [35]. □

Now, the new trial point is given by

$$X_{k+1} := Z_k(t) := \pi(Y_k(t)). \tag{3.9}$$

Regarding the definition of $Y_k(t)$ there are two proposals presented in the next subsections.

3.2.1 A scheme based on a linear combination

The first scheme is based on a linear combination of descent directions and is given by

$$Y_k^{CL}(t) := X_k - t(\lambda B_k L + \mu C_k R), \tag{3.10}$$

where $G_k = \mathrm{D}\mathcal{F}(X_k)$, $B_k = G_k L^\top - LG_k^\top$, $C_k = G_k R^\top - RG_k^\top$, $L, R \in \mathbb{R}^{n\times p}$, t is the step size and (λ, μ) are any two scalars satisfying:

$$\lambda\|B_k\|_F^2 + \mu\|C_k\|_F^2 > 0.$$

Lemma 3.2. *Let $Y_k^{CL}(t)$ be defined by Equation (3.10), then $Y_k^{CL}(t)$ is a descent curve at $t = 0$, i.e.,*

$$D\mathcal{F}(X_k)[\dot{Y}_k^{CL}(0)] = -\frac{\lambda}{2}\|B_k\|_F^2 - \frac{\mu}{2}\|C_k\|_F^2 < 0.$$

Proof. See [12]. □

Dalmau and Oviedo [12] use $L = X_k$, $R = X_{k-1}$ and $(\lambda, \mu) = (2/3, 1/3)$ to obtain a convex combination of the gradient in the current iteration $\nabla\mathcal{F}(X_k) = B_k L = B_k X_k$ and a term that approximates the gradient of the previous itertion if t is small. Finally, the updating formula is

$$Y_k^{CL}(t) := X_k - t(\lambda A_k X_k + \mu B_k X_{k-1}),$$

where $A_k = G_k X_k^\top - X_k G_k^\top$ and $B_k = G_k X_{k-1}^\top - X_{k-1} G_k^\top$.

3.2.2 The Adams-Moulton scheme

The second proposal for $Y_k(t)$ derives from the problem

$$\dot{x}(\tau) = f(t, x(\tau)), \quad x(\tau_0) = x_0, \tag{3.11}$$

which can be solved numerically via Adams-Moulton implicit methods. The two-steps Adams-Moulton method yields the following approximate solution [31]

$$x_{k+1} = x_k + \frac{h}{12}(5f(\tau_{k+1}, x_k + 1) + 8f(\tau_k, x_k) - f(\tau_{k-1}, x_{k-1})), \tag{3.12}$$

with $\tau_i = \tau_0 + ih$, $i \in 1, ..., N$ where N is a positive integer.

The updating formula is obtained by considering $f(\tau, x(\tau))$, in Equation (3.11), as $-\nabla\mathcal{F}(X(\tau)) = -A(\tau)X(\tau)$ and adapting formula (3.12) to obtain

$$Y_k^{AM}(t) := X_k - \frac{t}{12}A_k(5Y_k^{AM}(t) + 8X_k - X_{k-1}), \tag{3.13}$$

where $A_k := G_k X_k^\top - X_k G_k^\top$, $G_k := \mathrm{D}\mathcal{F}(X_k)$ and $t > 0$ is the step size.

Equation (3.13) defines an implicit iterative scheme for solving (1.1). However, the following lemma gives an explicit scheme.

Lemma 3.3. *1. Let $W \in \mathbb{R}^{n\times n}$ be any real skew-symmetric matrix, then $Q = I + W$ is a nonsingular matrix.*

2. $Y_k^{AM}(t)$ defined as in Equation (3.13) can be written as

$$Y_k^{AM}(t) = \left(I + \frac{5t}{12}A_k\right)^{-1}\left(X_k - \frac{t}{12}A_k(8X_k - X_{k-1})\right), \tag{3.14}$$

3. and its derivative with respect to t is

$$\dot{Y}_k^{AM}(t) = -\frac{1}{12}\left(I + \frac{5t}{12}A_k\right)^{-1} A_k\left(5Y_k^{AM}(t) + 8X_k - X_{k-1}\right), \tag{3.15}$$

in particular, $\dot{Y}_k^{AM}(0) = -\frac{1}{12}A_k(13X_k - X_{k-1})$.

Proof. See [12]. □

The cost associated to the computation of the inverse matrix on the right side of (3.14) can be reduced if $p < n/2$ using a similar factorization to that of Wen and Yin.

Lemma 3.4. *Let $U = [G_k, X_k] \in \mathbb{R}^{n\times 2p}$ and $V = [X_k, -G_k] \in \mathbb{R}^n \times 2p$. If $I + \frac{5t}{12}V^\top U$ is a nonsingular matrix then (3.14) is equivalent to*

$$Y_k^{AM}(t) = X_k - W(t)(13X_k - X_{k-1}), \tag{3.16}$$

where $W(t) = \frac{t}{12}U\left(I + \frac{5t}{12}V^\top U\right)^{-1}V^\top$.

Proof. See [12]. □

In order to guarantee that a descent direction is being used, the introduction of a parameter β is needed

$$Y_k^{AM2}(t) := X_k - \frac{t}{12} A_k \left(5\beta Y_k^{AM2}(t) + 8X_k - X_{k-1}\right), \tag{3.17}$$

where

$$\beta = \begin{cases} 2\mathrm{Tr}[G_k^\top A_k X_{k-1}]/5\|A_k\|_F^2 & \text{if } \mathrm{D}\mathcal{F}(X_k)[\dot{Y}_k^{AM}(0)] \geq 0 \\ 1 & \text{in other case.} \end{cases} \tag{3.18}$$

3.2.3 Nonmonotone search with Barzilai-Borwein stepsize

Algorithm 2 in [12] uses the Barzilai-Borwein step size [9] to improve its performance. Defining $S_k = X_{k+1} - X_k$ and $R_k = \mathrm{D}\mathcal{F}(X_{k+1}) - \mathrm{D}\mathcal{F}(X_k)$, two step size are proposed

$$\alpha_k^{BB1} = \frac{\|S_k\|_F^2}{\mathrm{Tr}[S_k^\top R_k]} \quad \text{and} \quad \alpha_k^{BB2} = \frac{\mathrm{Tr}[S_k^\top R_k]}{\|R_k\|_F^2}. \tag{3.19}$$

Dalmau and Oviedo also consider a nonmonotone line search due to Zhang and Hager [51] and propose algorithm 3.2.

Algorithm 3.2 Non-monotone linear search algorithm for solve optimization problems on Stiefel manifold

```
Input: $X_0 \in \mathrm{St}(n,p)$, $\epsilon, t > 0$, $0 < t_m \ll t_M$, $\sigma, \eta, \delta \in (0,1)$, $X_{-1} = X_0$, $C_0 = \mathcal{F}(X_0)$, $Q_0 = 1$, $k = 0$.
 1: while $\|\nabla\mathcal{F}(X_k)\|_F > \epsilon$ do
 2:     while $\mathcal{F}(Z_k(t)) > C_k + \sigma t \mathrm{D}\mathcal{F}(X_k)[\dot{Y}_k(0)]$ do
 3:         $t = \delta t$
 4:     end while
 5:     $X_{k+1} = Z_k(t) := \pi(Y_k(t))$, with $Y_k$ as in (3.10) or (3.17).
 6:     Calculate $Q_{k+1} = \eta Q_{k+1}$ and $C_{k+1} = (\eta Q_k C_k + \mathcal{F}(X_{k+1}))/Q_{k+1}$.
 7:     Choose $t = |\alpha_k^{BB1}|$ or well $t = |\alpha_k^{BB2}|$,
 8:     Set $t = \max(\min(t, t_M), t_m)$.
 9:     $k = k + 1$.
10: end while
11: $X^* = X_k$.
```

3.3 A Riemannian conjugate gradient method for optimization on the Stiefel manifold

In subsection 2.6.5 we stated the need for a vector transport in order to define a conjugate gradient method on $\mathrm{St}(n,p)$. Zhu [52] developed two vector transports with desirable qualities for optimization on Stiefel manifold.

Consider the Cayley Transform based retraction (3.1) and notice that for all $\eta \in T_X\mathrm{St}(n,p)$, it holds

$$\eta = W_\eta X,$$

with $W_\eta = P_X\eta X^\top - X\eta^\top P_X$ and $P_X = I - \frac{1}{2}XX^\top$. With this new definition, we restate the Cayley transform retraction

$$R_X(t\eta) = \left(I - \frac{t}{2}W_\eta\right)^{-1}\left(I + \frac{t}{2}W_\eta\right)X. \tag{3.20}$$

Note that $W_{\eta_k} = U_kV_k^\top$, with $U_k = [P_{X_k}\eta_k, X_k]$ and $V_k = [X_k, -P_{X_k}\eta_k]$, hence for $p \ll n$ we have an iterative scheme similar to (3.5),

$$X_{k+1}(\alpha_k) = X_k + \alpha_k U_k\left(I - \frac{\alpha_k}{2}V_k^\top U_k\right)^{-1}V_k^\top X_k. \tag{3.21}$$

3.3.1 Vector transport from differentiated retraction

According to equation (2.28), Zhu differentiated the retraction (3.20) to obtain a the associated vector transport. Consider

$$\left(I - \frac{1}{2}W_\eta - \frac{t}{2}W_\xi\right)R_X(\eta + t\xi) = \left(I + \frac{1}{2}W_\eta + \frac{t}{2}W_\xi\right)X$$

and differentiate both sides, with respect to t, to have

$$-\frac{1}{2}W_\xi R_X(\eta + t\xi) + \left(I_n - \frac{1}{2}W_\eta - \frac{t}{2}W_\xi\right)\frac{\mathrm{d}}{\mathrm{d}t}R_X(\eta + t\xi) = \frac{1}{2}W_\xi X,$$

therefore,

$$\begin{aligned}
\mathcal{T}_\eta^R(\xi) &= \frac{\mathrm{d}}{\mathrm{d}t}R_X(\eta + t\xi)\bigg|_{t=0}\\
&= \frac{1}{2}\left(I_n - \frac{1}{2}W_\eta\right)^{-1}W_\xi(X + R_X(\eta))\\
&= \frac{1}{2}\left(I_n - \frac{1}{2}W_\eta\right)^{-1}W_\xi\left(X + \left(I_n - \frac{1}{2}W_\eta\right)^{-1}\left(I_n + \frac{1}{2}W_\eta\right)X\right)\\
&= \frac{1}{2}\left(I_n - \frac{1}{2}W_\eta\right)^{-1}W_\xi\left(I_n - \frac{1}{2}W_\eta\right)^{-1}\left[\left(I_n - \frac{1}{2}W_\eta\right)X + \left(I_n + \frac{1}{2}W_\eta\right)X\right]
\end{aligned}$$

$$\mathcal{T}_\eta^R(\xi) = \left(I_n - \frac{1}{2}W_\eta\right)^{-1}W_\xi\left(I_n - \frac{1}{2}W_\eta\right)^{-1}X. \tag{3.22}$$

Actually, the desired transport is

$$\mathcal{T}_{\alpha_k\eta_k}^R(\eta_k) = \left(I_n - \frac{\alpha_k}{2}W_{_k}\right)^{-1}W_{\eta_k}\left(I_n - \frac{\alpha_k}{2}W_{\eta_k}\right)^{-1}X_k$$

$$\begin{aligned}
&= \left(I_n - \frac{\alpha_k}{2}W_{_k}\right)^{-1}\left(I_n - \frac{\alpha_k}{2}W_{_k}\right)^{-1}\left(I_n - \frac{\alpha_k}{2}W_{_k}\right)W_{\eta_k}\left(I_n - \frac{\alpha_k}{2}W_{_k}\right)^{-1}X_k \\
&= \left(I_n - \frac{\alpha_k}{2}W_{_k}\right)^{-2}W_{\eta_k}X_k \\
&= \left(I_n - \frac{\alpha_k}{2}W_{_k}\right)^{-2}\eta_k. \qquad (3.23)
\end{aligned}$$

Using (3.23) and (3.21) the desired vector transport is

$$\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k) = U_k\left[M_{k,1} + \frac{\alpha_k}{2}M_{k,2}M_{k,3} + \frac{\alpha_k}{2}\left(I - \frac{\alpha_k}{2}M_{k,2}\right)^{-1}M_{k,2}M_{k,3}\right], \qquad (3.24)$$

where

$$M_{k,1} = V_k^\top X_k, \quad M_{k,2} = V_k^\top U_k, \quad M_{k,3} = \left(I - \frac{\alpha_k}{2}V_k^\top U_k\right)^{-1}V_k^\top X_k.$$

Since $M_{k,1}$, $M_{k,2}$ and $M_{k,3}$ have already been computed to generate X_{k+1}, the computation of (3.24) takes only $4np^2 + O(p^3)$ flops.

Lemma 2 in [52] proves that $\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)$ satisfies the Ring-Wirth nonexpansive condition [39],

$$\langle\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k), \mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\rangle_{R_{X_k}(\alpha_k\eta_k)} \leq \langle\eta_k, \eta_k\rangle_{X_k}, \qquad (3.25)$$

needed to ensure global convergence of the Riemannian conjugate gradient method.

On the other hand, Zhu also differentiated the retraction corresponding to the matrix exponential, i.e.,

$$R_X^{\exp}(t\eta) = e^{tW_\eta}X.$$

In order to obtain a vector transport associated with the Exponential retraction consider Equation (2.28) and write

$$\frac{\mathrm{d}R_X^{\exp}(\eta + t\xi)}{\mathrm{d}t} = \frac{\mathrm{d}e^{W_\eta + tW_\xi}X}{\mathrm{d}t} = e^{W_\eta + tW_\xi}W_\xi X,$$

therefore,

$$\begin{aligned}
\mathcal{T}_\eta\exp(\xi) &= \left.\frac{\mathrm{d}R_X^{\exp}(\eta + t\xi)}{\mathrm{d}t}\right|_{t=0} \\
&= e^{W_\eta}W_\xi X. \qquad (3.26)
\end{aligned}$$

We are interested in

$$\mathcal{T}^{\exp}_{\alpha_k\eta_k}(\eta_k) = e^{\alpha_k W_{\eta_k}}W_{\eta_k}X_k = e^{\alpha_k W_{\eta_k}}\eta_k \qquad (3.27)$$

Vector transport (3.27) requires the computation of a matrix exponential, however, Zhu [52] proposed the use of the first-order diagonal Padé approximant in order to reduce the computational cost. Since said Padé approximant is actually the Cayley transform, the vector transport obtained is

$$\begin{aligned}\mathcal{T}_{\alpha_k\eta_k}(\eta_k) &= \left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)^{-1}\left(I + \frac{\alpha_k}{2}W_{\eta_k}\right)W_{\eta_k}X_k \\ &= \left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)^{-1}W_{\eta_k}\left(I + \frac{\alpha_k}{2}W_{\eta_k}\right)X_k \\ &= \left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)^{-1}W_{\eta_k}\left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)\left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)^{-1}\left(I + \frac{\alpha_k}{2}W_{\eta_k}\right)X_k \\ &= \left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)^{-1}\left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)W_{\eta_k}\left(I - \frac{\alpha_k}{2}W_{\eta_k}\right)^{-1}\left(I + \frac{\alpha_k}{2}W_{\eta_k}\right)X_k \\ &= W_{\eta_k}X_{k+1}. \end{aligned} \tag{3.28}$$

Considering $M_{k,1}$, $M_{k,2}$ and $M_{k,3}$ as defined above, we have

$$\mathcal{T}_{\alpha_k\eta_k}(\eta_k) = U_k(M_{k,1} + \alpha_k M_{k,2}M_{k,3}), \tag{3.29}$$

and the computation takes also $4np^2 + O(p^3)$ flops.

Lemma 3 in [52] states that $\mathcal{T}_{\alpha_k\eta_k}(\eta_k)$ is an isometric transport, that is,

$$\langle \mathcal{T}_{\alpha_k\eta_k}(\eta_k), \mathcal{T}_{\alpha_k\eta_k}(\eta_k)\rangle_{R_{X_k}(\alpha_k\eta_k)} = \langle \eta_k, \eta_k\rangle_{X_k},$$

hence, the Ring-Wirth nonexpansive condition is satisfied.

After the derivation of these vector transports, Zhu discuss the generation of a new class of isometric vector transports using higher order diagonal Padé approximants. Although, this approach could be associated with a high computational cost.

3.3.2 Generalization of Dai's nonmonotone method

The two vector transports of last section can be used to generalize known conjugate gradient methods on $\mathbb{R}^n$. In particular, Zhu chose to work with Dai's nonmonotone method [13].

Consider

$$\beta^D_{k+1} = \frac{\|\nabla\mathcal{F}(X_{k+1})\|^2}{\max\{y_k^\top\eta_k, -\nabla\mathcal{F}(X_k)^\top\eta_k\}}, \tag{3.30}$$

where $y_k = \nabla\mathcal{F}(X_{k+1}) - \nabla\mathcal{F}(X_k)$. Once again, notice that, in the Riemannian framework, y_k needs to compute the difference between two vectors of different tangent spaces. In order to overcome this difficulty, $\mathcal{T}_{\alpha_k\eta_k}(\eta_k)$ is used yielding

$$\beta^D_{k+1} = \frac{\|\nabla\mathcal{F}(X_{k+1})\|^2}{\max\{y_k^{\mathcal{T}}, -\langle\nabla\mathcal{F}(X_k), \eta_k\rangle_{X_k}\}}, \tag{3.31}$$

where $y_k^{\mathcal{T}} = \langle\nabla\mathcal{F}(X_{k+1}), \mathcal{T}_{\alpha_k\eta_k}(\eta_k)\rangle_{X_{k+1}} - \langle\nabla\mathcal{F}(X_k), \eta_k\rangle_{X_k}$.

Another aspect of importance is the generalization of the nonmonotone condition used in Dai's algorithm. Considering the Riemannian iterative scheme, we have [52]

$$\mathcal{F}(R_{X_k}(\alpha_k\eta_k)) \leq \max\{\mathcal{F}(X_k), \cdots, \mathcal{F}(X_{k-\min\{m-1,k\}})\} + \delta\alpha_k\langle\nabla\mathcal{F}(X_k), \eta_k\rangle_{X_k}, \tag{3.32}$$

where m is some positive integer and δ is the constant corresponding to the Armijo-type condition.

Finally, the algorithm of the Riemannian conjugate gradient method is

Algorithm 3.3 A Riemannian nonmonotone CG algorithm on $\mathrm{St}(n,p)$

```
Input: $\epsilon, \delta, \lambda \in (0,1)$, $m \in \mathbb{N}^+$, $\alpha_{\max} > \alpha_0 > \alpha_{\min} > 0$, $X_0 \in \mathrm{St}(n,p)$, $\eta_0 = -\nabla f(X_0)$, $k = 0$.
1: while $\|\nabla\mathcal{F}(X_k)\|_{X_k} > \epsilon$ do
2:     if nonmonotone condition (3.32) is satisfied then
3:         Set $X_{k+1} = R_{X_k}(\alpha_k\eta_k)$
4:     else
5:         Set $\alpha_k = \lambda\alpha_k$ and go to line 2;
6:     end if
7:     Compute $\eta_{k+1} = -\nabla\mathcal{F}(X_{k+1}) + \beta_{k+1}\mathcal{T}_{\alpha_k\eta_k}(\eta_k)$, where $\beta_{k+1} \in [0, \beta^D_{k+1}]$ with $\beta^D_{k+1}$ computed as in (3.31), and $\mathcal{T}_{\alpha_k\eta_k}(\eta_k)$ computed by (3.24) or (3.29) if $p \ll n$.
8:     Update $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$ and set $k \leftarrow k+1$
9: end while
```

Chapter 4

New Optimization Approach

The study of state-of the-art algorithms granted us with the idea of transforming problem (1.1) into an equivalent problem over a vector space and, then, use conjugate gradient method in $\mathbb{R}^d$ to solve it, without need for a vector transport.

In this chapter we explain our proposal for optimization starting by establishing an equivalent problem to (1.1).

4.1 An equivalent problem

Consider the Cayley transform iterative scheme

$$X_{k+1} = (I - W_k)^{-1}(I + W_k)X_k, \tag{4.1}$$

where I is the identity matrix of size n and $W_k^\top = -W_k$. Since

$$\begin{aligned} \det[(I - W_k)^{-1}(I + W_k)] &= \det(I - W_k)^{-1} \det(I + W_k) \\ &= \frac{1}{\det(I - W_k)} \det(I + W_k) \\ &= \frac{1}{\det(I - W_k)} \det(I + W_k)^\top \\ &= \frac{1}{\det(I - W_k)} \det(I - W_k) \\ &= 1, \end{aligned}$$

X_{k+1} is a rotation of X_k. Furthermore, the minimizer X^* is a rotation of the initial iterate X_0, i.e., $X^* = Q^*X$, where Q^* is the composition of all rotation matrix generated trough the iterative process. Hence, algorithms using Cayley Transform to generate iterates over Stiefel manifold are indirectly finding the matrix Q^*. The following theorem guarantees that, given an initial iterate $X_0 \in \mathrm{St}(n, p)$, the minimizer X^* always can be found as a rotation of X_0, if $p < n$ (we are considering $p \ll n$ for many practical cases).

Theorem 5. *Let X and Y be any two elements of St(n,p). If $p < n$ then there exists $Q \in$ St(n,n), with* $\det Q = 1$, *such that $Y = QX$.*

Proof. Choose $B_Y \in \mathbb{R}^{n\times(n-p)}$ such that $\mathcal{Y} = \begin{bmatrix} Y & B_Y \end{bmatrix}$ is an orthonormal basis of $\mathbb{R}^n$, then $\det \mathcal{Y} = \pm 1$. In the same manner, choose $B_X \in \mathbb{R}^{n\times(n-p)}$ such that $\mathcal{X} = \begin{bmatrix} X & B_X \end{bmatrix}$ is an orthonormal basis of $\mathbb{R}^n$ but, also, such that $\det \mathcal{X} = \det \mathcal{Y}$. Note that this selection is always possible since $p < n$, so that, if $\det \mathcal{X}$ has the opposite sign of $\det \mathcal{Y}$ it suffices to change the sign of one of the columns of B_X.

We have

$$\begin{aligned}
\mathcal{Y}\mathcal{X}^\top X &= \begin{bmatrix} Y & B_Y \end{bmatrix} \begin{bmatrix} X & B_X \end{bmatrix}^\top X \\
&= \begin{bmatrix} Y & B_Y \end{bmatrix} \begin{bmatrix} X^\top \\ B_X^\top \end{bmatrix} X \\
&= \begin{bmatrix} Y & B_Y \end{bmatrix} \begin{bmatrix} I_p \\ 0 \end{bmatrix} \\
&= Y.
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\det \mathcal{Y}\mathcal{X}^\top &= \det \mathcal{Y} \det \mathcal{X} \\
&= \det \mathcal{Y} \det \mathcal{Y} \\
&= (\pm 1)^2 \\
&= 1.
\end{aligned}$$

Hence, $Q = \mathcal{Y}\mathcal{X}^\top$ is the desired rotation matrix. □

Moreover, the Cayley transform assures the existence of a skew-symmetric W^* matrix such that $Q^* = (I - W^*)^{-1}(I + W^*)$ if -1 is not an eigenvalue of Q^*.

Regarding the above, we consider a new variable. For a fixed $X \in \mathrm{St}(n,p)$ define

$$Y(W) = (I - W)^{-1}(I + W)X \in \mathrm{St}(n,p), \tag{4.2}$$

for any skew-symmetric matrix W.

Consider $\mathcal{F} : \mathbb{R}^{n\times p} \to \mathbb{R}$. The composite function

$$\begin{aligned}
\mathcal{H}_X : \mathbb{R}^{n\times n} &\to \mathbb{R} \\
W &\mapsto \mathcal{H}_X(W) := \mathcal{F}(Y(W))
\end{aligned}$$

is defined over the space of skew-symmetric matrices but it is actually $\mathcal{F}$ evaluated at feasible points of (1.1).

Our proposal is to solve the next problem:

$$\min_W \mathcal{H}_X(W) \quad \text{s.t.} \quad W^\top = -W \tag{4.3}$$

The main feature of this approach is the optimization over W since the set of skew-symmetric matrices is a vector space. In this regard, there is no need for a vector transport when implementing the conjugate gradient method. Actually, classical optimization methods over $\mathbb{R}^d$ could be used to solve (4.3) without modification. Furthermore, in every iteration we have a skew-symmetric matrix that generates a feasible solution to (1.1), $Y_k = Y(W_k)$, hence, our proposal is a feasible method.

4.1.1 Gradient of composite function $\mathcal{H}_X$

We want to compute the gradient of $\mathcal{H}_X$ in order to be able to use first order optimization methods to solve (4.3). A simple way to find the required derivative is using the chain rule to obtain

$$\frac{\mathrm{d}\mathcal{H}_X(W)}{\mathrm{d}W} = \frac{\mathrm{d}\mathcal{F}(Y)}{\mathrm{d}Y}\frac{\mathrm{d}Y}{\mathrm{d}W}, \tag{4.4}$$

where Y is defined as in (4.2).

Computations can be made using derivative rules from Section 2.4. Nevertheless, there is the need to take the derivative of Y with respect to a skew-symmetric matrix. In order to compute this derivative, we follow the idea presented in [17] for derivatives with respect to symmetric matrices.

We define the *vech* operator for a skew-symmetric matrix

$$vech : \mathbb{R}^{n\times n} \to \mathbb{R}^{n(n-1)/2},$$

as

$$vech(W) = \begin{bmatrix} w_{21} \\ w_{31} \\ \vdots \\ w_{n1} \\ w_{32} \\ w_{42} \\ \vdots \\ w_{n2} \\ \vdots \\ w_{nn-1} \end{bmatrix}$$

so that, the columns of W are stacked in a way that only the elements below the diagonal are conserved.

Since $w_{ij} = -w_{ji}$, we can take derivatives with respect to $vech(W)$, instead of the whole matrix W. Let us begin by defining the derivative of $W \in \mathbb{R}^{n\times n}$ with respect to $vech(W)$ as

$$S_n := \frac{\mathrm{d}vec(W)}{\mathrm{d}vech(W)}, \tag{4.5}$$

where $S_n \in \mathbb{R}^{n^2 \times n(n-1)/2}$ has, in each column, a single 1, a single -1 and zeros. The application of $S_n^\top$ to the vectorization of $A \in \mathbb{R}^{n\times n}$ generates de *vech* form of $A - A^\top$, i.e.,

$$S_n^\top vec(A) = vech(A - A^\top). \tag{4.6}$$

For details on the derivation and action of S_n see Appendix B.

Using notation from Section 2.4, we actually want to calculate

$$\frac{\mathrm{d}\mathcal{H}_X(W)}{\mathrm{d}vech(W)} = \frac{\mathrm{d}\mathcal{F}(Y)}{\mathrm{d}vec(Y)}\frac{\mathrm{d}vec(Y)}{\mathrm{d}vech(W)}. \tag{4.7}$$

Since $\mathrm{d}\mathcal{F}(Y)/\mathrm{d}vec(Y)$ is a Jacobian matrix and, as we will show below, it is actually the Euclidean gradient G of $\mathcal{F}$ in the form $vec(G)^\top$, the following paragraphs will describe only the computation of $\mathrm{d}vec(Y)/\mathrm{d}vech(W)$. Details on the computation of G for some particular objective functions can be found in Appendix B.

Let us begin by calculating the derivative of $(I_n - W)^{-1}$. Using the product rule (2.6), we have

$$\begin{aligned} 0 &= \frac{\mathrm{d}vec(I_n)}{\mathrm{d}vech(W)} \\ &= \frac{\mathrm{d}vec((I_n - W)^{-1}(I_n - W))}{\mathrm{d}vech(W)} \\ &= ((I_n - W)^\top \otimes I_n)\frac{\mathrm{d}vec((I_n - W)^{-1})}{\mathrm{d}vech(W)} \\ &\quad + (I_n \otimes (I_n - W)^{-1})(-S_n), \end{aligned}$$

hence,

$$\begin{aligned} \frac{\mathrm{d}vec((I_n - W)^{-1})}{\mathrm{d}vech(W)} &= ((I_n - W)^{-\top} \otimes I_n)(I_n \otimes (I_n - W)^{-1})S_n \\ &= ((I_n - W)^{-\top} \otimes (I_n - W))S_n. \end{aligned} \tag{4.8}$$

On the other hand, the product rule (2.6) also yields

$$\frac{\mathrm{d}vec((I_n + W)X)}{\mathrm{d}vech(W)} = (X^\top \otimes I_n)S_n. \tag{4.9}$$

Once again we apply the product rule (2.6) and use Equations (4.8) and (4.9) to obtain

$$\begin{aligned} \frac{\mathrm{d}vec(Y(W))}{\mathrm{d}vech(W)} &= \frac{\mathrm{d}vec((I_n - W)^{-1}(I_n + W)X)}{\mathrm{d}vech(W)} \\ &= (X^\top (I_n - W)^\top \otimes I_n)((I_n - W)^{-\top} \otimes (I_n - W)^{-1})S_n \\ &\quad + (I_p \otimes (I_n - W)^{-1})(X^\top \otimes I_n)S_n \end{aligned}$$

$$= ((X+Y)^\top \otimes (I_n - W)^{-1})S_n. \tag{4.10}$$

Now, we can use (4.10) to find the derivative (4.7) in the following manner.

Since $\mathcal{F}$ is a real-valued function, we have

$$\frac{\mathrm{d}\mathcal{F}(Y)}{\mathrm{d}vec(Y)} = \begin{bmatrix} \frac{\partial \mathcal{F}(Y)}{\partial y_{11}} \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{21}} \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{31}} \\ \vdots \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{n1}} \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{12}} \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{22}} \\ \vdots \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{n2}} \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{13}} \\ \vdots \\ \frac{\partial \mathcal{F}(Y)}{\partial y_{np}} \end{bmatrix}^\top = vec(G)^\top \in \mathbb{R}^{1\times np}, \tag{4.11}$$

where G is the Euclidean gradient of $\mathcal{F}$ evaluated at Y. Note that $\frac{\mathrm{d}vec(Y(W))}{\mathrm{d}vech(W)} \in \mathbb{R}^{pn\times n(n-1)/2}$ because $((X+Y)^\top \otimes (I_n - W)^{-1}) \in \mathbb{R}^{pn\times n^2}$ and $S_n \in \mathbb{R}^{n^2\times n(n-1)/2}$. Hence,

$$\frac{\mathrm{d}\mathcal{F}(Y)}{\mathrm{d}vec(Y)}\frac{\mathrm{d}vec(Y)}{\mathrm{d}vech(W)} \in \mathbb{R}^{1\times n(n-1)/2}$$

is well defined.

The desired derivative is

$$\frac{\mathrm{d}\mathcal{H}_X(W)}{\mathrm{d}vech(W)} = \frac{\mathrm{d}\mathcal{F}(Y)}{\mathrm{d}vec(Y)}\frac{\mathrm{d}vec(Y)}{\mathrm{d}vech(W)} = vec(G)^\top((X+Y)^\top \otimes (I_n - W)^{-1})S_n. \tag{4.12}$$

Actually, transposing (4.12) makes easier to wee how to use properties (2.5) and (4.6) in order to avoid Kronecker products and explicit computation of S_n:

$$\begin{aligned} \overset{vec}{\nabla} \mathcal{H}_X(W) &= S_n^\top((X+Y)^\top \otimes (I_n - W)^{-\top})vec(G) \\ &= S_n^\top vec((I_n - W)^{-\top}G(X+Y)^\top) \\ &= vech((I_n - W)^{-\top}G(X+Y)^\top - (X+Y)G^\top(I_n - W)^{-1}), \end{aligned} \tag{4.13}$$

reordering to matrix form we have

$$\nabla\mathcal{H}_x = (I_n - W)^{-\top}G(X+Y)^\top - (X+Y)G^\top(I_n - W)^{-1}. \tag{4.14}$$

4.1.2 Comments on the use of Cayley Transform

We said before that there always exists a skew-symmetric matrix W such that $Q = (I - W)^{-1}(I + W)$ if -1 is not an eigenvalue of Q. This can be seen from

$$\begin{aligned} Q &= (I - W)^{-1}(I + W) \\ (I - W)Q &= I + W \\ Q - I &= (I + Q)W \\ (Q + I)^{-1}(Q - I) &= W, \end{aligned}$$

so that, the existence of W is reserved to the existence of $(Q + I)^{-1}$.

Consider the eigendecomposition $Q = M\Lambda M^{-1}$, we have

$$\begin{aligned} (Q + I)^{-1} &= (M\Lambda M^{-1} + I)^{-1} \\ &= (M(\Lambda + I)M^{-1})^{-1} \\ &= M(\Lambda + I)^{-1}M^{-1}, \end{aligned}$$

so that, if -1 is an element of Λ then $Q + I$ is a singular matrix.

The former discussion implies that $\{(I - W)^{-1}(I + W)X | W^\top = -W, X \in \mathrm{St}(n, p)\}$ is a proper subset of $\mathrm{St}(n, p)$ and, therefore, the composite function $\mathcal{H}_X$ could have left out some minimizer of $\mathcal{F}$. For example, suppose that $Y = -X$ is the minimizer of $\mathcal{F}$, there is no skew-symmetric matrix W such that $-X = (I - W)^{-1}(I + W)X$ since

$$\begin{aligned} -X &= (I - W)^{-1}(I + W)X \\ -(I - W)X &= (I + W)X \\ -X + WX &= X + WX \\ -X &= X, \end{aligned}$$

which is impossible since $X \in \mathrm{St}(n, p)$. Hence, solving (4.3) could mislead to a wrong solution for some X.

On the other hand, we have

$$\begin{aligned} Q &= (I - W)^{-1}(I + W) \\ &= V(I - D)^{-1}V^{-1}V(I + D)V^{-1} \\ &= V \begin{pmatrix} \frac{1+d_1}{1-d_1} & & \\ & \ddots & \\ & & \frac{1+d_n}{1-d_n}, \end{pmatrix} V^{-1} \end{aligned}$$

where VDV^{-1} is the eigendecomposition of W. The last equality implies that if the module of some eigenvalue d_i tends to ∞ then Q will have some eigenvalue equal to -1. Actually, eigenvalues of W are 0 or purely imaginary, that is, Q will have a pair of eigenvalues equal to -1. In this regard, we have $\lim_{a\to\pm\infty}(I-aW)^{-1}(I+aW) = -I$, that is, $\lim_{a\to\pm\infty} Y(aW) = -X$. Hence, theoretically, we can approximate to $-X$ with sufficiently large W. In general, the case when $w_{ij}\to\infty$, and $w_{ji}\to-\infty$ for only one, or a few indexes, imply some eigenvalues of Q approaching to -1. In practice, we must be aware of numerical limitations.

Riemannian methods, such as the Riemannian conjugate of Section 3.3 or the gradient descent method in Section 3.1, do not suffer from the problem we are describing because of the use of composite rotations. Hence, if the minimizer X^* is a rotation of initial iterate X_0 with eigenvalues equal to -1, the Riemannian methods are indirectly decomposing said rotation in many factors without any eigenvalues equal to -1. Therefore, we will use composite rotations in order to avoid the possibility of X^* not being reachable as from our first iterate with only one rotation with eigenvalues different to -1. Let us clarify with the following example.

Example 4.1 (Optimization on St(n,p)). *Let $X\in\mathbb{R}^2$ be a fixed element of the sphere S^1, i.e., $X\in\mathrm{St}(n,p)$. Then, an optimization problem is defined as*

$$\min_{Y\in\mathbb{R}^2}\mathcal{F}(Y) = \|Y+X\|_F^2 \quad s.t. \quad Y^\top Y = 1. \tag{4.15}$$

It is clear that the solution of Problem (4.15) is $Y^* = -X$. Considering our proposal, we proceed to parametrize Y by taking $Y = Y(W) = (I-W)^{-1}(I+W)X$ and write the following problem

$$\min_{W\in\mathbb{R}^{2\times2}}\mathcal{H}_X(W) = \|Y(W)+X\|_F^2 \quad s.t. \quad W^\top - W. \tag{4.16}$$

However, we have shown that there is no skew-symmetric matrix such that $Y(W) = -X$. Hence, if we keep X fixed and try to solve Problem (4.16) in order to find a solution for (4.15) we may not be able to reach the desired solution. Furthermore, Problem (4.16) does not have a solution in the space of skew-symmetric matrices as shown next.

Consider

$$\begin{aligned} Y(W) &= (I-W)^{-1}(I+W)X \\ &= \begin{pmatrix}1 & w\\ -w & 1\end{pmatrix}\begin{pmatrix}1 & -w\\ w & 1\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} \\ &= \frac{1}{1+w^2}\begin{pmatrix}1 & -w\\ w & 1\end{pmatrix}\begin{pmatrix}x_1 - wx_2\\ wx_1 + x_2\end{pmatrix} \\ &= \frac{1}{1+w^2}\begin{pmatrix}x_1 - wx_2 - w(wx_1+x_2)\\ w(x_1-wx_2)+wx_1+x_2\end{pmatrix} \\ &= \frac{1}{1+w^2}\begin{pmatrix}(1-w^2)x_1 - 2wx_2\\ 2wx_1 + (1-w^2)x_2\end{pmatrix}. \end{aligned}$$

Let $\hat{Y}(w)$ be $Y(W)$ expressed as a function of one real parameterw, that is,

$$\hat{Y}(w) = \begin{pmatrix} \frac{1-w^2}{1+w^2}x_1 - 2\frac{w}{1+w^2}x_2 \\ 2\frac{w}{1+w^2}x_1 + \frac{1-w^2}{1+w^2}x_2 \end{pmatrix}.$$

Hence,

$$\lim_{w\to\pm\infty} \hat{Y}(w) = \begin{pmatrix} -1\cdot x_1 - 2\cdot 0\cdot x_2 \\ 2\cdot 0\cdot x_1 - 1\cdot x_2 \end{pmatrix} = -\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = -X. \tag{4.17}$$

Equation (4.17) shows that as w tends to infinity a better solution can be found and therefore, in theory, an iterative optimization process could get arbitrarily close to the solution without reaching it.

On the other hand, it is clear that

$$-\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \cos\pi & -\sin\pi \\ \sin\pi & \cos\pi \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

so that, the required rotation matrix has eigenvalues $\lambda_{1,2} = -1$, as expected. Moreover, we have a rotation of angle π over the plane which is decomposable into two rotations of angle $\pi/2$, i.e.,

$$\begin{pmatrix} \cos\pi & -\sin\pi \\ \sin\pi & \cos\pi \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{2} & -\sin\frac{\pi}{2} \\ \sin\frac{\pi}{2} & \cos\frac{\pi}{2} \end{pmatrix}^2,$$

where each $\pi/2$ rotation has complex conjugate eigenvalues $\lambda = \pm i$, that is, different from -1. Note that this is not the only possible decomposition since choosing $0 < \theta < \pi$ and composing the two rotations, corresponding to angles θ and $\pi - \theta$, leads to the same result.

Hence, in order to solve (4.15) trough our proposal, we can adopt one, or two, of the following strategies

1. *Use composite rotations.*

2. *Choose $Y(W) = (I - W)^{-1}(I + W)Z$, where $Z \in \mathrm{St}(2, 1)$ and $Z \neq X$.*

Considering the above, implementation of our method will include a parameter k_r, a positive integer, indicating that after every k_r iterations, $Y_k = (I - W_k)^{-1}(I + W_k)X$ is assigned to X so that we can benefit from composite rotations.

4.1.3 Equivalence of Problems (1.1) and (4.3)

As shown before, Cayley Transform is unable to generate all rotation matrices and, in general, is impossible to know beforehand if, given an initial guess $X_0 \in \mathrm{St}(n, p)$, $\mathcal{F}$ has a minimizer of the form $Y^* = QX_0$, where Q does not have any eigenvalue equal to -1. Moreover, it is impossible to know if $\mathcal{H}_X(W)$ has a minimizer.

However, if $\mathcal{H}_X$ has a local minimizer W^*, for a given $X \in \mathrm{St}(n,p)$, then $Y^* = (I - W^*)^{-1}(I + W^*)X$ must be a local minimizer of $\mathcal{F}$. In this regard, a solution of (4.3) generates a solution of (1.1) and therefore, we have equivalent problems. We next develop an argument in order to establish a theorem that formalizes this equivalence.

First note that $Y := Y(W) := (I - W)^{-1}(I + W)X$ is a continuous mapping from the space of skew-symmetric matrices onto Stiefel manifold. This allows us to affirm that given $\epsilon > 0$ there exists $\delta(\epsilon) > 0$ such that for all V satisfying $\|V - W\|_F < \delta$ we then have $\|Y(V) - Y(W)\|_F < \epsilon$. This argument is enough to prove that if $Y(W)$ is a minimizer of $\mathcal{F}$ then W has to be a minimizer of $\mathcal{H}_X$. Unfortunately, the converse requires more effort to be proven.

For our purposes, consider $\mathcal{X} \in \mathrm{St}(n,n)$ and define

$$\mathcal{Y} := \mathcal{Y}(W) = (I - W)^{-1}(I + W)\mathcal{X}.$$

We can apply the Inverse Function Theorem to this mapping in order to obtain an inverse continuous function.

Theorem 6 (Inverse Function Theorems for Manifolds). [33] *Suppose $\mathcal{M}$ and $\mathcal{N}$ are smooth manifolds, and $F : \mathcal{M} \to \mathcal{N}$ is a smooth map. If $p \in \mathcal{M}$ is point such that the differential of F, $\mathrm{d}F_p$, is invertible, then there are connected neighborhoods U of p and V of $F(p)$ such that $F|_U : U \to V$ is a diffeomorphism.*

Corollary 4.1. *Let $W \in \mathbb{R}^{n\times n}$ a skew-symmetric matrix and $\mathcal{X} \in \mathrm{St}(n,n)$. Then, there is an open neighborhood U of W such that the mapping*

$$\begin{aligned} \mathcal{Y} : \mathbb{R}^{n\times n} &\to \mathrm{St}(n,n) \\ W &\mapsto \mathcal{Y}(W) = (I - W)^{-1}(I + W)\mathcal{X} \end{aligned}$$

restricted to U is a diffeomorphism.

Proof. The differential of $\mathcal{Y}$ at W is given by (see Equation (4.10))

$$d\mathcal{Y}_W = ((\mathcal{X} + \mathcal{Y})^\top \otimes (I - W)^{-1})S_n,$$

which maps skew-symmetric matrices, in their *vech* form, to vectorized tangent vectors in $T_{\mathcal{X}}\mathrm{St}(n,n)$. Explicitly, we have

$$vec(\eta) = ((\mathcal{X} + \mathcal{Y})^\top \otimes (I - W)^{-1})S_n vech(V)$$

and

$$\frac{1}{2}S_n^\top((\mathcal{X} + \mathcal{Y})^{-\top} \otimes (I - W))vec(\eta) = vech(V),$$

hence, $d\mathcal{Y}_W$ is a linear isomorphism and by Theorem 6, the result follows. □

On the other hand, the exponential of a skew-symmetric matrix is an orthogonal matrix and this mapping is surjective [11]. Hence, given $\epsilon > 0$, continuity of the exponential mapping yields the existence of $\delta = \delta(\epsilon) > 0$ such that if $\|S\|_F < \delta$ then

$$\|e^S Y - Y\|_F \leq \|e^S \begin{bmatrix} Y & Y^\perp \end{bmatrix} - \begin{bmatrix} Y & Y^\perp \end{bmatrix}\|_F < \epsilon,$$

where $S^\top = -S$. Note that surjectivity of the exponential mapping makes $e^S Y$ a parametrization of Stiefel manifold, in particular, a parametrization of a neighborhood of Y.

We are ready to present the main result in this section. To simplify notation we will denote the ball of radius $\gamma > 0$ centered at a matrix M as $B_M(\gamma) := \{N : \|N - M\|_F < \gamma\}$.

Theorem 7. *Let $W \in \mathbb{R}^{n\times n}$ be a skew-symmetric matrix and $X \in \mathrm{St}(n,p)$. Let $Y = (I - W)^{-1}(I + W)X \in \mathrm{St}(n,p)$. Then Y is a local minimizer of $\mathcal{F}$ if, and only if, W is a local minimizer of $\mathcal{H}_X$.*

Proof. ($\Rightarrow$) Let $Y(W)$ be a local minimizer of $\mathcal{F}$, that is, there exists $\epsilon > 0$ such that $\mathcal{F}(Y) < \mathcal{F}(Z)$ for any $Z \in B_{Y(W)}(\epsilon)$. Suppose that W is not a minimizer of $\mathcal{H}_X$ so that any neighborhood of W contains a matrix V such that $\mathcal{H}_X(V) < \mathcal{H}_X(W)$. Consider $\delta = \delta(\epsilon) > 0$ from the definition of continuity of Y and $V \in B_W(\delta)$ such that $\mathcal{H}_X(V) < \mathcal{H}_X(W)$. Let $L = (I - W)^{-1}(I + V)X$, we have

$$\mathcal{F}(L) = \mathcal{H}_X(V) < \mathcal{H}_X(W) = \mathcal{F}(Y(W)),$$

where $L \in B_{Y(W)}(\epsilon)$ by continuity. This implies that $Y(W)$ is not a local minimizer, which is a contradiction.

($\Leftarrow$) We consider $\mathcal{X} = \begin{bmatrix} X & X^\perp \end{bmatrix}$ in order to define the mapping $\mathcal{Y}(W) = (I - W)^{-1}(I + W)\mathcal{X} \in \mathrm{St}(n,n)$. Now, there exists a neighborhood U of W where the inverse of $\mathcal{Y}$ exists and it is continuous, by Corollary 4.1. Let $\gamma > 0$ be such that $B_W(\gamma) \subset U$ and W is a minimizer of $\mathcal{H}_X$ in $B_W(\gamma)$. In order to build a contradiction, suppose that $Y(W)$ is not a minimizer so that every neighborhood of $Y(W)$ contains an element $Z \in \mathrm{St}(n,p)$ such that $\mathcal{F}(Z) < \mathcal{F}(Y(W))$.

Existence and continuity of the inverse of $\mathcal{Y}$ yields the existence of $\epsilon = \epsilon(\gamma) > 0$ such that if $\|\mathcal{Z} - \mathcal{Y}(W)\|_F < \epsilon$ then there exits a $V \in B_W(\gamma)$ such that $\mathcal{Z} = \mathcal{Y}(V)$.

Hence, given $\epsilon > 0$ there exists $\delta = \delta(\epsilon) > 0$ such that $\|S\|_F < \delta$ implies $\|e^S \begin{bmatrix} Y & Y^\perp \end{bmatrix} - \begin{bmatrix} Y & Y^\perp \end{bmatrix}\|_F < \epsilon$. Since Y is not a local minimizer and without loss of generality, suppose that $\mathcal{F}(e^S Y) < \mathcal{F}(Y)$. Last two arguments yield the existence of $V \in B_W(\gamma)$ such that

$$e^S \begin{bmatrix} Y & Y^\perp \end{bmatrix} = (I - V)^{-1}(I + V)\mathcal{X},$$

that is,

$$e^S Y = (I - V)^{-1}(I + V)X.$$

We have

$$\mathcal{H}_X(V) = \mathcal{F}(e^S Y) < \mathcal{F}(Y) = \mathcal{H}_X(W),$$

which is a contradiction because $V \in B_W(\gamma)$ and W is local minimizer.

□

4.2 Transportless conjugate gradient on Stiefel manifold

As said before, in order to find a solution to (1.1) we solve (4.3). This problem is defined over a linear space and can be solved using classical (non Riemannian) conjugate gradient methods. This is the reason why we call it *transportless conjugate gradient.*

In order to have a comparison with Zhu's method [52], we consider the nonmonotone conjugate gradient method of Dai [13].

Input of the algorithm requires an element $X \in \mathrm{St}(n,p)$. We consider $W_0 = 0$ so that $Y(W_0) = X$ and our algorithm starts at $\mathcal{H}_X(W_0) = \mathcal{F}(X)$, hence, comparison with Riemannian methods is transparent considering $X_0 = X$ as starting point of these methods.

Conjugacy condition is given by Dai's beta defined as

$$\beta^D_{k+1} = \frac{\|\nabla\mathcal{H}_X(W_{k+1})\|^2}{\max\{\langle y_k, \eta_k\rangle, \langle -\nabla\mathcal{H}_X(W_k), \eta_k\rangle\}}, \tag{4.18}$$

where $y_k = \nabla\mathcal{H}_X(W_{k+1}) - \nabla\mathcal{H}_X(W_k)$ and $\eta_k = -\nabla\mathcal{H}_X(W_k) + \beta^D_k \eta_{k-1}$ is the conjugate direction. We use the following algorithm to solve (4.3).

Algorithm 4.1 Dai's nonmonotone CG algorithm in the context of (4.3)

```
Input: $\epsilon, \delta, \lambda \in (0,1)$, $m \in \mathbb{N}^+$, $\alpha_{\max} > \alpha_0 > \alpha_{\min} > 0$, $X \in \mathrm{St}(n,p)$, $W_0 = 0$,
    $\eta_0 = -\nabla\mathcal{H}_X(W_0)$, $k = 0$.
1: while $\|\nabla\mathcal{H}_X(W_k)\|_F > \epsilon$ do
2:     if $\mathcal{H}_X(W_k + \alpha_k\eta_k) \leq \max\{\mathcal{H}_X(W_k), \cdots, \mathcal{H}_X(W_{k-\min\{m-1,k\}})\} + \delta\alpha_k\langle\nabla\mathcal{H}_X(W_k), \eta_k\rangle$ then
3:         Set $W_{k+1} = W_k + \alpha_k\eta_k$
4:     else
5:         Set $\alpha_k = \lambda\alpha_k$ and go to line 2;
6:     end if
7:     Compute $\eta_{k+1} = -\nabla\mathcal{H}_X(W_{k+1}) + \beta_{k+1}\eta_k$, where $\beta_{k+1} \in [0, \beta^D_{k+1}]$ with $\beta^D_{k+1}$
       computed as in (4.18).
8:     Update $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$ and set $k \leftarrow k+1$
9: end while
```

Following the implementation of Zhu [52] we compute the beta of Fletcher-Reeves (2.7) at every iteration and select β_{k+1} as the minimum between β_{k+1}^{FR} and β_{k+1}^{D}. The initial guess for the step-size α_k^0 is computed according to the Barzilai-Borwein step-size [9],

$$\alpha_k^0 = \frac{\mathrm{Tr}[s_k^\top s_k]}{|\mathrm{Tr}[s_k^\top y_k]|}, \tag{4.19}$$

where $s_k = W_{k+1} - W_k$ and $y_k \nabla \mathcal{H}_X(W_{k+1}) - \nabla \mathcal{H}_X(W_k)$.

Although we have said that there is no need for a vector transport using this method, there are some disadvantages that have to be commented. First, we consider the case $p \ll n$ as in [52] and, as a consequence, our method will require a greater computational effort to perform inner products than Riemmanian methods. Our method needs to compute inner products between two n-by-n matrices whilst Riemannian methods perform inner products between n-by-p matrices.

Actually, in order to use the SMW formula (2.1) to compute the inverse of $I - W$ we have to save in memory two n-by-$2p$ matrices corresponding to the factorization of the gradient $\nabla \mathcal{H}_X(W_k) = U_k V_k^\top$, for each iteration k. Even if we were willing to save all these matrices, the application of the SMW formula (2.1) would imply a recursive implementation increasing its complexity as k increases itself.

4.2.1 Transportless Conjugate Gradient (TCG)

TCG stands for *Transportless Conjugate Gradient* and we will use this name to refer to the implementation of Algorithm 4.1 where no decomposition of W is considered when computing $(I - W)^{-1}$.

Once W_k is computed, there are two instances where $(I - W_k)^{-1}$ is needed. The first one appears in

$$Y_k = (I - W_k)^{-1}(I + W_k)X, \tag{4.20}$$

and the second one appears in the gradient of $\nabla \mathcal{H}_X(W_k)$, explicitly,

$$\nabla \mathcal{H}_X(W_k) = (I - W_k)^{-\top} G_k (X + Y_k)^\top - (X + Y_k) G_k^\top (I - W_k)^{-1}. \tag{4.21}$$

Note that actually computing $(I - W_k)^{-1}$ would be more complex than solving two linear systems, namely,

$$(I - W_k) Z = (I + W_k) X, \quad \text{for Z}, \tag{4.22}$$

and

$$(I - W_k)^\top \hat{Z} = G_k, \quad \text{for } \hat{Z}. \tag{4.23}$$

Hence, we propose to compute the LU decomposition of $I - W_k$ and use this decomposition to solve systems (4.22) and (4.23). However, this approach obviously has a high computational cost and the backtracking procedure could represent an increment on this cost.

Regarding inner products, Definition 2.4 implies that $\langle A, B\rangle = vec(A)^\top vec(B) = \sum_i A_i^\top B_i$ where A_i and B_i are the i-th column of A and B, respectively. So that, when computing $\langle A, B\rangle = \mathrm{Tr}[A^\top B]$, we do not perform the matrix multiplication $A^\top B$ and then compute its trace. We perform a sum of inner products $\sum_i A_i^\top B_i$, which is faster.

When computing $\|\nabla\mathcal{H}_X(W_k)\|_F^2$ the properties of Theorem 1 yield

$$\begin{aligned}\mathrm{Tr}[\nabla\mathcal{H}_X(W_k)^\top\nabla\mathcal{H}_X(W_k)] &= \mathrm{Tr}[V_k U_k^\top U_k V_k^\top] \\ &= \mathrm{Tr}[U_k^\top U_k V_k^\top V_k] \\ &= \langle U_k^\top U_k, V_k^\top V_k\rangle, \end{aligned} \tag{4.24}$$

where $U_k = \begin{bmatrix}(I-W_k)^{-\top}G_k & -(X+Y_k)\end{bmatrix}$ and $V_k = \begin{bmatrix}(X+Y_k) & (I-W_k)^{-\top}G_k\end{bmatrix}$. Now, computing $\sum_i \nabla\mathcal{H}_X(W_k)_i^\top\nabla\mathcal{H}_X(W_k)_i$ requires to perform n dot products in $\mathbb{R}^n$. However, with $U_k, V_k \in \mathbb{R}^{n\times 2p}$ the computation of $U_k^\top U_k$ requires $(2p)^2 = 4p^2$ dot products in $\mathbb{R}^n$, same as the computation of $V_k^\top V_k$. Hence, if (4.24) is computed as $\sum_i [U_k^\top U_k]_i^\top [V_k^\top V_k]_i$ another p dot products in $\mathbb{R}^p$ have to be computed. Therefore, if p satisfies $8p^2 + p < n$, is more efficient to compute $\|\nabla\mathcal{H}_X(W_k)\|_F^2$ by $\sum_i [U_k^\top U_k]_i^\top [V_k^\top V_k]_i$. When $8p^2 + p \geq n$ we consider $\sum_i \nabla\mathcal{H}_X(W_k)_i^\top\nabla\mathcal{H}_X(W_k)_i$.

Finally, every $k_r = 10$ iterations we will make $X = Y_k$, $W_k = 0$ and $\beta_{k+1} = 0$ and restart the algorithm. We can do this for three reasons. First one is to avoid the possibility where the minimizer can not be reached by only one rotation of X generated by the Cayley Transform. Second reason is that the conjugate directions are known to lose their properties after some iterations and it is a good idea to restart the conjugate gradient method. The third and main reason is to benefit from composite rotations because we have notice that this is one of the features that makes Wen and Yin's method [48] so effective.

4.3 A Steepest Descent Method

Since computational cost of TCG algorithm is high due to lack of an efficient way to compute $(I - W)^{-1}$, we consider a steepest descent method for only one n-by-n matrix has to be stored in each iteration in order to have an efficient way to compute $(I - W)^{-1}$.

The iterative scheme is

$$W_{k+1} = W_k - \alpha_k\nabla\mathcal{H}_X(W_k), \tag{4.25}$$

where the gradient can be decomposed as

$$\begin{aligned}\nabla\mathcal{H}_X(W_k) &= (I-W_k)^{-\top}G_k(X+Y_k)^\top - (X+Y_k)G_k^\top(I-W_k)^{-1} &(4.26)\\ &= \begin{bmatrix}(I-W_k)^{-\top}G_k & -(X+Y_k)\end{bmatrix}\begin{bmatrix}(X+Y_k) & (I-W_k) & (I-W_k)^{-\top}G_k\end{bmatrix}^\top &(4.27)\end{aligned}$$

$$= U_k V_k^\top, \tag{4.28}$$

where $Y_k := (I - W_k)^{-1}(I + W_k)X$. Define $C_k := (I - W_k)^{-1}$, we have

$$\begin{aligned}(I - W_{k+1})^{-1} &= (I - (W_k - \alpha_k U_k V_k^\top))^{-1} \\ &= C_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top C_k \end{aligned} \tag{4.29}$$

where the last inequality is obtained using the SMW formula (2.1). This means that $(I - W_{k+1})^{-1}$ can be efficiently computed if $(I - W_k)^{-1}$ is available in memory. Moreover, considering $W_0 = 0$ we have $(I - W_0)^{-1} = I$.

The iterative scheme is as follows (see Appendix C for details)

$$Y_{k+1} = Y_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top (X + Y_k). \tag{4.30}$$

The following algorithm is proposed.

Algorithm 4.2 Gradient descent method for (4.3)

```
Input: $\epsilon, \delta, \lambda \in (0,1)$, $\alpha > 0$, $X \in \mathrm{St}(n,p)$, $W_0 = 0$, $\eta_0 = -\nabla\mathcal{H}_X(W_0)$, $k = 0$.
 1: while $\|\nabla\mathcal{H}_X(W_k)\|_F > \epsilon$ do
 2:     $\alpha_k = \alpha$
 3:     if $\mathcal{H}_X(W_k + \alpha_k\eta_k) \le \mathcal{H}_X(W_k) + \delta\alpha_k\langle\nabla\mathcal{H}_X(W_k), \eta_k\rangle$ then
 4:         Set $W_{k+1} = W_k + \alpha_k\eta_k$
 5:     else
 6:         Set $\alpha_k = \lambda\alpha_k$ and go to line 2;
 7:     end if
 8:     $\eta_{k+1} = -\nabla\mathcal{H}_X(W_{k+1})$.
 9:     Set $k \leftarrow k + 1$
10: end while
```

For this algorithm, we also consider the Barzilai-Borwein step-size compute as in Equation (4.19) as initial guess for the backtracking procedure as acceleration strategy. As well, the implementation will consider restarting the algorithm after every $k_r = 10$ iterations, by taking $X = Y_k$, $W_k = 0$ and $C_k = I$.

Once again, $\|\nabla\mathcal{H}_X(W_k)\|_F^2$ can be computed considering Equation (4.24), when p is sufficiently small, in order to reduce computational time.

4.4 Convergence analysis

Convergence analysis for both unconstrained optimization methods is available in literature. In order to make this document self-contained we reproduce the convergence analysis in [13] for Algorithm 4.1. For Algorithm 4.3 the convergence analysis is based on [10].

4.4.1 Convergence of TCG

Throughout this section, we make the following assumption.

Assumption 4.1. *The objective function $\mathcal{H}_X$ is continuously differentiable and there exists a Lipschitzian constant $L > 0$ such that*

$$|\mathrm{D}\mathcal{H}_X(W + t\eta)[\eta] - \mathrm{D}\mathcal{H}_X(W)[\eta]| \leq Lt$$

for all skew-symmetric matrix η, $\|\eta\| = 1$, $t > 0$.

The following two lemmas are crucial for the final convergence theorem.

Lemma 4.1. *Suppose Algorithm 4.1 does not terminate in finitely many iterations. Then we have, for all k,*

$$\langle \mathcal{H}_X(W_k), \eta_k \rangle < 0. \tag{4.31}$$

Therefore, $\beta^D_{k+1} > 0$ and $\beta_{k+1} \in [0, \beta^D_{k+1}]$ is well defined.

Proof. We proceed by induction. Since $\eta_0 = -\nabla\mathcal{H}_X(W_0)$, (4.31) holds immediately for $k = 0$. Suppose (4.31) holds for some k. Then $\beta^D_{k+1} > 0$ according to (4.18) and therefore β_{k+1} is well defined and the ratio $r_{k+1} = \frac{\beta_{k+1}}{\beta^D_{k+1}}$ lies in $[0, 1]$. Consider the definition of β^D_{k+1}, (4.18), we have

$$\begin{aligned}
&\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1} \rangle \\
&= \langle \nabla\mathcal{H}_X(W_{k+1}), -\nabla\mathcal{H}_X(W_{k+1}) + \beta_{k+1}\eta_k \rangle \\
&= \langle \nabla\mathcal{H}_X(W_{k+1}), \beta_{k+1}\eta_k \rangle - \langle \nabla\mathcal{H}(W_{k+1}), \nabla\mathcal{H}_X(W_{k+1}) \rangle \\
&= r_{k+1}\langle \nabla\mathcal{H}_X(W_{k+1}), \beta^D_{k+1} \rangle - \langle \nabla\mathcal{H}(W_{k+1}), \nabla\mathcal{H}_X(W_{k+1}) \rangle \\
&= \frac{r_{k+1}\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_k \rangle - \max\{\langle y_k, \eta_k \rangle, \langle -\nabla\mathcal{H}_X(W_k), \eta \rangle\}}{\max\{\langle y_k, \eta_k \rangle, \langle -\nabla\mathcal{H}_X(W_k), \eta \rangle\}} \|\nabla\mathcal{H}_X(W_{k+1})\|^2
\end{aligned} \tag{4.32}$$

If $\langle \nabla\mathcal{H}_X(W_{k+1}), \eta \rangle \geq 0$, it follows from the induction hypothesis that

$$\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_k \rangle - \langle \nabla\mathcal{H}_X(W_k), \eta_k \rangle = \langle y_k, \eta_k \rangle > 0$$

and, by (4.32) and last inequation,

$$\begin{aligned}
\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1} \rangle &= \frac{(r_{k+1} - 1)\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_k \rangle + \langle \nabla\mathcal{H}_X(W_k), \eta_k \rangle}{\langle y_k, \eta_k \rangle} \|\nabla\mathcal{H}_X(W_{k+1})\|^2 \\
&< 0.
\end{aligned}$$

If $\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_k \rangle < 0$, it follows from (4.32) that

$$\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1} \rangle = -\frac{\langle \nabla\mathcal{H}_X(W_k), \eta_k \rangle + r_{k+1}\langle \nabla\mathcal{H}_X(W_{k+1}), \eta_k \rangle}{\langle \nabla\mathcal{H}_X(W_k), \eta_k \rangle} \|\nabla\mathcal{H}_X(W_{k+1})\|^2 < 0.$$

Therefore, (4.31) holds for $k + 1$ in both cases. By induction, we conclude (4.31) is true for all k. $\square$

Lemma 4.2. *Suppose that $\mathcal{H}_X$ satisfies Assumption 4.1. Then there exists a positive constant $\mu > 0$ for Algorithm 4.1 such that*

$$\alpha_k \geq \min\left\{\alpha_k^{\text{init}}, -\mu\frac{\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle}{\|\eta_k\|} \quad \textit{for all } k,\right\} \tag{4.33}$$

where α_k^{init} denotes the initial steplength at iteration k. Furthermore, we have

$$\sum_{j\geq 1}\min_{i=1,\cdots,m}\{-\alpha_{mj+i-2}\langle\nabla\mathcal{H}_X(W_{mj+i-2}),\eta_{mj+i-2}\rangle\} < +\infty. \tag{4.34}$$

Proof. It follows from Taylor's Theorem that

$$\begin{aligned}\mathcal{H}_X(W_k+\alpha_k\eta_k)-\mathcal{H}_X(W_k) &= \alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle + \int_0^{\alpha_k}(\mathrm{D}\mathcal{H}_X(W_k+t\eta_k)[\eta_k]-\mathrm{D}\mathcal{H}_X(W_k)[\eta_k])\mathrm{d}t\\ &\leq \alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle + \int_0^{\alpha_k}|\mathrm{D}\mathcal{H}_X(W_k+t\eta_k)[\eta_k]-\mathrm{D}\mathcal{H}_X(W_k)[\eta_k]|\mathrm{d}t\\ &\leq \alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle + \int_0^{\alpha_k\|\eta_k\|}Lt\mathrm{d}t\\ &= \alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle + \frac{1}{2}L\alpha_k^2\|\eta_k\|^2,\end{aligned}$$

where the last inequality uses the Lipschitzian property in Assumption 4.1. By Lemma 4.1, $\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle < 0$. Thus

$$\alpha_k \leq \frac{2(\delta-1)\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle}{L\|\eta_k\|^2}$$

implies

$$\mathcal{H}_X(W_{k+1}) \leq \mathcal{H}_X(W_k) + \delta\alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle.$$

The nonmonotone condition is

$$\mathcal{H}_X(W_{k+1}) \leq \max\{\mathcal{H}_X(W_k),\cdots,\mathcal{H}_X(W_{k-\min\{m-1,k\}})\} + \delta\alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle \tag{4.35}$$

hence (4.35) holds for $\mu = \frac{2(\delta-1)\lambda}{L}$, where λ is the reduction factor for the steplength.

The proof of the second part is as follows. For $j \geq 0$, define

$$H_j = \max\{\mathcal{H}_X(W_{mj}),\mathcal{H}_X(W_{mj+1}),\cdots,\mathcal{H}_X(W_{mj+m-j})\}.$$

We show by induction that

$$\mathcal{H}_X(W_{mj+i-1}) \leq H_{j-1} + \delta\alpha_{mj+i-2}\langle\nabla\mathcal{H}_X(W_{mj+i-2}),\eta_{mj+i-2}\rangle \quad \text{for all } i=1,\cdots,m. \tag{4.36}$$

By (4.35), it is easy to see that (4.36) holds for $i = 1$. Assume (4.36) holds for all $i = 1,\cdots,l$ for some $1 \leq l \leq m-1$. Then it follows from $\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle$ that

$$\mathcal{H}_X(W_{mj+i-1}) \leq H_{j-1} \quad \text{for all } i = 1,\cdots l.$$

This, together with (4.35), implies that

$$\begin{aligned}\mathcal{H}_X(W_{mj+l}) &\leq \max\{\mathcal{H}_X(W_{mj+l-1}), \mathcal{H}_X(W_{mj+l-2}), \cdots, \mathcal{H}_X(W_{mj+l-m})\} \\ &\quad + \delta\alpha_{mj+l-1}\langle\nabla\mathcal{H}_X(W_{mj+l-1}), \eta_{mj+l-1}\rangle \\ &\leq H_{j-1} + \delta\alpha_{mj+l-1}\langle\nabla\mathcal{H}_X(W_{mj+l-1}), \eta_{mj+l-1}\rangle\end{aligned}$$

which means (4.36) holds for $l+1$ and therefore holds for $i = 1, \cdots, m$. By (4.36) and the definition of H_j, we have

$$H_j \leq H_{j-1} + \delta \max_{i=1,\cdots,m}\{\alpha_{mj+i-2}\langle\nabla\mathcal{H}_X(W_{mj+i-2}), \eta_{mj+i-2}\rangle\}. \tag{4.37}$$

Since Assumption 4.1 implies $\{H_j\}$ is bounded below, we deduce by summing (4.37) over j that (4.34) is true. $\square$

Theorem 8. *Suppose Assumption 4.1 holds and Algorithm 4.1 does not terminate in finitely many iterations. Then the sequence $\{W_k\}$ generated by Algorithm 4.1 converges in the sense that*

$$\liminf_{k\to\infty} \|\nabla\mathcal{H}_X(W_k)\| = 0. \tag{4.38}$$

Proof. We prove this theorem by contradiction. Suppose $\liminf_{k\to\infty}\|\nabla\mathcal{H}_X(W_k)\| \neq 0$. This means that there exists a constant $\gamma \in (0,1)$ such that

$$\|\nabla\mathcal{H}_X(W_k)\| \geq \gamma \quad \text{for all } k. \tag{4.39}$$

It can be seen from (4.32) that the formula (4.18) of β^D_{k+1} can be rewritten as

$$\begin{aligned}\beta^D_{k+1} &= \frac{\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle}{r_{k+1}\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_k\rangle - \max\{\langle y_k, \eta_k\rangle, \langle -\nabla\mathcal{H}_X(W_k), \eta\rangle\}} \\ &= \frac{\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle}{\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_k\rangle - \max\{(1-r_{k+1})\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_k\rangle, -r_{k+1}\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_k\rangle\}},\end{aligned}$$

which, together with the results of Lemma 4.1, yields

$$0 \leq \beta_{k+1} \leq \beta^D_{k+1} \leq \frac{\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle}{\langle\nabla\mathcal{H}_X(W_k), \eta_k\rangle}. \tag{4.40}$$

Using $\eta_{k+1} = -\nabla\mathcal{H}_X(W_{k+1}) + \beta_{k+1}\eta_k$, we have

$$\|\eta_{k+1}\|^2 = -2\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle - \|\nabla\mathcal{H}_X(W_{k+1})\|^2 + \beta^2_{k+1}\|\eta_k\|^2. \tag{4.41}$$

Dividing (4.41) by $\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle^2$ and using (4.40), we obtain

$$\frac{\|\eta_{k+1}\|^2}{\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle^2} = -\frac{2}{\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle} - \frac{\|\nabla\mathcal{H}_X(W_{k+1})\|^2}{\langle\nabla\mathcal{H}_X(W_{k+1}), \eta_{k+1}\rangle^2}$$

$$
\begin{aligned}
&+\frac{\|\eta_k\|^2}{\langle\nabla\mathcal{H}_X(W_{k+1}),\eta_k\rangle^2}\\
&=-\left(\frac{\|HX(W_{k+1})\|}{\langle\nabla\mathcal{H}_X(W_{k+1}),\eta_{k+1}\rangle}+\frac{1}{\|\nabla\mathcal{H}_X(W_{k+1})\|}\right)^2\\
&\quad+\frac{1}{\|\nabla\mathcal{H}_X(W_{k+1})\|^2}+\frac{\|\eta_k\|^2}{\langle\nabla\mathcal{H}_X(W_{k+1}),\eta_k\rangle^2}\\
&\le\frac{1}{\|\nabla\mathcal{H}_X(W_{k+1})\|^2}+\frac{\|\eta_k\|^2}{\langle\nabla\mathcal{H}_X(W_{k+1}),\eta_k\rangle^2}. \qquad (4.42)
\end{aligned}
$$

Combining the recursion (4.42) and assumption (4.39), we obtain

$$
\begin{aligned}
\frac{\|\eta_k\|^2}{\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle}&\le\sum_{i=1}^{k}\frac{1}{\|\nabla\mathcal{H}_X(W_i)\|^2}+\frac{\|\eta_0\|^2}{\langle\nabla\mathcal{H}_X(W_0),\eta_0\rangle^2}\\
&=\sum_{i=1}^{k}\frac{1}{\|\nabla\mathcal{H}_X(W_i)\|^2}+1\\
&\le\frac{k+1}{\gamma^2.} \qquad (4.43)
\end{aligned}
$$

Then

$$\frac{\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle^2}{\|\eta_k\|^2}\ge\frac{\gamma^2}{k+1}. \qquad (4.44)$$

On the other hand, from the first inequality in (4.42) and (4.43), we have

$$\frac{\|\nabla\mathcal{H}_X(W_k)\|^2}{\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle^2}+\frac{2}{\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle}\le\frac{\|\eta_{k-1}\|^2}{\langle\nabla\mathcal{H}_X(W_{k-1}),\eta_{k-1}\rangle^2}\le\frac{k}{\gamma^2},$$

which yields

$$\frac{k}{\gamma^2}\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle^2\ge2\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle+\|\nabla\mathcal{H}_X(W_k)\|^2. \qquad (4.45)$$

If

$$-\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle\le\frac{3}{8}\|\nabla\mathcal{H}_X(W_k)\|^2,$$

it follows from (4.45) that

$$\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle^2\ge\frac{\gamma^2}{4k}\|\nabla\mathcal{H}_X(W_k)\|^2.$$

Therefore, we have from (4.31) and (4.39) that

$$-\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle\ge\min\left\{\frac{3\gamma^2}{8},\frac{\gamma^2}{2\sqrt{k}}\right\}. \qquad (4.46)$$

Using $\alpha_k^{\text{init}} \geq \alpha_{\min}$, (4.33), (4.44) and (4.46), we have

$$\sum_{j\geq 1} \min_{i=1,\cdots,m} \{-\alpha_{mj+i-2}\langle \nabla\mathcal{H}_X(W_{mj+i-2}), \eta_{mj+i-2}\rangle\}$$

$$\sum_{j\geq 1} \min_{i=1,\cdots,m} \min\left\{-\alpha_{mj+i-2}\langle\nabla\mathcal{H}_X(W_{mj+i-2}),\eta_{mj+i-2}\rangle, \mu\frac{\langle\nabla\mathcal{H}_X(W_{mj+i-2}),\eta_{mj+i-2}\rangle^2}{\|\eta_{mj+i-2}\|^2}\right\}$$

$$\geq \sum_{j\geq 1} \min_{i=1,\cdots,m} \min\left\{\frac{3\gamma^2\alpha_{\min}}{8}, \frac{\gamma^2\alpha_{\min}}{2\sqrt{mj+i-2}}, \frac{\mu\gamma^2}{mj+i-1}\right\}$$

$$\geq \min\left\{\frac{3\gamma^2\alpha_{\min}}{8}, \frac{\gamma^2\alpha_{\min}}{2\sqrt{m(j+1)}}, \frac{\mu\gamma^2}{m(j+1)}\right\}$$

$$= +\infty,$$

which contradicts (4.34). Therefore (4.39) is impossible and (4.38) follows. □

Note that we are considering convergence in the space of skew-symmetric matrices. Nevertheless, Theorem 7 assures that if we find a minimizer W^* of $\mathcal{H}_X$ then a minimizer of $\mathcal{F}$ can be found by taking $Y(W^*)$.

On the other hand, convergence in the Riemannian sense follows from the above analysis in the following manner.

The nonmonotone condition in Algorithm 4.1

$$\mathcal{H}_X(W_k+\alpha_k\eta_k) \leq \max\{\mathcal{H}_X(W_k),\cdots,\mathcal{H}_X(W_{k-\min\{m-1,k\}})\} + \delta\alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle$$

can be written as

$$\mathcal{F}(Y_{k+1}) \leq \max\{\mathcal{F}(Y_k),\cdots,\mathcal{F}(Y_{k-\min\{m-1,k\}})\} + \delta\alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle. \quad (4.47)$$

Moreover, Lemma 4.1 assures $\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle < 0$ for all k. Then there exists $\hat{\delta} \in (0,1)$ such that

$$\delta\alpha_k\langle\nabla\mathcal{H}_X(W_k),\eta_k\rangle \leq \hat{\delta}\alpha_k\langle\mathcal{F}(Y_k),\xi_k\rangle, \quad \text{for all } k,$$

where $\nabla\mathcal{F}(Y_k)$ is the Riemannian gradient of $\mathcal{F}$, $\xi_k = -\nabla\mathcal{F}(Y_k) + \eta_k\mathcal{T}_{\alpha_{k-1}\xi_{k-1}}(\xi_{k-1})$ is the conjugate direction corresponding to Riemannian conjugate gradient in Algorithm 3.3. The existence of $\hat{\delta}$ is assured since we have $\langle\mathcal{F}(Y_k),\xi_k\rangle < 0$ for all k, according to generalization of Lemma 4.1 made in [52]. Now, (4.47) can be written as

$$\mathcal{F}(Y_{k+1}) \leq \max\{\mathcal{F}(Y_k),\cdots,\mathcal{F}(Y_{k-\min\{m-1,k\}})\} + \hat{\delta}\alpha_k\langle\nabla\mathcal{F}(Y_k),\xi_k\rangle. \quad (4.48)$$

On the other hand, convergence of Algorithm 3.3 is proven in [52]. Since (4.48)is the same nonmonotone condition of Algorithm 3.3 and it is satisfied for all k, we have

$$\lim_{k\to\infty}\|\nabla\mathcal{F}(Y_k)\| = 0. \quad (4.49)$$

Now, our implementation considers and actualization of X every k_r iterations. Let us index with r every change on X, so that, we have functions $\mathcal{H}_{X_r}$, $r = 1, 2, ...$ Each $\mathcal{H}_{X_r}(W_k)$ is equivalent to $\mathcal{F}(Y_k)$, where $Y_k = (I - W_k)^{-1}(I + W_k)X_r$. Since all W_k is generated satisfying the nonmonotone condition (4.47), expression (4.48) also holds for $\{\mathcal{F}(Y_k) = \mathcal{H}_{X_r}(W_k)\}$. Therefore, (4.49) holds and our implementation with actualizations on X converges.

4.4.2 Convergence of line-search methods using Armijo condition

Convergence analysis of this method assumes the sequence $\{\eta_k\}$ to be gradient related to $\{W_k\}$, according to the following definition.

Definition 4.1. *Let $\{W_k\}$ a sequence of iterates and the corresponding sequence of search directions $\{\eta_k\}$. The sequence $\{\eta_k\}$ is gradient related to $\{W_k\}$ if, for any subsequence $\{W_k\}_{k\in\mathcal{K}}$ that converges to a nonstationary point, the corresponding sequence $\{\eta_k\}_{k\in\mathcal{K}}$ is bounded and satisfies*

$$\limsup_{k\to\infty, k\in\mathcal{K}} \langle \nabla\mathcal{H}_X(W_k), \eta_k\rangle < 0.$$

Theorem 9. *Let $\{W_k\}$ be a sequence generated by Algorithm 4.3. Then every accumulation point of $\{W_k\}$ is a critical point of $\mathcal{H}_X$.*

Proof. By contradiction. Suppose that there is a subsequence $\{W_k\}_{k\in\mathcal{K}}$ converging to some W_* with $\nabla\mathcal{H}_X(W_*) \neq 0$. Since $\{\mathcal{H}_X(W_k)\}$ is nonincreasing, it follows that the whole sequence converges to $\mathcal{H}_X(W_*)$. Hence,

$$\mathcal{H}_X(W_k) - \mathcal{H}_X(W_{k+1}) \to 0.$$

By definition of the Armijo rule, we have

$$\mathcal{H}_X(W_k) - \mathcal{H}_X(W_{k+1}) \geq -\delta\alpha_k\langle \nabla\mathcal{H}_X(W_k), \eta_k\rangle. \tag{4.50}$$

Since $\{\eta_k\}$ is gradient related, we must have $\{\alpha_k\}_{k\in\mathcal{K}} \to 0$. Hence, by the definition of the Armijo rule, we must have for some index $\bar{k} \geq 0$

$$\mathcal{H}_X(W_k) - \mathcal{H}_X(W_k + (\alpha_k/\lambda)\eta_k) < -\delta\frac{\alpha_k}{\lambda}\langle \nabla\mathcal{H}_X(W_k), \eta_k\rangle, \quad \forall k \in \mathcal{K}, k \geq \bar{k}, \tag{4.51}$$

that is, the initial stepsize will be reduced at least once for all $k \in \mathcal{K}$, $k \geq \bar{k}$. Denote

$$p_k = \frac{\eta_k}{\|\eta_k\|}, \quad \bar{\alpha_k} = \frac{\alpha_k\|\eta_k\|}{\lambda}.$$

Since $\{\eta_k\}$ is gradient related, $\{\|\eta_k\|\}_{\mathcal{K}}$ is bounded, it follows that

$$\{\bar{\alpha_k}\} \to 0.$$

Since $\|p_k\| = 1$ for all $k \in \mathcal{K}$, there exists a subsequence $\{p_k\}_{\bar{\mathcal{K}}}$ of $\{p_k\}_{\mathcal{K}}$ such that

$$\{p_k\}_{\bar{\mathcal{K}}} \to \bar{p},$$

$\|\bar{p}\| = 1$. From (4.51), we have

$$\frac{\nabla\mathcal{H}_X(W_k) - \nabla\mathcal{H}_X(W_k + \bar{\alpha}_k p_k)}{\bar{\alpha_k}} < \delta\langle\nabla\mathcal{H}_X(W_k), p_k\rangle, \quad \forall k \in \bar{\mathcal{K}}, k \geq \bar{k}.$$

By using the mean value theorem, this relation is written as

$$-\langle\nabla\mathcal{H}_X(W_k + \tilde{\alpha}_k p_k), p_k\rangle < -\delta\langle\nabla\mathcal{H}_X(W_k), p_k\rangle, \quad \forall k \in \bar{\mathcal{K}}, k \geq \bar{k},$$

where $\tilde{\alpha}_k$ is a scalar in the interval $[0, \bar{\alpha}_k]$. Taking limits in the above equation we obtain

$$-\langle\mathcal{H}_X(W_*), \bar{p}\rangle \leq -\delta\langle\mathcal{H}_X(W_*), \bar{p}\rangle$$

or

$$0 \leq (1-\delta)\langle\mathcal{H}_X(W_*), \bar{p}\rangle.$$

Since $\delta < 1$, it follows that

$$0 \leq \langle\mathcal{H}_X(W_*), \bar{p}\rangle \tag{4.52}$$

On the other hand, we have

$$\langle\nabla\mathcal{H}_X(W_k), \eta_k\rangle = \frac{\langle\nabla\mathcal{H}_X(W_k), \eta_k\rangle}{\|\eta_k\|}.$$

By taking limit as $k \in \bar{\mathcal{K}}$, $k \to \infty$,

$$\langle\nabla\mathcal{H}_X(W_*), \bar{p}\rangle < 0,$$

which contradicts (4.52). □

Corollary 4.2. *Let $\{W_k\}$ be an infinite sequence of iterates generated by Algorithm 4.3. Assume that the level set $\mathcal{L} = \{W : \nabla\mathcal{H}_X(W) \leq \nabla\mathcal{H}_X(W_0)\}$ is compact. Then $\lim_{k\to\infty} \|\nabla\mathcal{H}_X(W_k)\| = 0$.*

Proof. By contradiction, assume the contrary. Then there is a subsequence $\{W_k\}_{k\in\mathcal{K}}$ and $\epsilon > 0$ such that $\|\nabla\mathcal{H}_X(W_k)\| > \epsilon$ for all $k \in \mathcal{K}$. Because is nonincreasing on $\{W_k\}$, it follows that $W_k \in \mathcal{L}$ for all k. Since $\mathcal{L}$ is compact, $\{W_k\}_{k\mathcal{K}}$ has an accumulation point W_* in $\mathcal{L}$. By the continuity of $\nabla\mathcal{H}_X$, one has $\|\nabla\mathcal{H}_X(W_*)\| \geq \epsilon$, i.e., W_* is not critical, a contradiction to Theorem 9. □

4.5 Transportless Conjugate Gradient using the Polar Decomposition

As an alternative to $Y = (I - W)^{-1}(I + W)X$, we can parametrize $\mathrm{St}(n, p)$ using the Polar Decomposition as

$$Y^P = M(M^\top M)^{-1/2}, \quad M \in \mathbb{R}^{n\times p}, \mathrm{rank}(M) = p. \tag{4.53}$$

It is clear that Y^P is a surjective mapping from $\mathrm{dom}(Y^P) \subset \mathbb{R}^{n\times p}$ onto $\mathrm{St}(n, p)$. Hence, it makes sense to define

$$\begin{aligned} \mathcal{H}(M) : \mathrm{dom}(Y^P) &\to \mathbb{R} \\ M &\mapsto \mathcal{H}(M) := \mathcal{F}(Y^P(M)), \end{aligned}$$

and work on the optimization of $\mathcal{H}$ instead of $\mathcal{F}$ in order to apply Euclidean optimization methods. Since we have to maintain all iterates in $\mathrm{dom}(Y^P)$ all $M_k^\top M_k$ should be invertible and we deal with this in the implementation of the algorithm. Details on this implementation will be given after the next section.

On the other hand, it is clear that the computational cost of the eigendecomposition of $M^\top M$ is reasonable for small p, which makes this approach cheaper than our approach using the Cayley Transform. Nevertheless, computing the gradient of $\mathcal{H}$ requires the solution of a p^2-by-p^2 linear system. Hence, this approach works fine for our experiments since small p is being considered, but may not be competitive for large p.

4.5.1 Gradient of $\mathcal{H}(M)$

Let $A \in \mathbb{R}^{p\times p}$ be a symmetric matrix. We have

$$\begin{aligned} I_{p^2} &= \frac{\mathrm{d}A^{1/2}A^{1/2}}{\mathrm{d}A} \\ &= (A^{1/2} \otimes I_p)\frac{\mathrm{d}A^{1/2}}{\mathrm{d}A} + (I_p \otimes A^{1/2})\frac{\mathrm{d}A^{1/2}}{\mathrm{d}A}, \end{aligned}$$

hence,

$$\frac{\mathrm{d}A^{1/2}}{\mathrm{d}A} = [(A^{1/2} \otimes I_p) + (I_p \otimes A^{1/2})]^{-1} = (A^{1/2} \oplus A^{1/2})^{-1}.$$

Now,

$$\begin{aligned} 0 &= \frac{\mathrm{d}A^{1/2}A^{-1/2}}{\mathrm{d}A} \\ &= (A^{-1/2} \otimes I_p)(A^{1/2} \oplus A^{1/2})^{-1} + (I_p \otimes A^{1/2})\frac{\mathrm{d}A^{-1/2}}{\mathrm{d}A}, \end{aligned}$$

so that,

$$\frac{\mathrm{d}A^{-1/2}}{\mathrm{d}A} = -(A^{-1/2} \otimes A^{-1/2})(A^{1/2} \oplus A^{1/2})^{-1}.$$

We are interested in

$$\begin{aligned}\frac{\mathrm{d}(M^\top M)^{-1/2}}{\mathrm{d}M} &= \frac{\mathrm{d}(M^\top M)^{-1/2}}{\mathrm{d}M^\top M}\frac{\mathrm{d}M^\top M}{\mathrm{d}M} \\ &= -((M^\top M)^{-1/2} \otimes (M^\top M)^{-1/2})((M^\top M)^{1/2} \oplus (M^\top M)^{1/2})^{-1} \\ &\quad [(M^\top \otimes I_p)T_{n,p} + (I_p \otimes M^\top)],\end{aligned}$$

and

$$\begin{aligned}\frac{\mathrm{d}Y^P}{\mathrm{d}M} &= \frac{\mathrm{d}M(M^\top M)^{-1/2}}{\mathrm{d}M} \\ &= ((M^\top M)^{-1/2} \otimes I_p) \\ &\quad - ((M^\top M)^{-1/2} \otimes M(M^\top M)^{-1/2})((M^\top M)^{1/2} \oplus (M^\top M)^{1/2})^{-1} \\ &\quad [(M^\top \otimes I_p)T_{n,p} + (I_p \otimes M^\top)].\end{aligned}$$

The chain rule yields

$$\nabla\mathcal{H}(M) = \left[\frac{\mathrm{d}\mathcal{F}(Y^P)}{\mathrm{d}Y^p}\frac{\mathrm{d}Y^P}{\mathrm{d}M}\right]^\top = G(M^\top M)^{-1/2} - M(Z^\top + Z), \tag{4.54}$$

where G is the Euclidean gradient of $\mathcal{F}$ and $Z \in \mathbb{R}^{p\times p}$ is the solution to

$$(M^\top M)^{1/2}Z + Z(M^\top M)^{1/2} = (M^\top M)^{-1/2}M^\top G(M^\top M)^{-1/2}. \tag{4.55}$$

Note that Z can be found by solving the Sylvester Equation (4.55) or by solving the sparse linear system

$$((M^\top M)^{1/2} \oplus (M^\top M)^{1/2})vec(Z) = vec((M^\top M)^{-1/2}M^\top G(M^\top M)^{-1/2}). \tag{4.56}$$

The size of the linear system (4.56) is p^2, so that, for $p^2 < n$ it is less expensive using $Y^P = M(M^\top M)^{-1/2}$ than $Y = (I - W)^{-1}(I + W)X$, which requires to solve a n-by-n linear system.

4.5.2 Non-singularity of $M^\top M$

For any $X \in \mathrm{St}(n,p)$, we have

$$\nabla\mathcal{H}(X) = G - X\left(\frac{X^\top G + G^\top X}{2}\right), \tag{4.57}$$

that is, $\nabla\mathcal{H}$ coincides with the Riemannian gradient of $\mathcal{F}$ at X [4]. In this regard, if $M = X - \alpha\nabla\mathcal{H}(X)$ is the new trial point we can assure that $M^\top M$ is not singular since

$$\begin{aligned} M^\top M &= (X - \alpha\mathcal{H}(X))^\top (X - \alpha\mathcal{H}(X)) \\ &= (X - \alpha\nabla\mathcal{F}(X))^\top (X - \alpha\nabla\mathcal{F}(X)) \\ &= I_p - \alpha\nabla\mathcal{F}(X)^\top X - \alpha X^\top \nabla\mathcal{F}(X) + \alpha^2\nabla\mathcal{F}(X)^\top\nabla\mathcal{F}(X) \\ &= I_p + \alpha^2\nabla\mathcal{F}(X)^\top\nabla\mathcal{F}(X), \end{aligned}$$

that is, $M^\top M$ is a positive definite matrix. Thus, from M_0 it is always possible to generate the next iterate M_1 using this scheme. However, when $X \notin \mathrm{St}(n,p)$ we can not assure that $M^\top M$ is not singular.

If the trial point $M = M_k + \alpha\eta_k$ is such that $M^\top M$ is singular we can restart the algorithm by taking $M_k = X(M_k)$ and $\eta_k = \nabla\mathcal{H}(M_k)$. Although, it is unlikely for the stepsize to be chosen such that $M^\top M$ is singular. However, we have notice that using the Riemannian gradient as search direction can accelerate the convergence of our method and for our implementation we restart the algorithm every 10 iterations.

Since the numerical experiments will consider only low-rank cases, this approach based on the Polar Decomposition is expected to be more efficient than the one based on the Cayley Transform.

Once again, we will use Dai's nonmonotone conjugate gradient for unconstrained optimization [13]. Note that convergence of this method, proved in Section 4.4.1 for $\mathcal{H}_X$, can be easily adapted for $\mathcal{H}$. A noticeable advantage of $\mathcal{H}$ is that inner products $\langle\mathcal{H}(M), \eta_k\rangle$ are computed for n-by-p matrices, which for a small p is less expensive than inner products between n-by-n matrices corresponding to optimization over the space of skew-symmetric matrices. .

Implementation of Algorithm 4.1 considers a restarting strategy every k_r iterations. This strategy is experimentally proved to enhance the performance of the algorithm. Considering this, we use the same strategy for the implementation used in the implementation based on the Cayley Transform. In the case of the Polar Decomposition, after every k_r iterations we take $M = Y^P$ and $\beta_{k+1} = 0$.

Chapter 5

Numerical experiments

In this chapter we analyze the performance of our two optimization methods in several instances of problem (1.1). The experiments are simulated and inspired on those presented in [52]. Also, a comparison between the performance of algorithms of Chapter 3 and our method is presented in this chapter.

5.1 Implementation Details

All experiments were executed using MATLAB R2014a in an ASUS laptop with an Intel Corei5-8250U processor, 3.4GHz, 1TB of HD and 8GB of RAM.

In order to have fare comparisons, we will use the same constant parameters for all algorithms whenever possible. The initial step-size is set up as α_0 =1e-3. The value δ =1e-4 corresponds to the constant in Armijo condition. For the backtracking procedure, the current step-size will be diminished by a factor of λ =0.2. The nonmonotone line search backward integer for consecutive previous function values is set up as m =2. The constants that appear specifically in just one algorithm are kept to default in the author's implementation.

Regarding stopping criteria, although our algorithm works in the space of skew-symmetric matrices, we will use the same conditions of Riemannian methods in order to have a fare comparison. Tolerance on the norm gradient will consider $\|\nabla_c \mathcal{F}(Y_k)\|_F < \epsilon$ where $Y_k = (I - W_k)^{-1}(I + W_k)X$ and $\nabla_c \mathcal{F}(Y_k)$ is the Riemannian gradient under the canonical metric

$$\nabla_c \mathcal{F}(Y_k) = G_k - Y_k G_k^\top Y_k,$$

where G_k is the Euclidean gradient of $\mathcal{F}$ at Y_k. The norm of the difference between two iterates will be computed as $\|Y_{k+1} - Y_k\|_F$. Regarding the difference of the value of objective function between iterates there is no confusion since $\mathcal{F}(Y_{k+1}) - \mathcal{F}(Y_k) = \mathcal{H}_X(W_{k+1}) - \mathcal{H}_X(W_k)$.

We let our algorithms run up to K iterations and stop at iteration $k < K$ if one of the following stopping criteria is satisfied:

1. $\text{tol}_k^G := \|\nabla_c \mathcal{F}(Y_k)\|_F < \epsilon$
2. $\text{tol}_k^x := \frac{\|Y_{k+1}-Y_k\|_F}{\sqrt{n}} < x_{\text{tol}}$ and $\text{tol}_k^f := \frac{|\mathcal{F}(Y_{k+1})-\mathcal{F}(Y_k)|}{|\mathcal{F}(Y_k)|+1} < f_{\text{tol}}$
3. $\text{mean}([\text{tol}^x_{k-\min(k,T)+1}, \cdots, \text{tol}^x_k]) < 10x_{\text{tol}}$ and $\text{mean}([\text{tol}^f_{k-\min(k,T)+1}, \cdots, \text{tol}^f_k]) < 10f_{\text{tol}}$

For our experiments we use $\epsilon = 1e-6$, $x_{\text{tol}} = 1e-6$ and $f_{\text{tol}} = 1e-12$, $K = 1000$ and $T = 5$.

The results of experiments will be reported considering:

- Nfe: Number of function evaluations
- Nitr: Number of iterations
- Time: CPU time in seconds
- NrmG: Norm of the Riemannian gradient under the canonical metric
- Fval: Objective function value
- Feasi: Feasibility, defined as $\|X^\top X - I_p\|_F$ and reported for the las iterate generated by the algorithm

We use the following names to denote algorithms being compared. We refer to Wen and Yin's method from Section 3.1 as *OptiStiefelGBB*. Riemannian conjugate gradient method from Section 3.3 will be denoted as *OptiStiefelCGC* as in [52]. *Ad-Moul* will denote the algorithm based on the Adam-Moulton scheme in Section 3.2. Our Transportless Conjugate Gradient methods will be denoted as *CTCG*, for the one based on the Cayley Transform, and *PTCG*, the one based on the Polar Decomposition. Our Steepest Descent method is denoted as *GDW*.

Zhu [52] tested the Riemannian conjugate gradient for low-rank matrices because there is an efficient manner to compute the inverse matrix appearing in the Cayley Transform. Therefore, values of $p = 10$ or $p = 5$ are common in the experiments presented in [52]. For n the values 500, 1000, 5000 and 10000 are used. Nevertheless, our CTCG method has no efficient way to compute the required inverse matrix and the computational cost of this inversion depends on n. Hence, we limit our experiments to $n = 500, 1000$.

5.2 Linear Eigenvalue Problem

Our first test problem is the linear eigenvalue problem formulated as

$$\max_{X \in \mathbb{R}^{n\times p}} \text{Tr}[X^\top A X] \quad \text{s.t.} \quad X^\top X = I_p, \tag{5.1}$$

where A is a symmetric n-by-n matrix. The corresponding objective function and its Euclidean gradient are

$$\mathcal{F}(X) = -\mathrm{Tr}[X^\top AX] \quad \text{and} \quad G = -2AX. \tag{5.2}$$

Optimum value of $\mathcal{F}$ is $\mathcal{F}(X_*) = \sum_{i=1}^{p} \lambda_i$, $i = 1, ..., p$, are the leading eigenvalues of A. Note that $\mathcal{F}$ has several minimizers since, for any given minimizer X_* and $Q \in \mathrm{St}(p,p)$, trace properties yield

$$\begin{aligned}\mathcal{F}(X_*Q) &= -\mathrm{Tr}[Q^\top X_*^\top AX_*Q] \\ &= -\mathrm{Tr}[X_*^\top AX_*QQ^\top] \\ &= -\mathrm{Tr}[X_*^\top AX_*] \\ &= \mathcal{F}(X_*).\end{aligned}$$

So that, is unlikely that two different optimization methods starting at the same initial iterate would converge to the same minimizer.

5.2.1 Linear Eigenvalue problem: Experiment 1

First numerical experiment appearing in [52] considers a $A = \mathrm{diag}(1, 2, \cdots, n)$. We consider $n = 500$ and $p = 10$ for this experiment. Note that A is ill-conditioned.

We present statistics for 100 executions in Table 5.1, each of them starting at different random initial iterates $X_0 \in \mathrm{St}(n,p)$.

Regarding objective function, OptiStiefelCGC has smaller variance than other methods. However, greater value of variance belongs to GDW with 1.97e-2 which is still acceptable.

Feasibility is always maintained below 1e-13 since implementation of all algorithms use the modified Gram-Schmidt process for last iterate when its feasibility is above 1e-13. For this experiment, CTCG had the maximum value of feasibility with 6.12e-14 which us still under 1e-13. So that, all algorithms performed satisfactorily in this regard. In average, our GDW had the smallest value for feasibility very close to the corresponding value for OptiStiefelGBB; variance is also similar for both algorithms and smaller than those of other algorithms.

In average, gradient norm is in order of 1e-3 for all algorithms but is smaller for OptiStiefelGBB. Note that our GDW method has a minimum value of 2.55e-5 for gradient norm, which is smaller than the corresponding minimums of other algorithms.

Number of iterations is an important feature concerning performance of the algorithms. Regarding this aspect, state-of-the-art algorithms have similar results and they are better than those of our algorithms. Nevertheless, note that the minimum value for CTCG is 129 which is close 127, the minimum number of iterations for OptiStiefelCGC. But the variance is greater for our methods, same for the mean number of iterations.

All methods have to evaluate the objective function at least once per iteration. However, additional evaluations imply higher computational cost. Note that number of iterations and number of function evaluations is close for OptiStiefelGBB and Ad-Moul meaning that stepsize estimation is fine in both algorithms. On the other hand, our methods and OptiStiefelCGC need 100 more function evaluations than the number of iterations which means that the backtracking procedure is invoked many times increasing computational cost.

Regarding computational time, CTCG was expected to be the slowest method and this is confirmed by the results. While CTCG takes 3.53 seconds to solve the problem, all other methods need less than a second. The fastest method is OptiStiefelGBB with an average of 0.21 seconds. Note that PTCG is comparable with Ad-Moul concerning computational time.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-4.9550e+03	5.89e-15	5.61e-03	190.0	290.3	0.40
	min	-4.9550e+03	3.51e-15	1.28e-03	141.0	216.0	0.30
	max	-4.9550e+03	9.66e-15	3.31e-02	295.0	457.0	0.62
	var	1.8574e-13	1.21e-30	2.50e-05	734.2	1952.7	0.00
CTCG	mean	-4.9550e+03	1.46e-14	6.10e-03	187.2	286.8	3.83
	min	-4.9550e+03	8.94e-16	1.19e-03	129.0	184.0	2.52
	max	-4.9550e+03	6.12e-14	3.09e-02	287.0	459.0	6.03
	var	2.2797e-13	3.93e-29	2.87e-05	913.1	2542.0	0.42
GDW	mean	-4.9550e+03	7.53e-16	3.90e-03	202.6	319.8	0.76
	min	-4.9550e+03	5.05e-16	2.55e-05	133.0	208.0	0.49
	max	-4.9550e+03	1.09e-15	2.23e-02	311.0	503.0	1.19
	var	1.9782e-12	1.24e-32	1.30e-05	1447.4	3944.5	0.02
OptiStiefelGBB	mean	-4.9550e+03	7.59e-16	3.58e-03	172.3	184.0	0.21
	min	-4.9550e+03	5.57e-16	1.74e-04	123.0	128.0	0.15
	max	-4.9550e+03	1.13e-15	2.65e-02	258.0	275.0	0.31
	var	7.1860e-13	1.37e-32	1.48e-05	540.7	656.1	0.00
Ad-Moul	mean	-4.9550e+03	3.34e-15	4.57e-03	169.5	179.7	0.47
	min	-4.9550e+03	1.64e-15	4.82e-04	115.0	118.0	0.31
	max	-4.9550e+03	8.66e-15	3.53e-02	242.0	256.0	0.67
	var	1.1441e-12	1.86e-30	3.51e-05	559.0	697.1	0.00
OptiStiefelCGC	mean	-4.9550e+03	5.08e-15	5.24e-03	176.3	260.5	0.32
	min	-4.9550e+03	3.38e-15	4.58e-04	127.0	179.0	0.22
	max	-4.9550e+03	7.56e-15	1.42e-02	242.0	379.0	0.49
	var	9.7389e-14	7.29e-31	9.14e-06	594.6	1653.6	0.00

Table 5.1: Results of experiment 1 for Linear Eigenvalue Problem

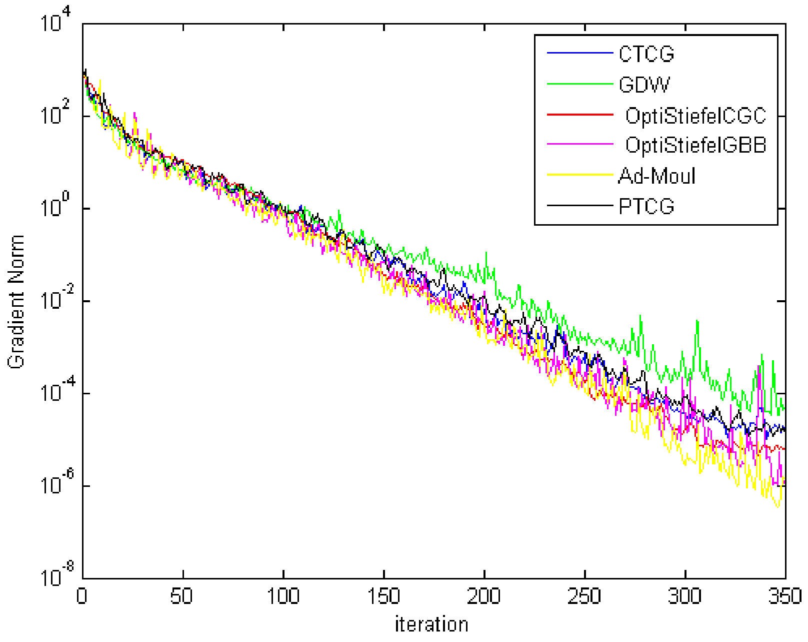

Figure 5.1: Linear Eigenvalue Problem Experiment 1: Average gradient norm

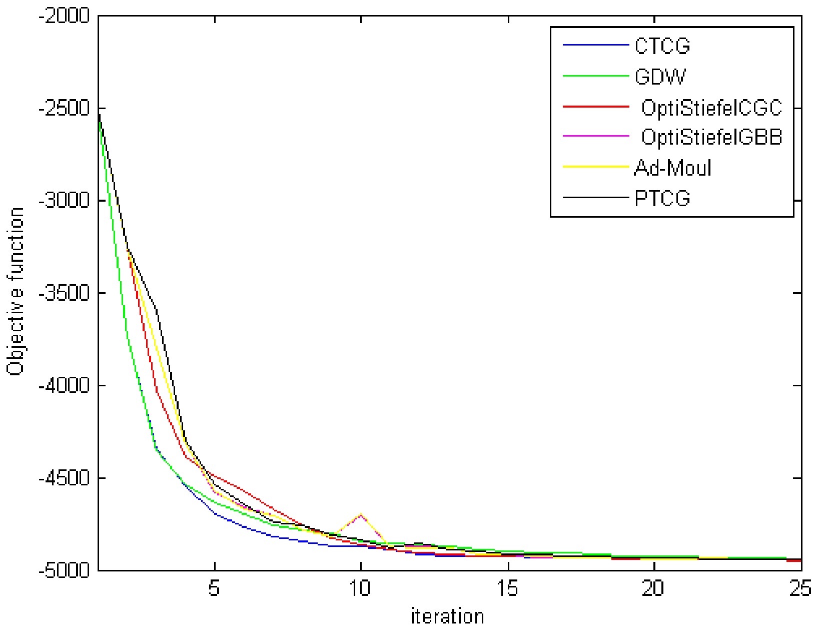

Figure 5.2: Linear Eigenvalue Problem Experiment 1: Average objective function

Aside of results in Table 5.1, we show graphics of the average descending process of gradient norm and objective function. We let algorithms run for 350 iterations without other stopping criteria. Figures 5.1 and 5.2 show that Ad-Moul and OptiStiefelGBB have faster overall descending rates than other methods, which agree with results of Table 5.1. Nevertheless, our GDW and CTCG methods descend faster for the first ten iterations.

Results Summary: Linear Eigenvalue Problem Experiment 1

Results in Table 5.1 and Figures 5.1 and 5.2 show OptiStiefelGBB and Ad-Moul to have a better performance in this ill-conditioned experiment. Mainly, smaller number of iterations and computational time are the features that make these methods better than others. However, OptiStiefelCGC and our PTCG method have comparable performances, specifically they are close to OptiStiefelGBB and Ad-Moul regarding computational time.

5.2.2 Linear Eigenvalue problem: Experiment 2

As said before, matrix $A = \text{diag}(1, 2, \cdots n)$ of the first experiment is an ill-conditioned matrix. In order to test performance of algorithms in a well conditioned problem, we keep a diagonal matrix but built as $A = \text{diag}(2 * \text{ones}(n, 1) + \text{rand}(n, 1))$, that is, the elements in the diagonal of A are numbers between 2 and 3. Dimensions of the problem are kept as $n = 500$ and $p = 10$.

Results displayed in Table 5.2 show statistic for 100 executions of this problem, starting at different random initial iterates.

Observe in the Fval column that OptiStiefelCGC, PTCG and GDW did not reach the optimal value in all occasions. Also, variance for these methods is large, being above 1e-10 while CTCG, OptiStiefelGBB and Ad-Moul have variance below 1e-15. Our CTCG method has the smaller variance.

Note that NrmG, the gradient norm, is larger for those algorithms that did not reach the solution every time, i.e., PTCG, GDW and OptiStiefel have maximum value of NrmG in the order of 1e-3, while this value for other methods other methods is around 1e-5.

Number of iterations and number of function evaluations are significantly larger for PTCG, GDW y OptiStiefelCGC. In average, these methods required 1000 iterations in order to reach an optimal solution but other methods have an average number of iterations below 250.

Regarding CPU time, note that GDW requires more time that TCGC in average, which was not expected since GDW is less expensive. However, large number of iterations for GDW makes it slower for this experiment. Still, average time for TCGC is around 19 times the time of OptiStiefelGBB, that is, we have no comparison with state-of-the art algorithms in this experiment.

Figure 5.3 shows that after 250 iterations GDW, PTCG and OptiStiefelCGC practically stopped descending. On the other hand, Figure 5.4 shows that the value of objective function descends faster for GDW and CTCG considering only the first ten iterations.

Results summary: Linear Eigenvalue Problem Experiment 2

This experiment revealed that GDW, PTCG and OptiStiefelCGC have a bad performance for this well-conditioned problem. Again, OptiStifelGBB and Ad-Moul are the best algorithms. Our CTCG is close regarding number of iterations but there is no comparison concerning computational time.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-2.9881e+01	5.37e-15	7.15e-05	978.0	979.0	1.51
	min	-2.9881e+01	3.16e-15	1.24e-05	671.0	672.0	1.02
	max	-2.9880e+01	8.50e-15	1.39e-03	1000.0	1001.0	1.60
	var	6.9497e-09	8.97e-31	2.59e-08	3190.8	3190.8	0.01
CTCG	mean	-2.9881e+01	2.63e-14	2.25e-05	223.3	302.6	4.22
	min	-2.9881e+01	2.19e-15	3.50e-06	161.0	213.0	3.00
	max	-2.9881e+01	9.66e-14	8.92e-05	296.0	409.0	5.65
	var	3.1094e-17	2.23e-28	2.89e-10	874.5	1698.2	0.33
GDW	mean	-2.9881e+01	7.85e-16	1.76e-04	990.1	1276.9	10.42
	min	-2.9881e+01	5.12e-16	1.51e-05	728.0	941.0	6.77
	max	-2.9879e+01	1.15e-15	1.70e-03	1000.0	1298.0	11.17
	var	2.4764e-08	1.70e-32	8.34e-08	1322.2	2175.0	0.32
OptiStiefelGBB	mean	-2.9881e+01	1.97e-15	1.12e-05	200.7	216.9	0.24
	min	-2.9881e+01	7.25e-16	1.13e-06	149.0	165.0	0.18
	max	-2.9881e+01	4.23e-15	9.57e-05	271.0	288.0	0.34
	var	1.1003e-16	4.87e-31	1.62e-10	734.1	827.0	0.00
Ad-Moul	mean	-2.9881e+01	5.11e-15	1.14e-05	197.7	212.3	0.54
	min	-2.9881e+01	1.92e-15	2.62e-06	133.0	141.0	0.36
	max	-2.9881e+01	1.19e-14	5.84e-05	275.0	289.0	0.75
	var	1.2471e-16	4.36e-30	7.33e-11	643.8	721.3	0.00
OptiStiefelCGC	mean	-2.9881e+01	8.03e-15	6.56e-05	967.3	968.3	1.32
	min	-2.9881e+01	5.16e-15	1.21e-05	635.0	636.0	0.85
	max	-2.9880e+01	1.06e-14	1.25e-03	1000.0	1001.0	1.42
	var	1.9377e-09	1.08e-30	2.86e-08	4762.1	4762.1	0.01

Table 5.2: Results of experiment 2 for Linear Eigenvalue Problem

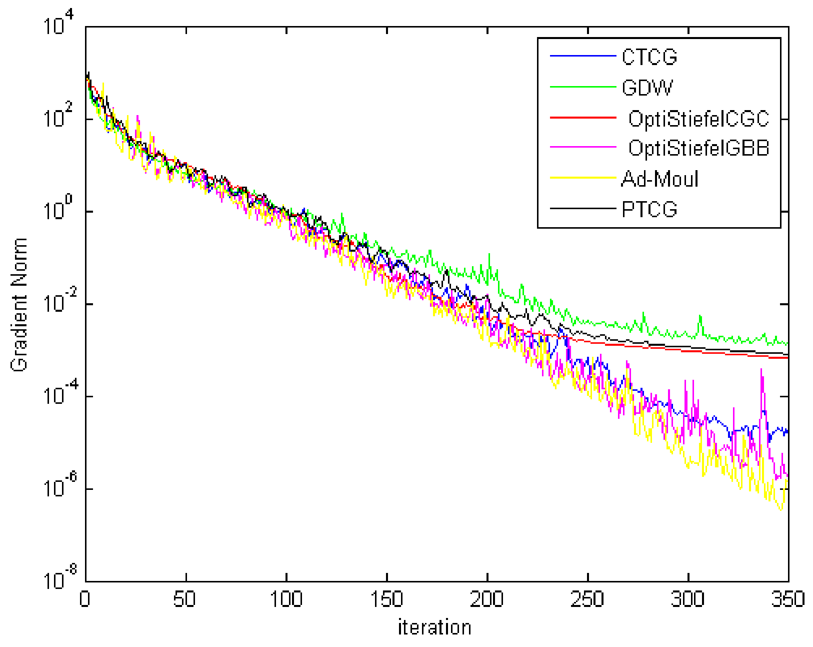

Figure 5.3: Linear Eigenvalue Problem Experiment 2: Average gradient norm

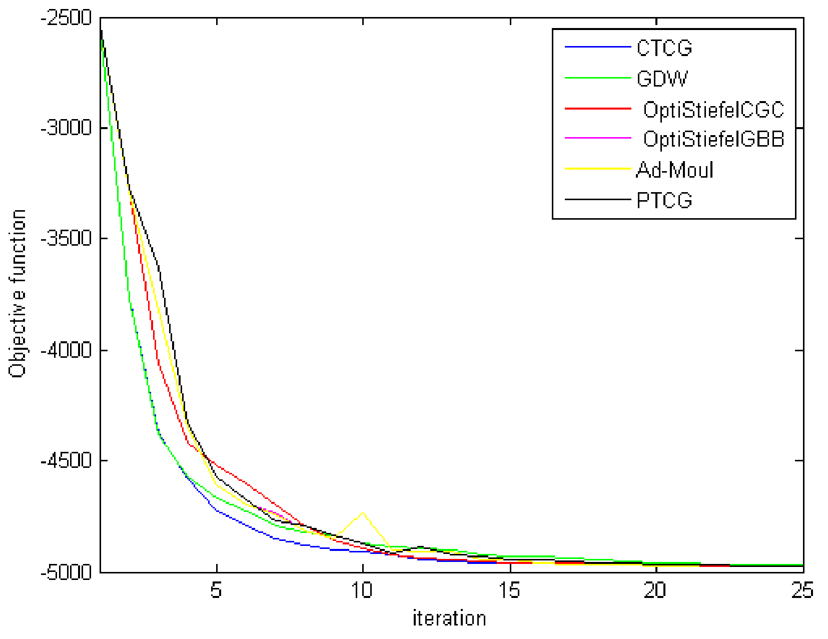

Figure 5.4: Linear Eigenvalue Problem Experiment 2: Average objective function

5.2.3 Linear Eigenvalue problem: Experiment 3

For our third experiment, we repeat experiment 1 changing the problem size. Let $n = 1000$ and $p = 5$, we consider the ill-conditioned matrix $A = \mathrm{diag}(1, 2, \cdots, n)$. Results of this experiment are displayed in Table 5.3.

Column Fval of Table 5.3 shows that optimal objective function value is reached by all algorithms. Also, variance is small for all algorithms although, OptiStiefelCGC has the smallest.

Note that both gradient descent algorithms, GDW and OptiStiefelGBB, have smaller average value for feasibility. Nevertheless, all algorithms maintain feasibility under 1e-13.

Average of gradient norm is in order of 1e-3 for all algorithms. However, GDW, OptiStiefelGBB and Ad-Moul have minimum values of gradient norm in order of 1e-4. Variance for this feature is similar for all methods.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-4.9900e+03	2.96e-15	9.00e-03	271.1	428.8	1.73
	min	-4.9900e+03	1.28e-15	1.94e-03	192.0	304.0	1.23
	max	-4.9900e+03	6.17e-15	2.81e-02	402.0	675.0	2.71
	var	4.1388e-12	7.77e-31	3.52e-05	1852.8	5113.6	0.08
CTCG	mean	-4.9900e+03	1.45e-14	8.65e-03	267.2	423.1	22.63
	min	-4.9900e+03	5.87e-15	1.56e-03	171.0	259.0	14.05
	max	-4.9900e+03	5.32e-14	2.74e-02	356.0	574.0	30.51
	var	1.1021e-12	4.96e-29	3.16e-05	1320.8	3769.3	10.27
GDW	mean	-4.9900e+03	4.77e-16	7.01e-03	281.4	454.1	4.18
	min	-4.9900e+03	2.22e-16	2.48e-04	175.0	277.0	2.58
	max	-4.9900e+03	8.97e-16	3.15e-02	421.0	675.0	6.24
	var	1.0963e-11	1.45e-32	4.00e-05	2833.2	7814.4	0.64
OptiStiefelGBB	mean	-4.9900e+03	4.80e-16	5.14e-03	231.6	248.3	0.99
	min	-4.9900e+03	2.65e-16	8.90e-04	160.0	169.0	0.67
	max	-4.9900e+03	8.05e-16	4.70e-02	356.0	384.0	1.52
	var	6.0724e-12	9.90e-33	3.26e-05	1319.5	1583.2	0.02
Ad-Moul	mean	-4.9900e+03	1.71e-15	5.52e-03	235.3	252.1	2.33
	min	-4.9900e+03	4.87e-16	4.86e-04	164.0	176.0	1.62
	max	-4.9900e+03	4.84e-15	3.30e-02	315.0	332.0	3.13
	var	1.1229e-11	5.61e-31	3.59e-05	1183.3	1376.9	0.12
OptiStiefelCGC	mean	-4.9900e+03	3.79e-15	9.24e-03	242.0	374.9	1.52
	min	-4.9900e+03	1.65e-15	1.27e-03	183.0	280.0	1.14
	max	-4.9900e+03	6.19e-15	2.26e-02	313.0	518.0	2.08
	var	6.0712e-13	8.38e-31	2.40e-05	804.3	2330.1	0.04

Table 5.3: Results of experiment 3 for Linear Eigenvalue Problem

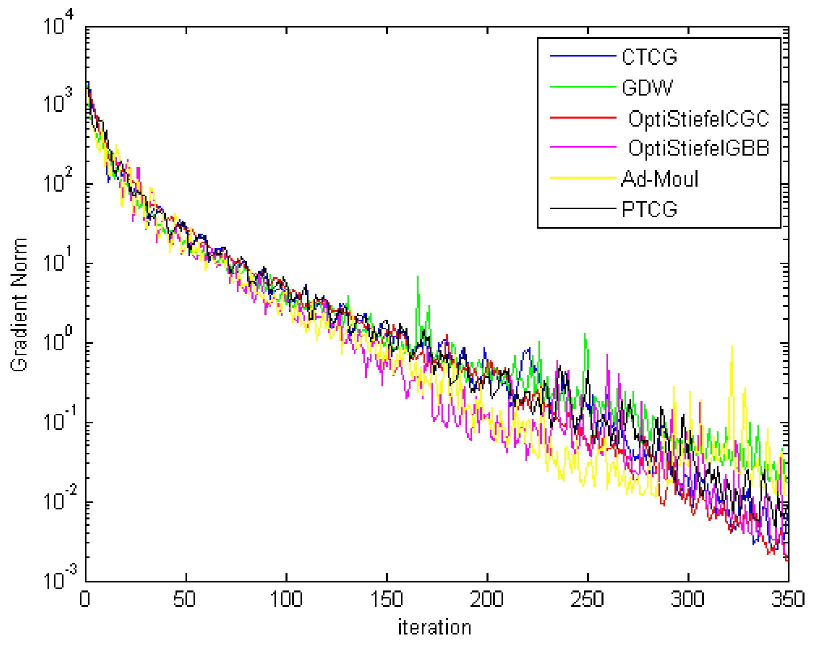

Figure 5.5: Linear Eigenvalue Problem Experiment 3: Average gradient norm

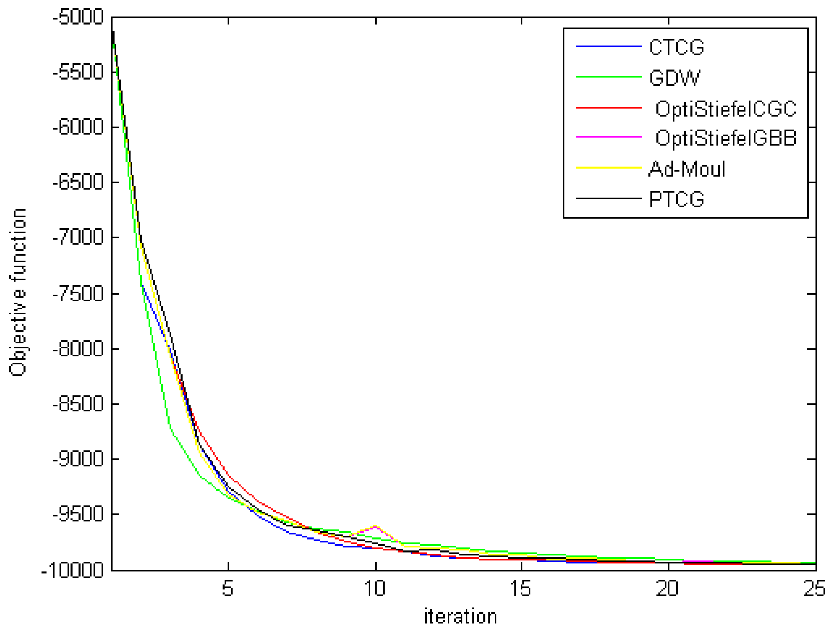

Figure 5.6: Linear Eigenvalue Problem Experiment 3: Average objective function

Ad-Moul and OptiStiefelGBB are the methods with smaller number of iterations, again. Also, the number of function evaluations is close to the number of iterations which means that the backtracking procedure is invoked few times. Our methods need 35 more iterations than OptiStiefelGBB or Ad-Moul, but the most important thing to notice is the large number of function evaluations in our methods.

Concerning computational time, only our PTCG algorithm is comparable to state-of-the-art algorithms. Actually, in spite of the larger number of iterations and function evaluations, PTCG takes less time than Ad-Moul and slightly more time than OptiStiefelCGC. As expected, CTCG is the worst algorithm in this regard, with average CPU time of 22.63 seconds. Furthermore, variance for this algorithm is 10.27 which is too large.

Results summary: Linear Eigenvalue Problem Experiment 3

OptiStiefelCGC, OptiStiefelGBB and Ad-Moul have the best results for this experiment regarding number of iterations and function evaluations. Nevertheless, the value $p = 5$ makes our PTCG competitive regarding computational time, since it is faster than Ad-Moul and slightly slower than OptiStiefelCGC.

5.2.4 Linear Eigenvalue problem: Experiment 4

Since experiment 2 has shown OptiStiefelCGC, PTCG and GDW had a bad performance for the well-conditioned problem, we repeat the experiment changing the problem size. We consider $n = 1000$ and $p = 5$ and the well-conditioned matrix $A = \text{diag}(2 * \text{ones}(n, 1) + \text{rand}(n, 1))$.

Column Fval of Table 5.4 shows that PTCG, OptiStiefelCGC and GDW fail to reach the optimal value of $\mathcal{F}$ for some iterations. Also, these algorithms have a large variance compared to others.

Moreover, PTCG, OptiStiefelCGC and GDW stopped because the maximum number of iterations was reached. Hence, these algorithms are have better performance for ill-conditioned problems of experiments 1 and 3 than for well-conditioned problems of experiments 2 and 4.

Performance of our CTCG method is closer to the performance of OptiStiefelGBB and Ad-Moul but it is still unsatisfactory. Note that variance for CTCG is high for Nitr, Nfe and Time which makes this method unreliable. Furthermore, mean time is 40.83 seconds which is not comparable to 4.03 seconds corresponding to Ad-Moul.

Results summary: Linear Eigenvalue Problem Experiment 4

Ad-Moul and OptiStiefelGBB are the only two methods that solved this problem satisfactorily. All features are similar for these two methods except for computational time. OptiStiefelGBB is faster with mean time of 1.68 while Ad-Moul has a mean time of 4.03.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-1.4991e+01	2.80e-15	3.15e-04	1000.0	1001.0	4.18
	min	-1.4991e+01	1.37e-15	5.16e-05	1000.0	1001.0	4.16
	max	-1.4990e+01	5.01e-15	9.94e-04	1000.0	1001.0	4.22
	var	2.0379e-08	6.71e-31	2.13e-08	0.0	0.0	0.00
CTCG	mean	-1.4991e+01	3.14e-14	1.74e-05	530.1	716.5	40.83
	min	-1.4991e+01	1.19e-15	4.43e-06	246.0	323.0	18.69
	max	-1.4991e+01	9.73e-14	5.99e-05	921.0	1231.0	70.48
	var	3.0512e-16	4.89e-28	1.58e-10	17323.4	31131.5	101.57
GDW	mean	-1.4990e+01	5.50e-16	5.70e-04	1000.0	1292.2	25.52
	min	-1.4991e+01	1.81e-16	1.13e-04	1000.0	1284.0	24.62
	max	-1.4990e+01	1.03e-15	2.79e-03	1000.0	1300.0	26.31
	var	2.7031e-08	2.31e-32	1.46e-07	0.0	13.2	0.11
OptiStiefelGBB	mean	-1.4991e+01	1.24e-15	1.03e-05	399.4	424.0	1.68
	min	-1.4991e+01	4.30e-16	1.46e-06	212.0	232.0	0.92
	max	-1.4991e+01	3.75e-15	4.66e-05	643.0	682.0	2.70
	var	1.2031e-15	3.29e-31	4.32e-11	6597.9	7133.9	0.11
Ad-Moul	mean	-1.4991e+01	2.45e-15	8.67e-06	408.5	431.7	4.03
	min	-1.4991e+01	7.49e-16	9.07e-07	223.0	231.0	2.17
	max	-1.4991e+01	6.60e-15	2.27e-05	671.0	711.0	6.64
	var	1.4407e-15	9.34e-31	9.68e-12	8796.4	9462.2	0.85
OptiStiefelCGC	mean	-1.4991e+01	5.33e-15	2.92e-04	1000.0	1001.0	4.15
	min	-1.4991e+01	2.45e-15	3.93e-05	1000.0	1001.0	4.11
	max	-1.4990e+01	8.21e-15	1.05e-03	1000.0	1001.0	4.22
	var	1.7706e-08	1.32e-30	2.09e-08	0.0	0.0	0.00

Table 5.4: Results of experiment 4 for Linear Eigenvalue Problem

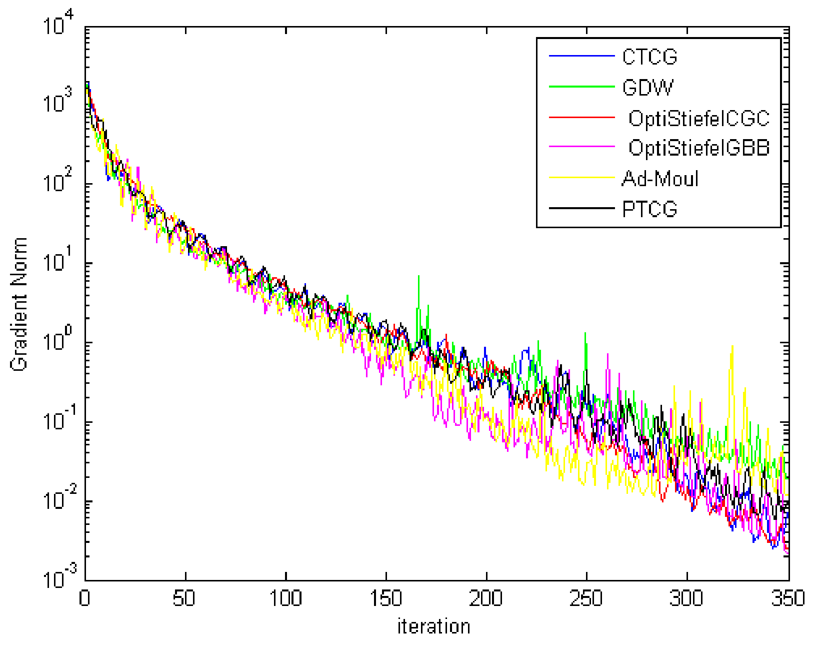

Figure 5.7: Linear Eigenvalue Problem Experiment 4: Average gradient norm

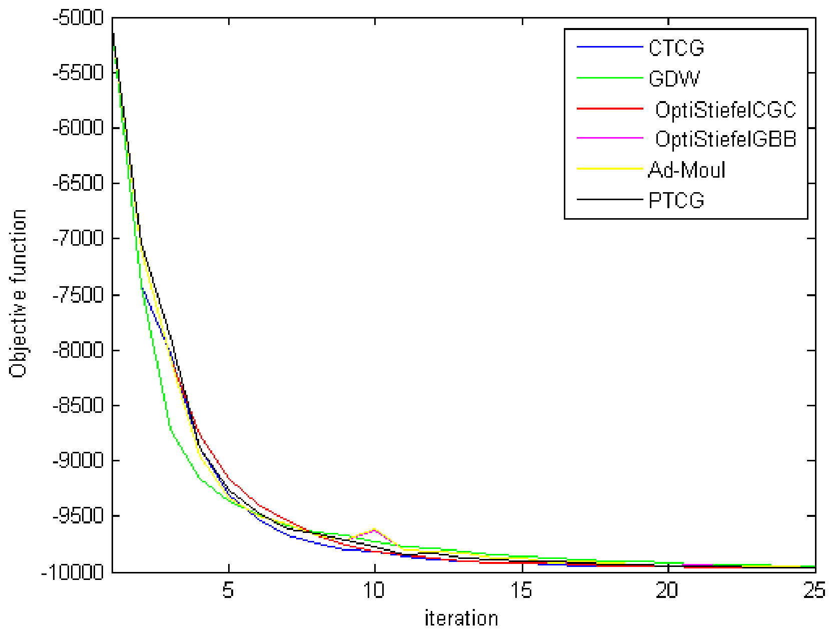

Figure 5.8: Linear Eigenvalue Problem Experiment 4: Average objective function

5.2.5 Linear Eigenvalue problem: Experiment 5

The next experiment has size $n = 500$ and $p = 10$. Symmetric matrix A is not diagonal like in the last four experiments. A full symmetric matrix is built taking $M = \text{randn}(n)$ and $A = M^\top M$.

Results in Table 5.5 show that this problem was more easy to solve for all methods. First column, Fval, shows that all algorithms reached the optimal value in all occasions. Actually, variance for this feature is similar and small for all methods.

Regarding feasibility all methods have values below 1e-13. Also variance is in the order of 1e-30, so that, results for feasibility are satisfactory.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-1.8399e+04	5.86e-15	1.98e-02	111.6	161.6	0.22
	min	-1.8399e+04	3.67e-15	1.97e-03	75.0	101.0	0.14
	max	-1.8399e+04	8.59e-15	8.41e-02	170.0	257.0	0.34
	var	3.5418e-13	1.06e-30	2.28e-04	245.8	693.2	0.00
CTCG	mean	-1.8399e+04	4.01e-15	1.75e-02	112.7	162.1	2.22
	min	-1.8399e+04	2.76e-15	2.90e-03	79.0	108.0	1.51
	max	-1.8399e+04	6.11e-15	7.44e-02	172.0	252.0	3.43
	var	3.3753e-13	4.44e-31	1.73e-04	259.9	721.4	0.12
GDW	mean	-1.8399e+04	1.99e-15	1.19e-02	124.4	189.2	0.45
	min	-1.8399e+04	9.43e-16	3.82e-05	87.0	130.0	0.31
	max	-1.8399e+04	3.08e-15	7.25e-02	206.0	322.0	0.76
	var	3.1074e-12	1.78e-31	1.48e-04	549.2	1457.4	0.01
OptiStiefelGBB	mean	-1.8399e+04	2.04e-15	1.11e-02	104.0	107.8	0.12
	min	-1.8399e+04	5.18e-16	6.45e-04	62.0	65.0	0.08
	max	-1.8399e+04	3.00e-15	1.30e-01	192.0	205.0	0.23
	var	9.7493e-13	2.30e-31	3.12e-04	306.2	350.6	0.00
Ad-Moul	mean	-1.8399e+04	3.86e-15	1.22e-02	106.9	109.6	0.29
	min	-1.8399e+04	2.04e-15	6.58e-04	64.0	66.0	0.18
	max	-1.8399e+04	7.71e-15	1.53e-01	157.0	167.0	0.43
	var	1.7559e-12	1.70e-30	3.29e-04	292.2	344.0	0.00
OptiStiefelCGC	mean	-1.8399e+04	4.70e-15	1.84e-02	106.0	147.2	0.18
	min	-1.8399e+04	3.03e-15	3.77e-03	78.0	104.0	0.13
	max	-1.8399e+04	6.68e-15	9.17e-02	169.0	263.0	0.31
	var	1.3080e-13	6.43e-31	1.74e-04	234.1	682.1	0.00

Table 5.5: Results of experiment 5 for Linear Eigenvalue Problem

Average gradient norm is in order of 1e-2 which is not so small but all algorithms have similar values for this feature. However, our GDW method obtained a minimum value of 3.82e-5 which is smaller than other minimums.

Note that average number of iterations for state-of-the-art algorithms is similar, around 105. Our PTCG and CTCG methods are slightly above this number with an average of 112 iterations. In this sense, our methods are comparable with state-of-the-art algorithms.

Once again, Ad-Moul and OptiStiefelGBB have the smaller number of function evaluations. Our methods are closer to OptiStiefelCGC in this manner.

OptiStiefelGBB is faster than all other algorithms, with an average CPU time of 0.12. Second faster algorithm is OptiStiefelCGC with 0.18 seconds. Third faster algorithm is our PTCG method with 0.22 seconds which is close to the time of OptiStiefelCGC and less than the time of Ad-Moul. In this regard, our PTCG method is competitive with state-of-the-art algorithms.

Figure 5.9 shows behavior of methods without stopping criteria running for 350 iterations. Note that Ad-Moul is the only method that keeps descending over all the process. Our methods and OptiStiefelCGC stop descending around iteration 150.

Results summary: Linear Eigenvalue Problem Experiment 5

Considering results of Table 5.5, the performance of OptiStiefelGBB is best of all. However, OptiStiefelCGC, Ad-Moul and PTCG have competitive performances.

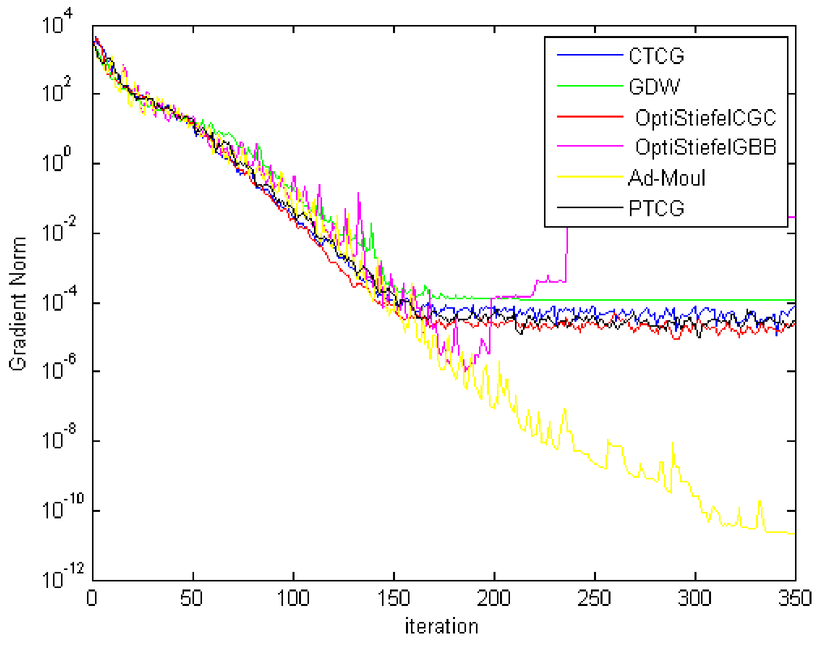

Figure 5.9: Linear Eigenvalue Problem Experiment 5: Average gradient norm

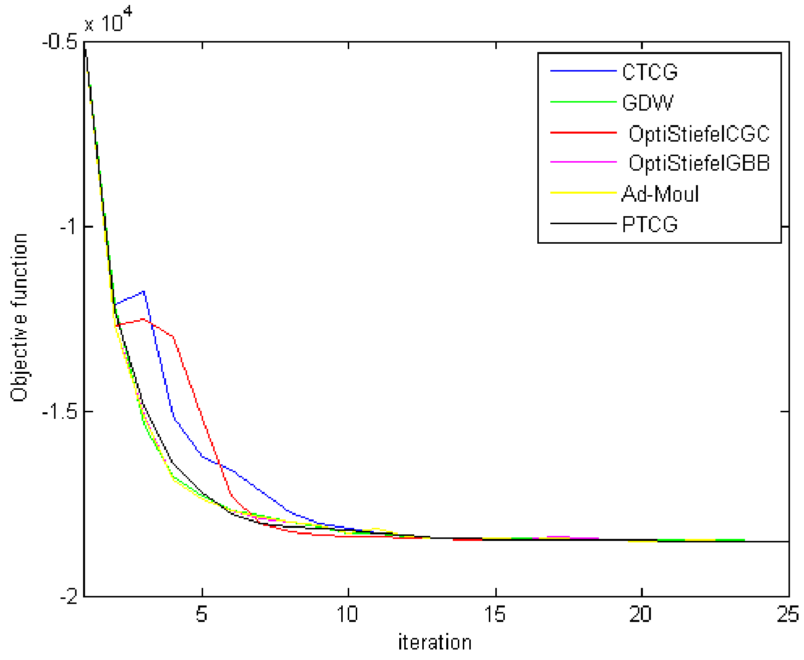

Figure 5.10: Linear Eigenvalue Problem Experiment 5: Average objective function

OptiStiefelGBB, OptiStiefelCGC and Ad-Moul have similar performance considering number of iterations. But regarding computational time, our PTCG method is slightly slower than OptiStiefelCGC and slightly faster than Ad-Moul which makes it competitive for this problem.

On the other hand, Figure 5.9 show Ad-Moul to have better descending properties than other algorithms on asymptotic behavior. Actually, is the only method that did not stop descending over all the process.

5.2.6 Linear Eigenvalue problem: Experiment 6

For this last experiment, we consider $M = \text{randn}(n)$ and $A = M^{\top}M$ but problem size is $n = 1000$ and $p = 5$.

Table 5.6 shows similar results than last experiment, except for computational time. First, Fval column shows that all algorithms reached optimal solutions every time. Also, all algorithms had a small value for variance.

As said before, feasibility of last iterate is always checked and if it is above 1e-13 a modified Gram-Schmidt process is applied in order to reduce it. In this regard, all values in Feasi column are expected to be below 1e-13. Actually, mean values are in the order of 1e-15.

Average gradient norm is in the order of 1e-2 for all methods, however, Ad-Moul had a minimum value of 6.35e-4, one order below others. Note that GDW had a maximum value of 1.08e-1 which is one order above others.

State-of-the-art algorithms have very similar values for number of iterations. The minimum number of iterations belongs to OptiSteifelGBB with 75. Nevertheless, OptiStiefelCGC and Ad-Moul are close with 77 and 78 iterations, respectively. Regarding our methods, the minimum number of iterations for CTCG is 7, same as OptiStiefelCGC. For GDW and PTCG we have 84 and 86 iterations respectively.

A bigger difference appears in the Nfe column since OptiStiefelGBB and Ad-Moul have a minimum of 77 and 79 evaluations, respectively. For our methods, minimum number of function evaluations is above 105 in all cases.

Comparison in computational time shows our PTCG to be competitive with state-of-the-art algorithms since its mean value is 0.68 seconds, above OptiStiefel-CGC which has 0.59 and below Ad-Moul which has 1.05. GDW and CTCG are non comparable in this regard.

Figure 5.11 shows that Ad-Moul have better descending properties since it is the only algorithm that does not stop descending after 200 iterations.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-1.9352e+04	3.12e-15	2.48e-02	114.1	165.9	0.68
	min	-1.9352e+04	1.41e-15	3.20e-03	86.0	114.0	0.47
	max	-1.9352e+04	5.40e-15	9.88e-02	171.0	251.0	1.02
	var	2.1084e-13	6.66e-31	3.05e-04	246.8	663.0	0.01
CTCG	mean	-1.9352e+04	2.47e-15	2.56e-02	111.3	160.8	8.96
	min	-1.9352e+04	1.43e-15	5.12e-03	77.0	108.0	6.11
	max	-1.9352e+04	4.31e-15	7.13e-02	148.0	225.0	12.45
	var	2.8957e-13	4.45e-31	2.39e-04	183.7	498.0	1.40
GDW	mean	-1.9352e+04	1.94e-15	1.73e-02	124.7	190.2	1.82
	min	-1.9352e+04	6.29e-16	6.29e-04	84.0	128.0	1.23
	max	-1.9352e+04	3.78e-15	1.08e-01	187.0	286.0	2.72
	var	3.9038e-12	3.82e-31	4.73e-04	342.8	882.5	0.07
OptiStiefelGBB	mean	-1.9352e+04	2.07e-15	1.56e-02	106.8	110.9	0.44
	min	-1.9352e+04	5.60e-16	1.08e-03	75.0	77.0	0.31
	max	-1.9352e+04	4.53e-15	9.69e-02	156.0	166.0	0.66
	var	1.1801e-12	5.75e-31	3.02e-04	246.5	285.7	0.00
Ad-Moul	mean	-1.9352e+04	2.62e-15	1.73e-02	107.4	110.7	1.05
	min	-1.9352e+04	8.89e-16	6.35e-04	78.0	79.0	0.75
	max	-1.9352e+04	4.75e-15	8.23e-02	151.0	165.0	1.53
	var	1.2080e-12	6.95e-31	3.03e-04	229.1	275.9	0.02
OptiStiefelCGC	mean	-1.9352e+04	3.71e-15	2.98e-02	105.6	144.7	0.59
	min	-1.9352e+04	1.56e-15	5.59e-03	77.0	98.0	0.40
	max	-1.9352e+04	5.85e-15	9.28e-02	134.0	194.0	0.79
	var	2.0494e-13	7.62e-31	3.79e-04	135.1	379.2	0.01

Table 5.6: Results of experiment 6 for Linear Eigenvalue Problem

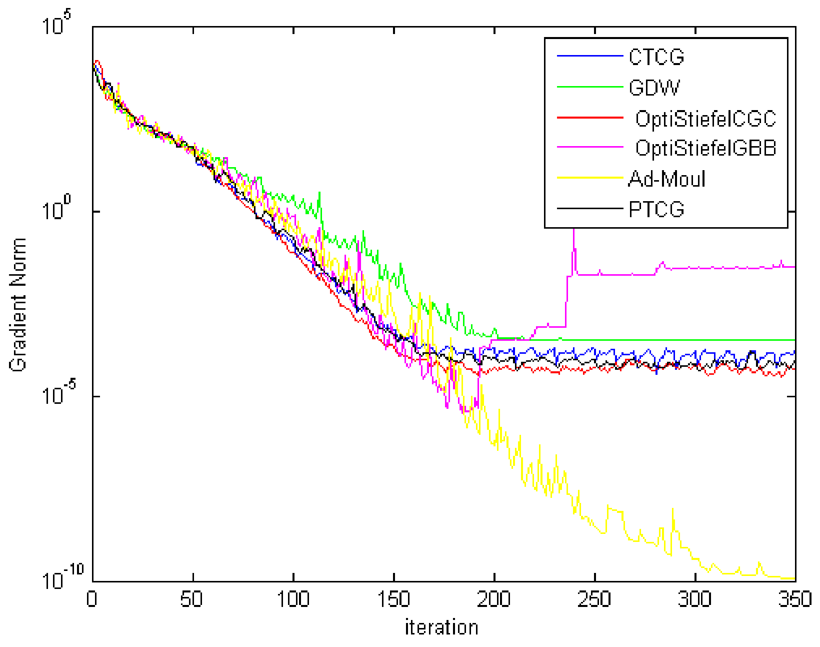

Figure 5.11: Linear Eigenvalue Problem Experiment 6: Average gradient norm

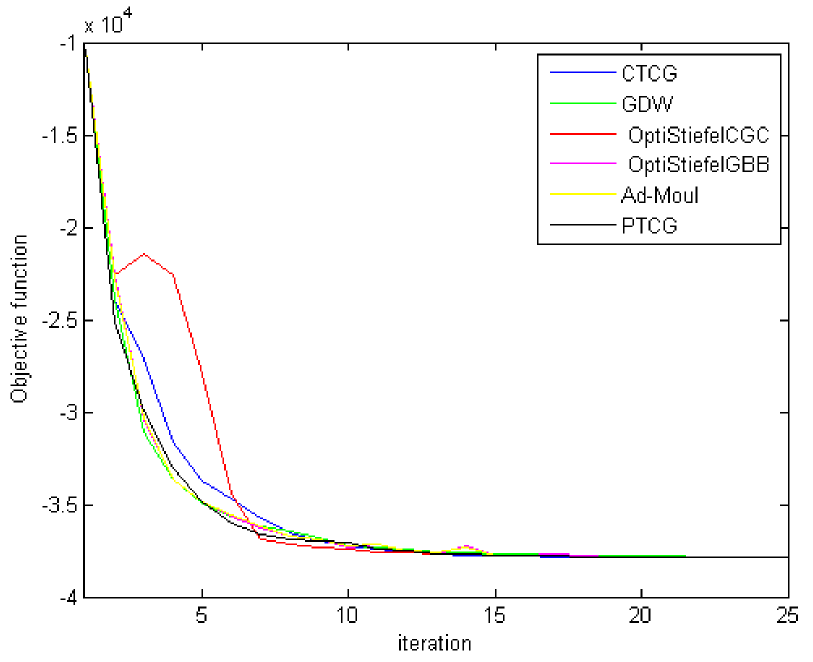

Figure 5.12: Linear Eigenvalue Problem Experiment 6: Average objective function

Results summary: Linear Eigenvalue Problem Experiment 6

This experiment has shown that our PTCG can be competitive with state-of-the-art algorithms regarding computational time. Nevertheless, OptiStiefelGBB has better results than all other algorithms and our methods are still far from showing that level of performance. Asymptotic behavior is better for Ad-Moul which shows great descending properties.

CTCG is shown to have a fine performance without considering computational time.

5.2.7 Results summary: Linear Eigenvalue Problem

Results of all six experiments have shown that performance of GDW, PTCG and OptiStiefelCGC in experiments 2 and 4 (well-conditioned diagonal matrices) was not always satisfactory. For thess experiments, OptiStiefelGBB and Ad-Moul are the methods with best performance. Our CTCG method had a satisfactory performance is we are not to consider computational time, i.e., main disadvantage of this method is its elevate computational cost.

For experiments 1, 3, 5 and 6, all state-of-the-art algorithms have satisfactory performance, being OptiStiefelGBB and Ad-Moul the best. Our methods have a fine performance for these experiments but computational cost of GDW and CTCG makes them non comparable with state-of-the-art algorithms. Nevertheless, our PTCG is shown to be competitive since its computational time is close to that of OptiStiefelGBB, Ad-Moul and OptiStiefelCGC. An important comment is that PTCG has higher number of function evaluations in general, so that, for more complex functions the performance of PTCG could be non satisfactory.

5.3 Orthogonal Procrustes Problem

Orthogonal Procrustes Problem, OPP, is formulated as

$$\min_{X\in\mathbb{R}^{n\times p}} \|AX-B\|_F^2 \quad \text{s.t.} \quad X^\top X = I_p,$$

where $A \in \mathbb{R}^{l\times n}$ and $B \in \mathbb{R}^{l\times p}$.

Since

$$\begin{aligned}\|AX-B\|_F^2 &= \mathrm{Tr}[(AX-B)^\top(AX-B)]\\ &= \mathrm{Tr}[X^\top A^\top AX - 2B^\top AX + B^\top B],\end{aligned}$$

the objective function for this problem is

$$\mathcal{F}(X) = \mathrm{Tr}[X^\top A^\top A X - 2B^\top A X] \tag{5.3}$$

and its gradient is

$$G = 2A^\top A X - 2A^\top B \tag{5.4}$$

(see Appendix B for details).

5.3.1 OPP: Experiment 1

First experiment is taken from [52] where a simple OPP is generated considering $n = 1000$, $p = 5$, $A = I_n$ and $B = \mathrm{ones}(1000, 5)/\mathrm{sqrt}(1000)$.

Results are displayed in Table 5.7. Column Fval shows that all algorithms reached optimal solutions with a small variance, in the order of 1e-26 or smaller.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	5.2786e-01	2.56e-15	1.09e-06	20.3	21.3	0.03
	min	5.2786e-01	1.18e-15	2.06e-07	18.0	19.0	0.02
	max	5.2786e-01	5.54e-15	3.23e-06	26.0	27.0	0.05
	var	4.9365e-26	7.23e-31	4.85e-13	1.5	1.5	0.00
CTCG	mean	5.2786e-01	2.64e-15	4.03e-07	26.9	29.6	1.77
	min	5.2786e-01	1.10e-15	1.05e-07	20.0	22.0	1.30
	max	5.2786e-01	4.40e-15	3.38e-06	31.0	34.0	2.03
	var	1.6427e-26	4.79e-31	1.46e-13	3.0	4.6	0.01
GDW	mean	5.2786e-01	1.42e-14	1.40e-07	18.9	21.5	0.19
	min	5.2786e-01	6.14e-16	8.34e-11	16.0	17.0	0.16
	max	5.2786e-01	9.83e-14	8.61e-07	19.0	22.0	0.21
	var	8.8546e-28	8.81e-28	3.25e-14	0.3	0.9	0.00
OptiStiefelGBB	mean	5.2786e-01	1.22e-14	7.95e-08	11.4	13.5	0.01
	min	5.2786e-01	9.52e-16	6.23e-14	9.0	10.0	0.01
	max	5.2786e-01	9.37e-14	9.44e-07	14.0	17.0	0.06
	var	6.5023e-28	5.46e-28	3.26e-14	1.7	3.5	0.00
Ad-Moul	mean	5.2786e-01	2.40e-15	6.23e-08	11.5	12.8	0.08
	min	5.2786e-01	8.78e-16	1.93e-13	9.0	10.0	0.06
	max	5.2786e-01	4.15e-15	8.20e-07	13.0	16.0	0.14
	var	1.1275e-28	4.71e-31	2.76e-14	1.0	1.8	0.00
OptiStiefelCGC	mean	5.2786e-01	1.27e-14	7.74e-07	18.9	19.9	0.02
	min	5.2786e-01	7.73e-15	2.84e-07	18.0	19.0	0.02
	max	5.2786e-01	1.86e-14	1.63e-06	19.0	20.0	0.04
	var	9.7054e-28	5.45e-30	8.13e-14	0.1	0.1	0.00

Table 5.7: Results of experiment 1 for Orthogonal Procrustes Problem

Note that gradient norm is smaller for this experiment than for all experiments concerning the Linear Eigenvalue Problem. The minimum gradient norm for OptiStiefelGBB is 6.23e-14, being the smallest minimum value for all algorithms. However, our algorithms are far from this value since the minimum gradient norm for PTCG is 2.06e-7, for CTCG is 1.05e-7 and for GDW is 8.34e-11.

Regarding number of iterations, OptiStiefelGBB and Ad-Moul outperformed all other methods since their average number of iterations is around 11.5 and other methods have an average number above 18 iterations. Number of iterations of OptiStiefelGBB and Ad-Moul is close to their number of function evaluations, which means that stepsize estimation is good in these methods and the backtracking procedure is invoked few times. Aside of having large number of iterations, our methods have large number of function evaluations.

Time comparison shows that CTCG and GDW are too expensive and need more time than other methods to solve this simple problem. PTCG is the fastest algorithm among our methods being slightly slower than OptiStiefelCGC and slightly faster than Ad-Moul.

Figure 5.13 shows behavior of gradient norm when we let the algorithms run for 70 iterations, without other stopping criteria. Note that Ad-Moul and OptiStiefel-GBB descend faster than other algorithms. Behavior of our PTCG is close to the behavior of OptiStiefelCGC. Our GDW and CTCG methods stop descending early in the process.

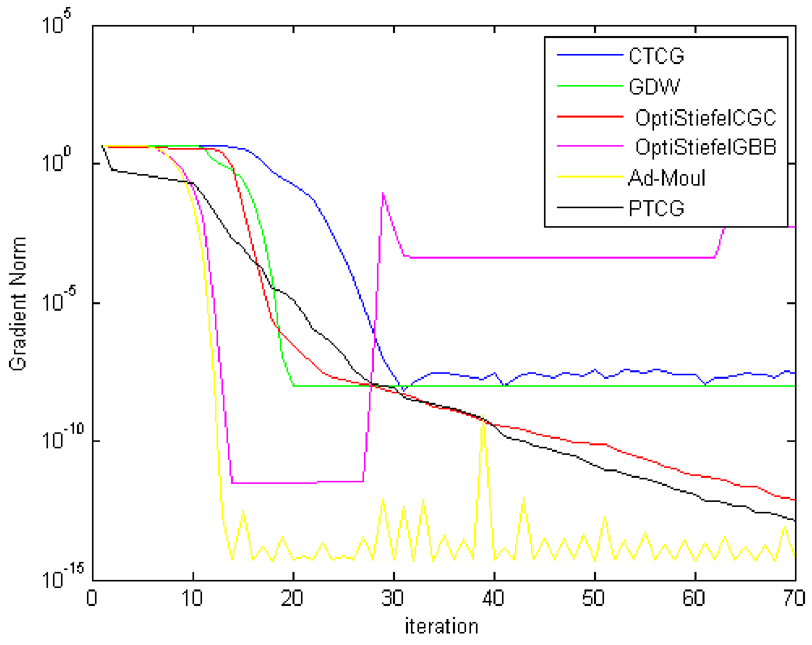

Figure 5.13: OPP 1: Average gradient norm

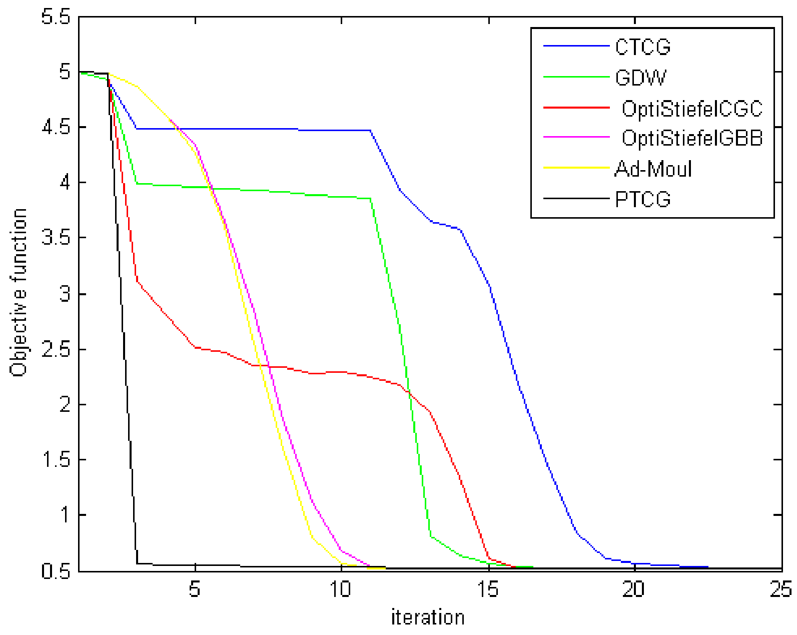

Figure 5.14: OPP 1: Average objective function

Results summary: OPP Experiment 1

OptiStiefelGBB and Ad-Moul outperformed other methods. Main features of these algorithms are small number of iterations and function evaluations. However, OptiStiefelCGC and PTCG methods are competitive regarding computational time. Figure 5.14 shows the objective function descending faster for our PTCG than other methods, during the first ten iterations.

5.3.2 OPP: Experiment 2

Experiment 2 for the Orthogonal Procrustes Problem considers $n = 500$ and $p = 10$. The corresponding matrices are built as $A = \text{randn}(n)/\text{sqrt}(n)$ and $B = \text{randn}(n, p)$.

Results in Table 5.8 show that the value of objective function -4.1054 was always reached by the algorithms with a variance in the order of 1e-20 or smaller.

Gradient norm is comparable for all algorithms. Smallest gradient norm value corresponds to our PTCG method with 4.26e-6. However, largest gradient value also corresponds to one of our algorithms, CTCG.

Comparison concerning number of iterations shows that our PTCG method is close to OptiStiefelGBB and Ad-Moul. While OptiStiefelGBB and Ad-Moul have 17.9 and 18 iterations in average, PTCG has 17.3. For other algorithms this value is larger, for example, CTCG and OptiStiefelCGC need 28.7 and 27.4 iterations, respectively.

Regarding CPU time, our PTCG method have a competitive value with those of the state-of-the-art algorithms. Mean value of 0.02 seconds is above of 0.01, for OptiStiefelGBB, equal to 0.02, for OptiStiefelCGC, and below 0.04, for Ad-Moul.

Figure 5.15 shows asymptotic behavior of gradient norm. Note that Ad-Moul has the best descending rate. Our GDW y CTCG methods practically stop descending after 30 iterations. In this regard, our best method is PTCG which keeps descending but not as much as Ad-Moul.

Results summary: OPP Experiment 2

Due to the small value $p = 10$, our PTCG method is competitive with state-of-the-art algorithms for this experiment. Number of iterations, function evaluations and computational time are similar for OptiStiefelGBB, Ad-Moul and PTCG and also these algorithms have the best performance for this experiment.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-4.1054e+02	6.08e-15	1.71e-05	17.3	19.3	0.02
	min	-4.1054e+02	3.83e-15	4.26e-06	16.0	18.0	0.01
	max	-4.1054e+02	8.82e-15	4.47e-05	20.0	22.0	0.02
	var	2.4091e-23	1.52e-30	1.07e-10	0.3	0.3	0.00
CTCG	mean	-4.1054e+02	3.61e-15	1.09e-04	28.7	40.6	0.54
	min	-4.1054e+02	2.38e-15	3.31e-05	24.0	36.0	0.46
	max	-4.1054e+02	4.89e-15	7.66e-04	34.0	47.0	0.64
	var	4.2851e-20	2.82e-31	8.59e-09	6.2	6.5	0.00
GDW	mean	-4.1054e+02	2.34e-14	8.13e-05	21.3	28.6	0.05
	min	-4.1054e+02	1.26e-14	6.69e-06	19.0	23.0	0.05
	max	-4.1054e+02	4.10e-14	7.11e-04	23.0	33.0	0.07
	var	2.0103e-20	3.59e-29	7.70e-09	1.0	4.6	0.00
OptiStiefelGBB	mean	-4.1054e+02	1.49e-14	3.81e-05	17.9	18.9	0.01
	min	-4.1054e+02	7.17e-15	1.08e-05	17.0	18.0	0.01
	max	-4.1054e+02	2.30e-14	1.24e-04	19.0	20.0	0.01
	var	4.7594e-22	8.16e-30	3.02e-10	0.1	0.1	0.00
Ad-Moul	mean	-4.1054e+02	4.49e-15	6.22e-05	18.0	19.0	0.04
	min	-4.1054e+02	2.35e-15	1.05e-05	17.0	18.0	0.03
	max	-4.1054e+02	7.78e-15	3.17e-04	19.0	20.0	0.05
	var	6.4445e-22	1.84e-30	1.13e-09	0.1	0.1	0.00
OptiStiefelCGC	mean	-4.1054e+02	1.11e-14	2.05e-05	27.4	44.9	0.02
	min	-4.1054e+02	6.91e-15	1.27e-05	22.0	34.0	0.02
	max	-4.1054e+02	1.84e-14	2.95e-05	34.0	58.0	0.03
	var	1.5492e-24	5.12e-30	8.58e-12	5.0	19.6	0.00

Table 5.8: Results of experiment 2 for Orthogonal Procrustes Problem

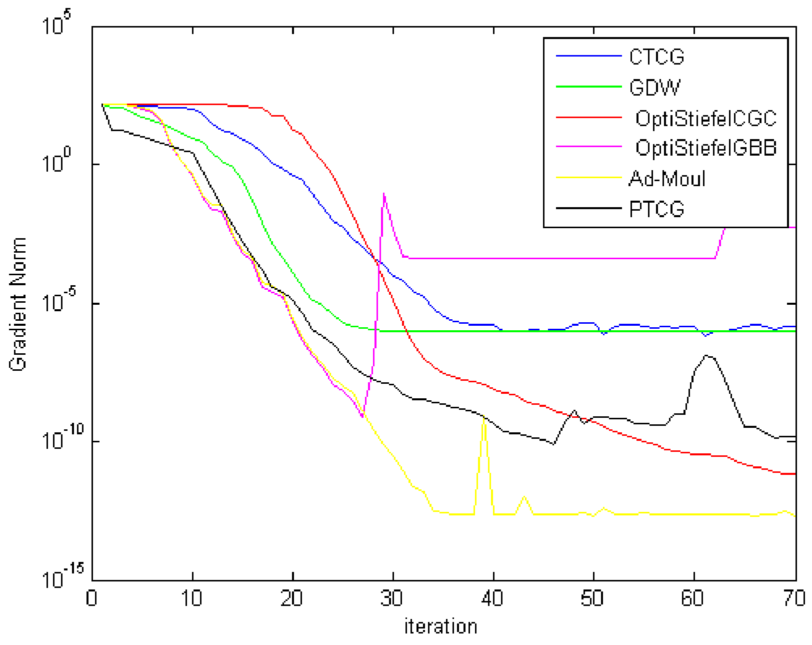

Figure 5.15: OPP 2: Average gradient norm

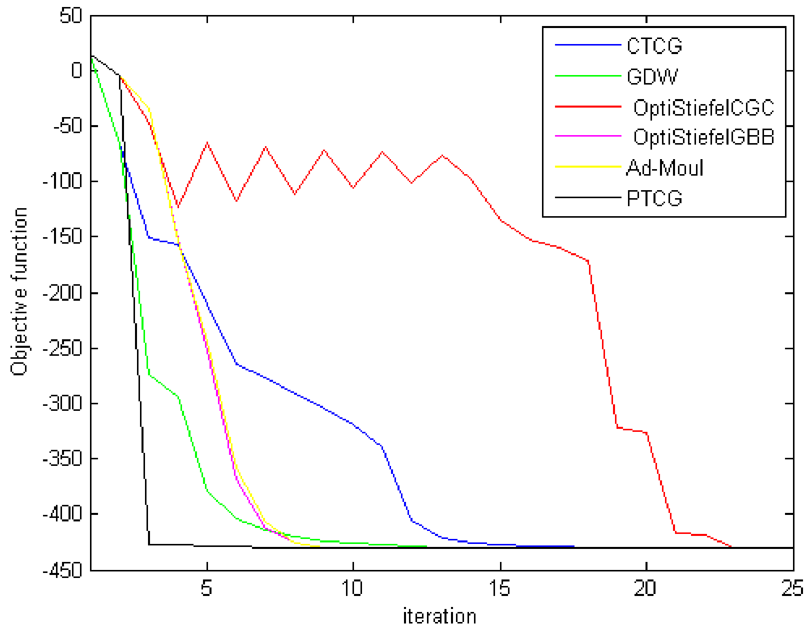

Figure 5.16: OPP 2: Average objective function

5.3.3 OPP: experiment 3

Experiment 3 for the Orthogonal Procrustes Problem considers $n = 1000$ and $p = 5$. The corresponding matrices are built as $A = \text{randn}(n)/\text{sqrt}(n)$ and $B = \text{randn}(n, p)$. Note that difference between this experiment and the last one is the problem size. Parameter n increased but p diminished, so that, our GDW and CTCG methods are expected to take more time solving this problem since their computational cost relies mainly in the value of n.

Note that mean number of iterations is smaller for GDW than PTCG and CTCG, that is, the gradient descent method is faster than the conjugate gradient methods in this regard. However, PTCG and GDW are close to OptiStiefelGBB and Ad-Moul concerning number of iterations.

Note that CTCG and OptiStiefelCGC have the largest mean values of function evaluations, 33.1 and 34.1, respectively. Meanwhile, OptiStiefelGBB, Ad-Moul and PTCG have a mean value of 16.1, 16.2 and 19.3, respectively.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-3.0578e+02	3.17e-15	1.19e-05	17.1	19.3	0.02
	min	-3.0578e+02	1.48e-15	2.39e-06	15.0	18.0	0.02
	max	-3.0578e+02	6.40e-15	3.03e-05	18.0	20.0	0.03
	var	2.5113e-24	7.74e-31	5.15e-11	0.6	0.3	0.00
CTCG	mean	-3.0578e+02	2.46e-15	8.66e-05	24.3	33.1	1.78
	min	-3.0578e+02	1.25e-15	8.23e-06	21.0	30.0	1.56
	max	-3.0578e+02	4.04e-15	7.02e-04	33.0	40.0	2.26
	var	1.1479e-20	4.94e-31	5.55e-09	5.1	4.2	0.02
GDW	mean	-3.0578e+02	9.64e-15	4.03e-05	16.7	21.6	0.17
	min	-3.0578e+02	2.70e-15	3.21e-06	15.0	19.0	0.15
	max	-3.0578e+02	1.70e-14	1.39e-04	19.0	26.0	0.19
	var	5.4846e-22	8.07e-30	1.03e-09	0.6	1.8	0.00
OptiStiefelGBB	mean	-3.0578e+02	7.65e-15	4.39e-05	15.1	16.1	0.01
	min	-3.0578e+02	2.57e-15	2.42e-06	13.0	14.0	0.01
	max	-3.0578e+02	1.55e-14	1.51e-03	18.0	19.0	0.02
	var	1.9128e-19	6.03e-30	2.23e-08	0.2	0.2	0.00
Ad-Moul	mean	-3.0578e+02	2.54e-15	5.17e-05	15.2	16.2	0.10
	min	-3.0578e+02	1.28e-15	3.79e-06	15.0	16.0	0.10
	max	-3.0578e+02	4.72e-15	1.23e-04	18.0	19.0	0.12
	var	7.8154e-22	4.85e-31	6.44e-10	0.2	0.2	0.00
OptiStiefelCGC	mean	-3.0578e+02	6.04e-15	1.66e-05	21.7	34.1	0.03
	min	-3.0578e+02	1.87e-15	2.87e-06	15.0	20.0	0.02
	max	-3.0578e+02	1.39e-14	2.49e-05	40.0	70.0	0.06
	var	3.6461e-24	6.33e-30	8.44e-11	29.4	115.3	0.00

Table 5.9: Results of experiment 3 for Orthogonal Procrustes Problem

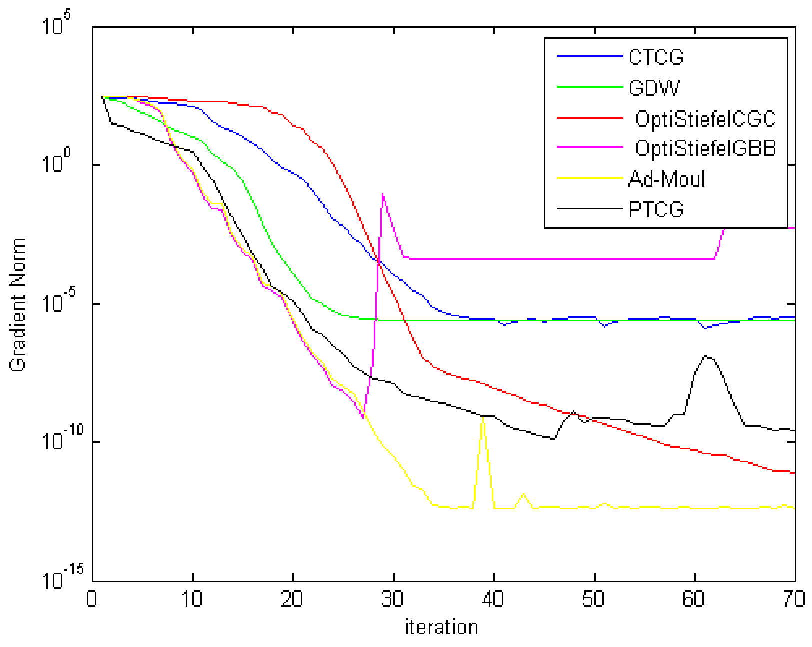

Figure 5.17: OPP 3: Average gradient norm

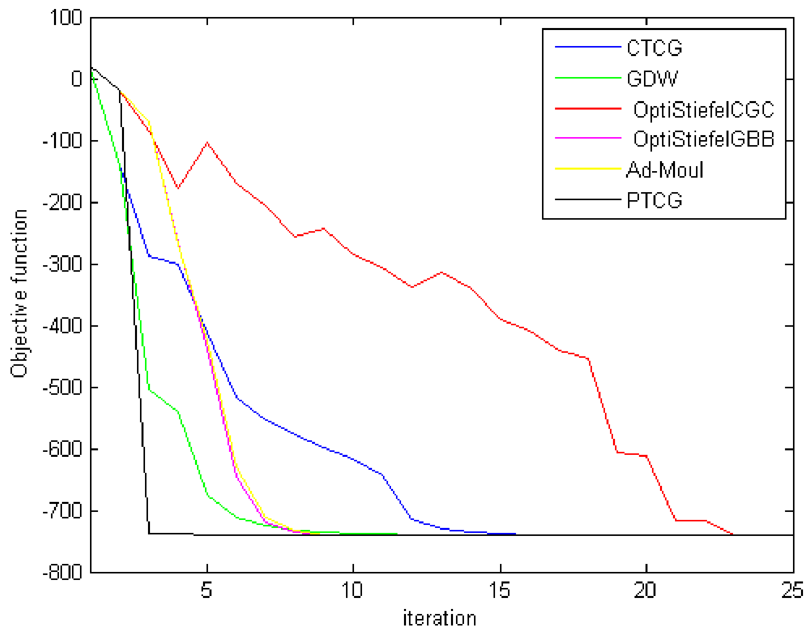

Figure 5.18: OPP 3: Average objective function

Note that, to the set precision, PTCG and OptiStiefelGBB did not incremented computational time with respect to last experiment. This was expected for PTCG since p diminished from 10 to 5 and therefore a smaller matrix have to be inverted. In this regard, PTCG is still comparable with state-of-the-art algorithms for this experiment.

Figures 5.17 and 5.17 are similar to those of last experiment, regarding that Ad-Moul has the best descending rate.

Results summary: OPP Experiment 3

Results for this experiment are very similar to those of the last experiment. Main difference concerns GDW having a small number of iterations, close to state-of-the-art algorithms. As expected, computational cost of GDW makes it not competitive. On the other hand, our PTCG method maintained its features and its competitiveness. Nonetheless, OptiStiefelGBB and Ad-Moul are confirmed to be the best algorithms because of their descending properties.

5.3.4 OPP: Experiment 4

Last experiment of this type also considers $A = \mathrm{randn}(n)/\mathrm{sqrt}(n)$ and $B = \mathrm{randn}(n, p)$ but problem size is increased to $n = 1000$ and $p = 10$.

Table 5.10 shows that all algorithms reached the same value of objective function in all times with a variance lesser than 1e-19.

Note that our PTCG method has the smallest minimum value of gradient norm with 4.43e-6. Nevertheless, all algorithms performed satisfactory in this regard.

Note that, once again, GDW needed a smaller number of iterations than CTCG. Actually, CTCG has the largest mean number of iterations. GDW and PTCG have mean number of iterations of 17.8 and 17.4, respectively, which is not much more than the 15.1 iterations of OptiStiefelGBB. Actually, both our algorithms have values below 21.4, the mean number of iterations of OptiStiefelCGC.

Regarding computational time, PTCG augmented its mean value, with respect to last experiment, but this was expected since p augmented form 5 to 10 and a p^2 linear system has to be solved for each iteration of this method. Nonetheless, the mean value of 0.3 for PTCG is still smaller than the CPU times of OptiStiefelCGC and Ad-Moul.

Figures 5.19 and 5.19 are similar to those of last experiment, regarding that Ad-Moul has the best descending rate.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
	mean	-6.1947e+02	5.94e-15	1.94e-05	17.4	19.4	0.03
PTCG	min	-6.1947e+02	3.76e-15	4.43e-06	15.0	18.0	0.03
	max	-6.1947e+02	8.39e-15	3.95e-05	18.0	20.0	0.04
	var	1.5505e-23	1.18e-30	1.57e-10	0.4	0.3	0.00
	mean	-6.1947e+02	4.15e-15	1.25e-04	24.3	34.1	1.83
CTCG	min	-6.1947e+02	2.71e-15	1.83e-05	21.0	31.0	1.64
	max	-6.1947e+02	6.16e-15	8.36e-04	29.0	40.0	2.19
	var	4.7870e-20	5.03e-31	1.53e-08	3.6	3.1	0.01
	mean	-6.1947e+02	1.82e-14	9.95e-05	17.8	22.4	0.19
GDW	min	-6.1947e+02	8.53e-15	1.08e-05	16.0	20.0	0.17
	max	-6.1947e+02	3.01e-14	8.87e-04	19.0	26.0	0.21
	var	3.0037e-20	2.28e-29	1.59e-08	0.7	1.3	0.00
	mean	-6.1947e+02	1.39e-14	1.13e-04	15.1	16.1	0.02
OptiStiefelGBB	min	-6.1947e+02	6.57e-15	3.42e-05	15.0	16.0	0.02
	max	-6.1947e+02	2.27e-14	1.81e-04	16.0	17.0	0.02
	var	4.1752e-21	7.52e-30	9.49e-10	0.1	0.1	0.00
	mean	-6.1947e+02	3.71e-15	9.89e-05	15.9	16.9	0.11
Ad-Moul	min	-6.1947e+02	2.15e-15	6.11e-06	15.0	16.0	0.10
	max	-6.1947e+02	5.96e-15	1.81e-04	17.0	18.0	0.12
	var	5.1483e-21	6.90e-31	1.36e-09	0.2	0.2	0.00
	mean	-6.1947e+02	9.13e-15	7.24e-06	21.4	32.8	0.04
OptiStiefelCGC	min	-6.1947e+02	4.79e-15	6.17e-06	16.0	22.0	0.03
	max	-6.1947e+02	1.70e-14	8.30e-06	30.0	50.0	0.07
	var	9.1805e-25	5.33e-30	2.38e-13	15.7	62.9	0.00

Table 5.10: Results of experiment 4 for Orthogonal Procrustes Problem

Results summary: OPP Experiment 4

Ad-Moul and OptiStiefelGBB outperformed other methods regarding number of iterations and number of function evaluations. However, CPU time for PTCG and OptiStiefelCGC is sufficiently small to be competitive with Ad-Moul and OptiStiefelGBB. Our GDW method does not perform badly but its computational cost makes it not competitive.

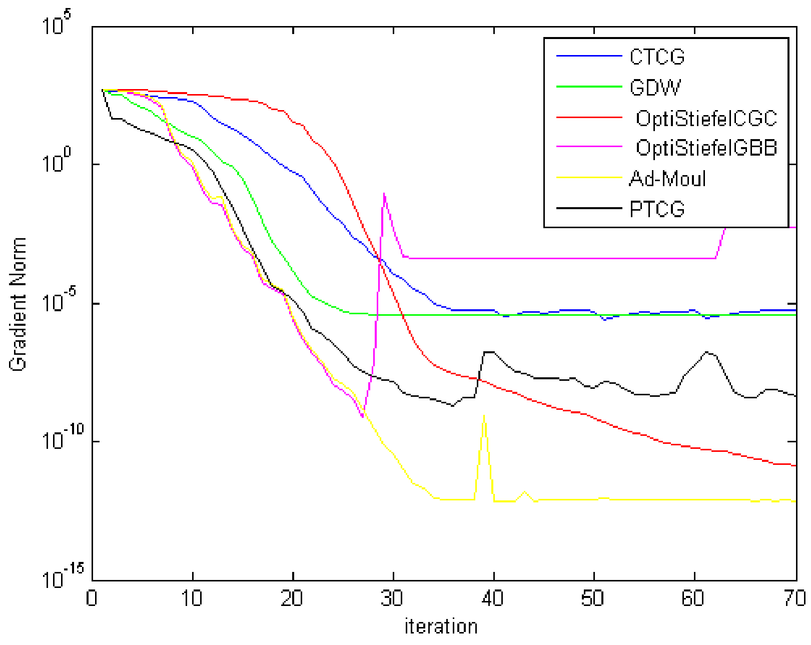

Figure 5.19: OPP 4: Average gradient norm

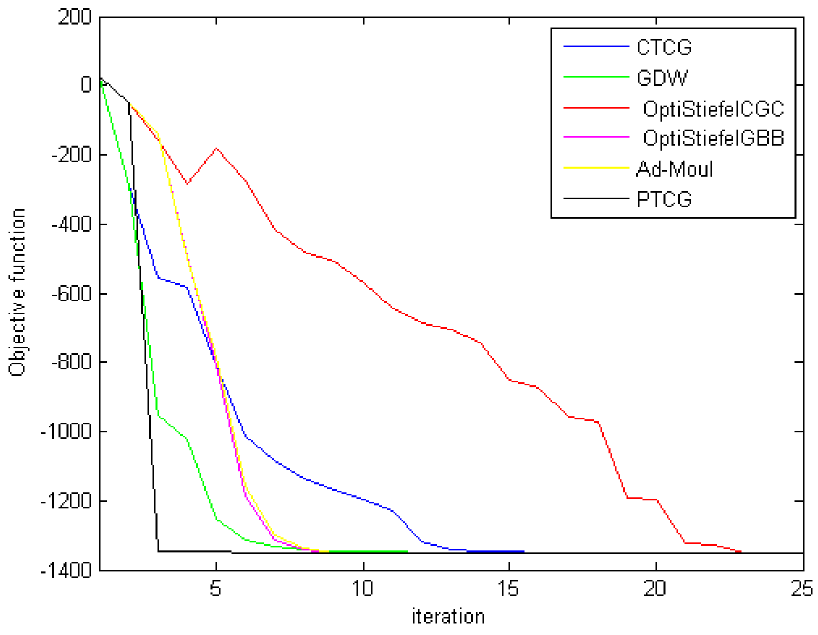

Figure 5.20: OPP 4: Average objective function

5.4 Heterogeneous Quadratics Minimization Problem

Heterogeneous Quadratics Minimization Problem, HQM, is formulated as

$$\min_{X\in\mathbb{R}^{n\times p}} \sum_{i=1}^{p} X_{[i]}^\top A_i X_{[i]} \quad \text{s.t.} \quad X^\top X = I_p,$$

where A_i, $i = 1, ..., p$, are n-by-n symmetric matrices and $X_{[i]}$ denotes the i-th column of X.

The objective function and its gradient are

$$\mathcal{F}(X) = \sum_{i=1}^{p} X_{[i]}^\top A_i X_{[i]} \quad \text{and} \quad G = [2A_i X_{[i]}]_i. \tag{5.5}$$

5.4.1 HQM: experiment 1

This experiment considers matrices A_i defined as in the experiments of [52], that is,

$$A_i = \operatorname{diag}\left(\frac{(i-1)n+1}{p} : \frac{1}{p} : \frac{in}{p}\right). \tag{5.6}$$

For the size problem we consider $n = 500$ and $p = 10$.

Results for this experiment are displayed in Table 5.11. Fval column shows all algorithms to have reached similar values of the objective function with a variance in the order of 1e-13 or less.

Mean values of gradient norm are smaller for OptiStiefelGBB and Ad-Moul, in the order of 1e-4, meanwhile other algorithms have a mean value in the order of 1e-3. However, all algorithms have small variance of gradient norm which is desirable.

Mean number of iterations is very similar for OptiStiefelGBB and OptiStiefel-CGC around 163 iterations. However, Ad-Moul have a lesser number of iterations with 156.7. This value is higher for our methods: 180.9 for PTCG, 172.3 for CTCG and 200.2 for GDW.

One noticeable disadvantage of PTCG, CTCG, GDW and OptiStiefelCGC is the high number of function evaluations. OptiStiefelGBB and Ad-Moul have mean number of function evaluations close to the mean number of iterations, that is, the backtracking procedure is invoked only a few times. Nonetheless, the ratio between number of iterations and number of function evaluations indicate that the backtracking procedure is invoked in every iteration.

Regarding computational time, our CTCG method is expensive as expected with a mean value of 3.54 seconds. This is a large value considering that none of the other algorithms needed more than a second. Our GDW method is faster because of the use of the SMW formula, but it is still not competitive. From our algorithms, only PTCG is fast enough to be competitive with state-of-the-art algorithms. OptiStiefel-GBB is the fastest algorithm with an average of 0.09 seconds. Then OptiStiefelCGC is the second fastest with 0.14 seconds. Our PTCG method takes 0.23 seconds in average and Ad-Moul needs 0.31 seconds.

On the other hand, Ad-Moul has better descending properties as seen in Figure 5.21, actually, is the only algorithms that keeps descending over all the process. Note that our PTCG and CTCG methods are comparable with OptiStiefelCGC in this sense.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	2.2555e+03	6.09e-15	1.21e-03	180.9	272.0	0.23
	min	2.2555e+03	3.80e-15	2.87e-04	118.0	168.0	0.13
	max	2.2555e+03	9.40e-15	4.47e-03	298.0	457.0	0.39
	var	4.3850e-14	1.00e-30	7.27e-07	838.1	2312.5	0.00
CTCG	mean	2.2555e+03	2.94e-14	1.24e-03	172.3	262.4	3.54
	min	2.2555e+03	5.27e-15	2.30e-04	121.0	179.0	2.43
	max	2.2555e+03	9.04e-14	6.54e-03	293.0	445.0	6.02
	var	5.4467e-14	1.60e-28	9.37e-07	843.6	2258.5	0.39
GDW	mean	2.2555e+03	1.42e-15	1.10e-03	200.2	313.4	0.49
	min	2.2555e+03	5.95e-16	8.24e-05	131.0	205.0	0.33
	max	2.2555e+03	3.80e-15	6.92e-03	283.0	451.0	0.70
	var	4.2393e-13	4.00e-31	1.20e-06	1212.0	3248.0	0.01
OptiStiefelGBB	mean	2.2555e+03	3.31e-15	7.25e-04	163.7	176.0	0.09
	min	2.2555e+03	8.64e-16	5.14e-05	110.0	115.0	0.06
	max	2.2555e+03	6.55e-15	6.49e-03	213.0	234.0	0.14
	var	2.5597e-13	1.78e-30	7.50e-07	536.3	686.8	0.00
Ad-Moul	mean	2.2555e+03	7.47e-15	7.76e-04	156.7	166.6	0.31
	min	2.2555e+03	2.89e-15	9.60e-05	110.0	118.0	0.22
	max	2.2555e+03	1.60e-14	5.42e-03	214.0	230.0	0.45
	var	2.2636e-13	7.01e-30	7.61e-07	483.0	575.5	0.00
OptiStiefelCGC	mean	2.2555e+03	5.85e-15	1.18e-03	163.6	243.8	0.14
	min	2.2555e+03	3.75e-15	2.49e-04	114.0	158.0	0.10
	max	2.2555e+03	8.62e-15	5.12e-03	214.0	342.0	0.20
	var	2.4748e-14	9.97e-31	7.86e-07	484.0	1491.0	0.00

Table 5.11: Results of experiment 1 for Heterogeneous Quadratic Minimization

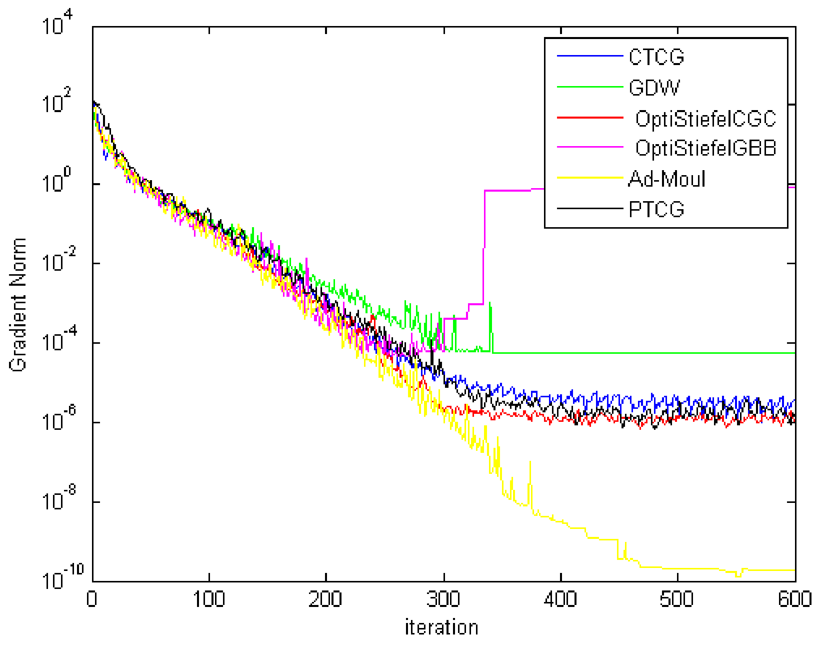

Figure 5.21: HQM 1: Average gradient norm

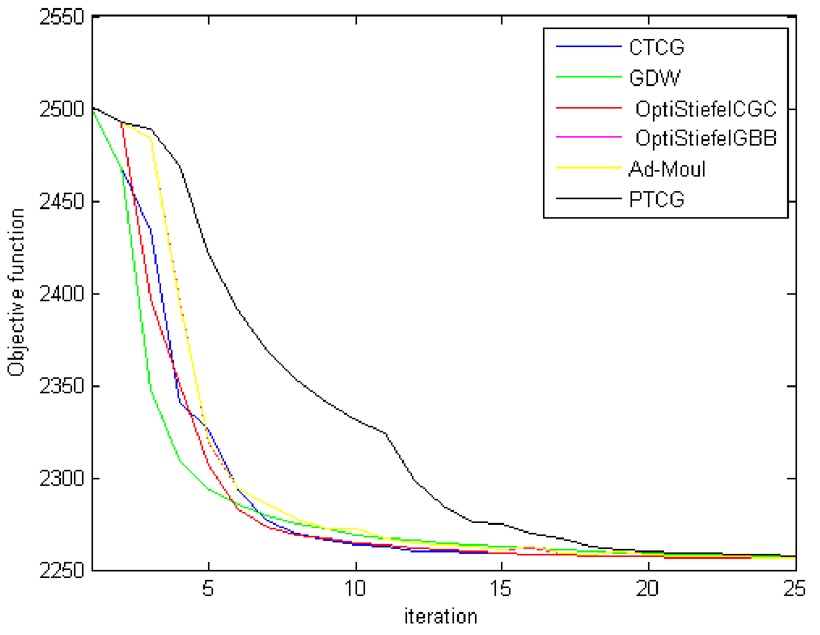

Figure 5.22: HQM 1: Average objective function

Results summary: HQM Experiment 1

OptiStiefelGBB and Ad-Moul outperformed other algorithms. Principal features of OPtiStiefelGBB are the small number of iterations and its small CPU time. Ad-Moul has a larger CPU time but smaller number of iterations. Our CTCG method is comparable in number of iterations but not in computational time. PTCG is competitive regarding computational time and number of iterations.

5.4.2 HQM: experiment 2

For this experiment we keep matrices A_i defined as in (5.6) but change the size problem to $n = 1000$ and $p = 5$.

Similar results to those of the last experiment are expected since only the problem size has changed. Table 5.12 reflects this changes mainly on the number of iterations and computational time.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	2.0030e+03	3.04e-15	2.12e-03	269.3	426.8	0.26
	min	2.0030e+03	1.59e-15	4.00e-04	188.0	280.0	0.16
	max	2.0030e+03	5.57e-15	6.81e-03	426.0	679.0	0.39
	var	2.1287e-13	7.12e-31	2.07e-06	1642.2	4564.0	0.00
CTCG	mean	2.0030e+03	1.95e-14	2.55e-03	261.8	413.1	21.14
	min	2.0030e+03	6.27e-15	3.32e-04	185.0	293.0	15.02
	max	2.0030e+03	7.48e-14	1.31e-02	368.0	588.0	29.88
	var	3.1958e-13	1.13e-28	4.65e-06	1441.0	3994.3	10.16
GDW	mean	2.0030e+03	1.24e-15	1.74e-03	275.8	444.1	2.44
	min	2.0030e+03	3.48e-16	1.16e-04	191.0	301.0	1.70
	max	2.0030e+03	5.03e-15	6.90e-03	422.0	694.0	3.70
	var	1.7207e-12	5.49e-31	1.63e-06	2046.0	5960.4	0.16
OptiStiefelGBB	mean	2.0030e+03	2.53e-15	1.44e-03	232.4	249.3	0.10
	min	2.0030e+03	4.53e-16	2.13e-04	172.0	181.0	0.07
	max	2.0030e+03	6.80e-15	1.08e-02	360.0	385.0	0.17
	var	1.0667e-12	1.86e-30	1.85e-06	1395.7	1680.7	0.00
Ad-Moul	mean	2.0030e+03	4.06e-15	1.41e-03	226.9	241.8	1.37
	min	2.0030e+03	1.14e-15	1.29e-04	171.0	177.0	1.00
	max	2.0030e+03	1.13e-14	6.96e-03	297.0	319.0	1.82
	var	9.8438e-13	4.43e-30	1.58e-06	761.9	884.6	0.03
OptiStiefelCGC	mean	2.0030e+03	4.24e-15	2.67e-03	242.7	377.2	0.19
	min	2.0030e+03	1.88e-15	5.27e-04	165.0	233.0	0.12
	max	2.0030e+03	7.04e-15	1.42e-02	375.0	604.0	0.32
	var	7.0161e-14	1.05e-30	4.11e-06	1245.4	3677.6	0.00

Table 5.12: Results of experiment 2 for Heterogeneous Quadratic Minimization

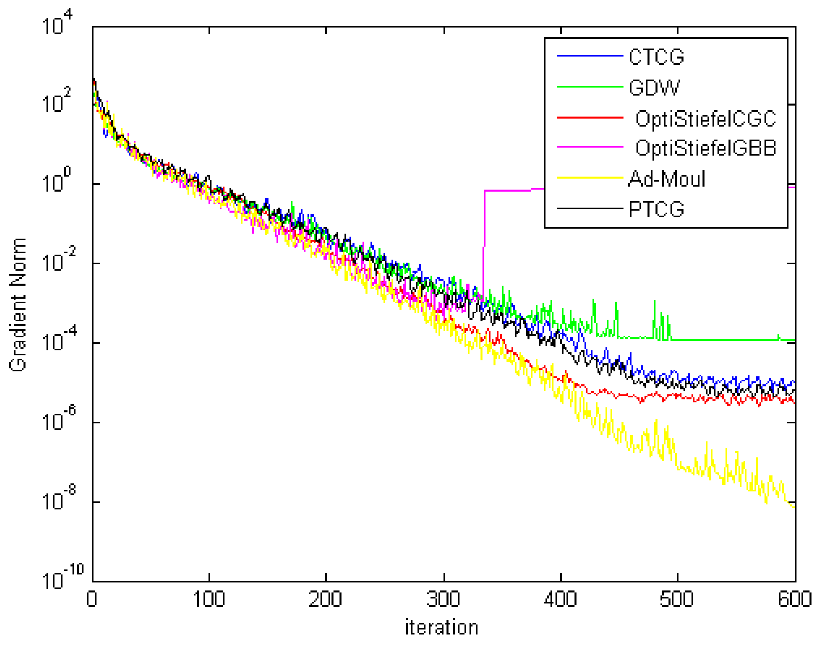

Figure 5.23: HQM 2: Average gradient norm

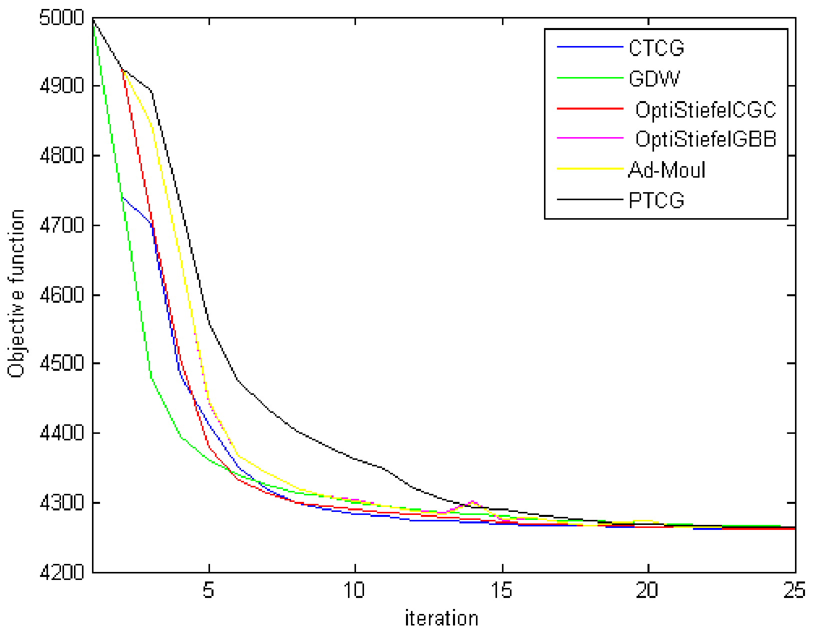

Figure 5.24: HQM 2: Average objective function

Fval column shows all algorithms to have reached similar values of the objective function with a variance in the order of 1e-12 or less. However, OptiStiefelCGC has the smallest variance with 7.01e-14.

Values for gradient norm are similar for all algorithms, with a mean gradient norm in the order of 1e-3.

Larger number of iterations is expected with respect to the last experiment since the size of the problem increased. The smallest mean value belongs to Ad-Moul with 226.9 iterations. Considering only our methods, CTCG have the smallest mean value with 261.8 iterations. In general, state-of-the-art algorithms needed a smaller number of iterations than our methods to solve the problem.

Note that maximum value of Time for CTGC is 29.88 seconds, so that, this method has become extremely expensive for this problem. Our GDW reduces the computational time to 2.44 seconds in average but it is still not competitive. Concerning this feature, PTCG is the only one of our algorithms that can be competitive with state-of-the-art methods. The mean value of 0.26 for PTCG is close to 0.19 corresponding to OptiStiefelCGC.

Average behavior of gradient norm is shown in Figure 5.23 where it is seen that Ad-Moul has better descending properties than all other algorithms. Our two conjugate gradient methods have similar behavior and end close to OptiSteifelCGC.

Results summary: HQM Experiment 2

For this experiment, behavior of algorithms were similar to that of the last experiment. However, incrementing the problem size did increment a lot the cost of our CTCG which has a mean Time of 21.14 seconds. Ad-Moul and OptiStiefelGBB were the best methods and our PTCG method is comparable regarding computational time.

5.4.3 HQM: experiment 3

This experiment adds a random element to the matrices A_i considering

$$A_i = \operatorname{diag}\left(\frac{(i-1)n+1}{p} : \frac{1}{p} : \frac{in}{p}\right) + B_i + B_i^\top, \tag{5.7}$$

where B_is were random matrices generated by $B_i = 0.1\text{randn}(n)$. First size problem is $n = 500$ and $p = 10$.

Results in Table 5.13 show that all algorithms reached the value of 4.7049e2 for the objective function, with a variance in the order of 1e-14 or lesser.

Note that state-of-the-art algorithms have a minimum value of gradient norm in the order of 1e-5 while our algorithms have a minimum value of gradient norm in the order of 1e-4.

For this experiment, OptiStiefelCGC have the smaller mean number of iterations with 251.4. Our PTCG and CTCG methods have 272.3 and 278.7, respectively. These values are similar to those of OptiStiefelGBB and Ad-Moul which are 271.1 and 270.3, respectively.

Note that mean values for Time column are all above 5 seconds. Slowest algorithms are CTCG and GDW with 13.28 and 11.91, respectively. Our PTCG method is faster with a value of 7.46 seconds but still not faster than state of the art algorithms. OptiStiefelGBB is the fastest algorithm followed by Ad-Moul.

Once again, Figure 5.25 shows that Ad-Moul have better descending properties since it does not stop descending when the algorithms run for 600 iterations. Our PTCG and CTCG descend slower but are close to OptiStiefelCGC.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	4.7049e+02	5.80e-15	5.69e-04	272.3	418.9	7.46
	min	4.7049e+02	3.59e-15	1.41e-04	207.0	308.0	5.53
	max	4.7049e+02	8.73e-15	1.79e-03	376.0	587.0	10.48
	var	5.0867e-15	1.33e-30	1.54e-07	1583.4	4077.6	1.27
CTCG	mean	4.7049e+02	1.99e-14	5.55e-04	278.7	437.8	13.28
	min	4.7049e+02	3.97e-15	1.04e-04	204.0	315.0	9.59
	max	4.7049e+02	8.35e-14	2.05e-03	381.0	603.0	18.28
	var	4.7732e-15	1.23e-28	1.52e-07	1160.2	3032.8	2.75
GDW	mean	4.7049e+02	2.79e-15	5.52e-04	401.2	634.0	11.91
	min	4.7049e+02	1.11e-15	1.54e-04	287.0	447.0	8.33
	max	4.7049e+02	4.31e-15	1.65e-03	593.0	937.0	17.66
	var	6.1842e-14	5.70e-31	1.23e-07	4833.3	12594.5	4.51
OptiStiefelGBB	mean	4.7049e+02	2.90e-15	3.35e-04	271.1	289.0	5.05
	min	4.7049e+02	1.33e-15	8.64e-05	212.0	221.0	3.85
	max	4.7049e+02	4.13e-15	1.28e-03	368.0	389.0	6.69
	var	3.2837e-14	4.68e-31	5.63e-08	975.3	1056.4	0.32
Ad-Moul	mean	4.7049e+02	5.34e-15	3.47e-04	270.3	287.0	5.49
	min	4.7049e+02	2.53e-15	6.61e-05	216.0	229.0	4.36
	max	4.7049e+02	9.35e-15	1.36e-03	370.0	389.0	7.41
	var	2.9616e-14	2.42e-30	7.19e-08	1016.5	1086.1	0.40
OptiStiefelCGC	mean	4.7049e+02	5.31e-15	6.03e-04	251.4	387.9	6.82
	min	4.7049e+02	3.67e-15	8.29e-05	194.0	291.0	5.14
	max	4.7049e+02	8.23e-15	2.88e-03	333.0	529.0	9.34
	var	2.6015e-15	1.20e-30	2.41e-07	798.4	2426.3	0.76

Table 5.13: Results of experiment 3 for Heterogeneous Quadratic Minimization

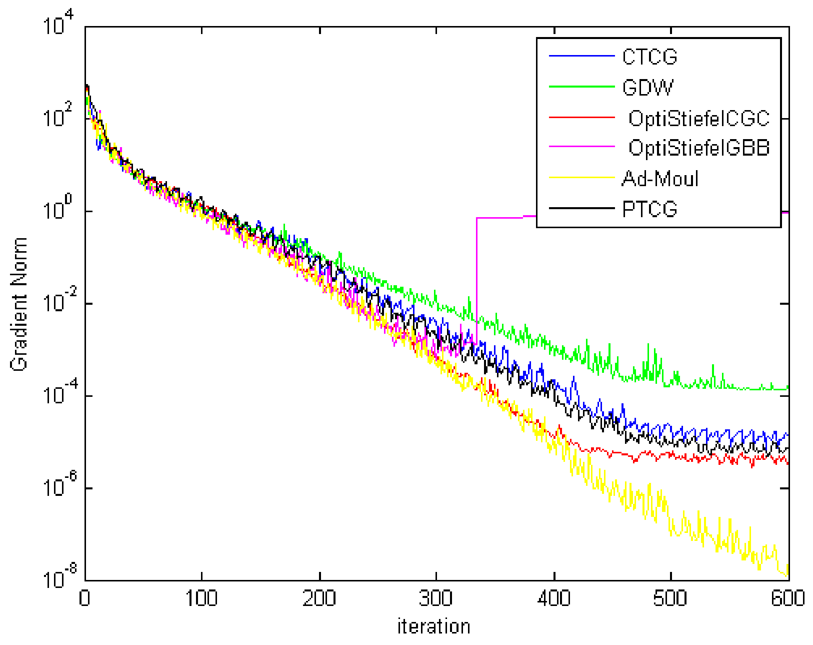

Figure 5.25: HQM 3: Average gradient norm

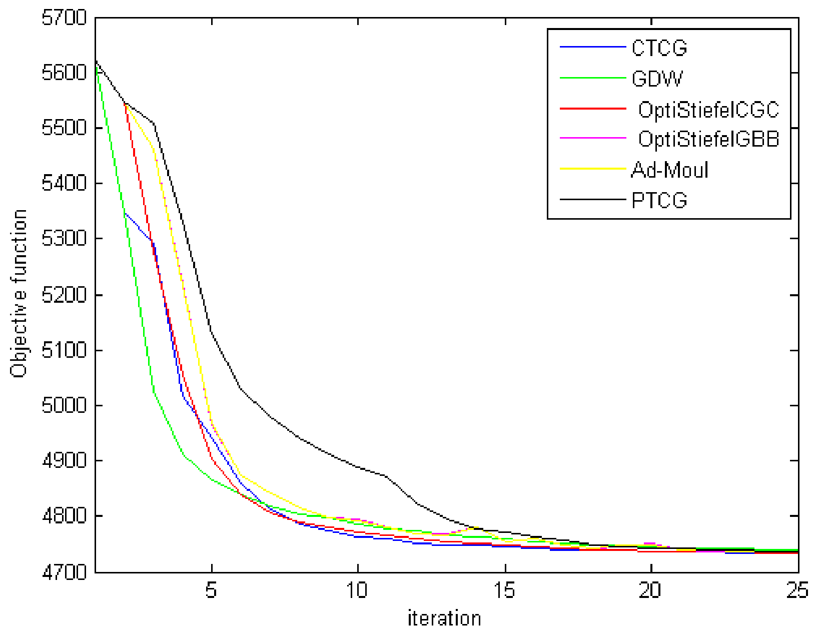

Figure 5.26: HQM 3: Average objective function

Results summary: HQM Experiment 3

None of our algorithms outperformed state-of-the-art algorithms. However, performance of PTCG is satisfactory regarding number of iterations and CPU time. Nonetheless, OptiStiefelGBB and Ad-Moul have the best performance for this experiment, with smaller number of iterations and CPU time.

5.4.4 HQM: experiment 4

For this experiment we consider a problem size of $n = 1000$ and $p = 5$. The matrices A_i, $i = 1, ..., p$, are defined as

$$A_i = \text{diag}\left(\frac{(i-1)n+1}{p} : \frac{1}{p} : \frac{in}{p}\right) + B_j + B_j^\top,$$

where $B_j = \text{randn}(n)$.

Results of Table 5.14 show a small variance for all algorithms regarding the value of the objective function. Our PTCG method have the minimum value of variance, 0.2463e-14, while all other algorithms have a value in the order of 1e-13.

Regarding gradient norm OptiStiefelGBB and Ad-Moul have the smallest mean values with 9.32e-4 and 9.88e-4, respectively. This value is slightly higher for all other algorithms, in the order of 1e-3.

OptiStiefelCGC has the smallest mean number of iterations. However, the number of function evaluations is high and this makes it a solwer algorithm, regarding computational time. For OPtiStiefelGBB and Ad-Moul the number of function evaluations are close to the number of iterations, that is, stepsize estimation is better for these algorithms. Hence, these are the fastest algorithms.

Concerning our methods, CTCG became to expensive for this experiment since it could take up to 96.92 seconds to solve an instance of the problem. GDW reduces CPU time to a maximum of 59.46 seconds but is still slow. Only our PTCG method has a comparable CPU time with a mean value of 26.53 seconds. However, the number of function evaluations for PTCG is still larger than those of state-of-the-art algorithms.

Note that Figure 5.27 shows less difference in the behavior of all algorithms, compared with last experiment. That is, the reached values of gradient norm are not so different between algorithms.

Results summary: HQM Experiment 4

OptiStiefelGBB and Ad-Moul outperformed the other algorithms. Mainly, number of function evaluations slowed down our algorithms and OptiStiefelCGC. Our PTCG method have a mean CPU time of 26.53 which is not far from the 23.69 seconds corresponding to OptiStiefelCGC. However, variance of Time for our PTCG method is high making it less reliable.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	5.8336e+02	3.26e-15	1.81e-03	442.0	712.2	26.53
	min	5.8336e+02	1.41e-15	4.45e-04	303.0	471.0	17.51
	max	5.8336e+02	6.85e-15	4.77e-03	747.0	1214.0	45.30
	var	9.2463e-14	1.22e-30	1.10e-06	8776.7	24647.0	34.21
CTCG	mean	5.8336e+02	1.45e-14	2.04e-03	436.0	703.2	61.53
	min	5.8336e+02	6.73e-15	5.73e-04	300.0	472.0	41.50
	max	5.8336e+02	3.60e-14	4.67e-03	686.0	1109.0	96.92
	var	8.2006e-13	3.32e-29	1.35e-06	5400.7	14951.0	112.45
GDW	mean	5.8336e+02	2.71e-15	1.43e-03	577.8	936.0	40.30
	min	5.8336e+02	5.88e-16	3.25e-04	352.0	558.0	24.26
	max	5.8336e+02	4.82e-15	5.15e-03	853.0	1385.0	59.46
	var	8.3931e-13	6.23e-31	1.23e-06	12904.2	35195.8	64.33
OptiStiefelGBB	mean	5.8336e+02	2.71e-15	9.32e-04	398.9	421.4	15.64
	min	5.8336e+02	1.30e-15	1.77e-04	281.0	297.0	11.00
	max	5.8336e+02	5.90e-15	2.63e-03	554.0	577.0	21.31
	var	5.1096e-13	8.45e-31	3.13e-07	4202.9	4609.8	6.29
Ad-Moul	mean	5.8336e+02	4.02e-15	9.88e-04	401.9	423.7	18.16
	min	5.8336e+02	1.99e-15	2.06e-04	286.0	302.0	12.90
	max	5.8336e+02	9.64e-15	4.16e-03	594.0	610.0	26.38
	var	5.2338e-13	2.29e-30	5.60e-07	4263.6	4518.2	8.42
OptiStiefelCGC	mean	5.8336e+02	3.62e-15	1.61e-03	394.4	633.1	23.69
	min	5.8336e+02	2.04e-15	2.87e-04	280.0	440.0	16.40
	max	5.8336e+02	6.27e-15	5.19e-03	527.0	860.0	32.20
	var	6.4149e-13	1.06e-30	1.21e-06	3202.2	8880.0	12.44

Table 5.14: Results of experiment 4 for Heterogeneous Quadratic Minimization

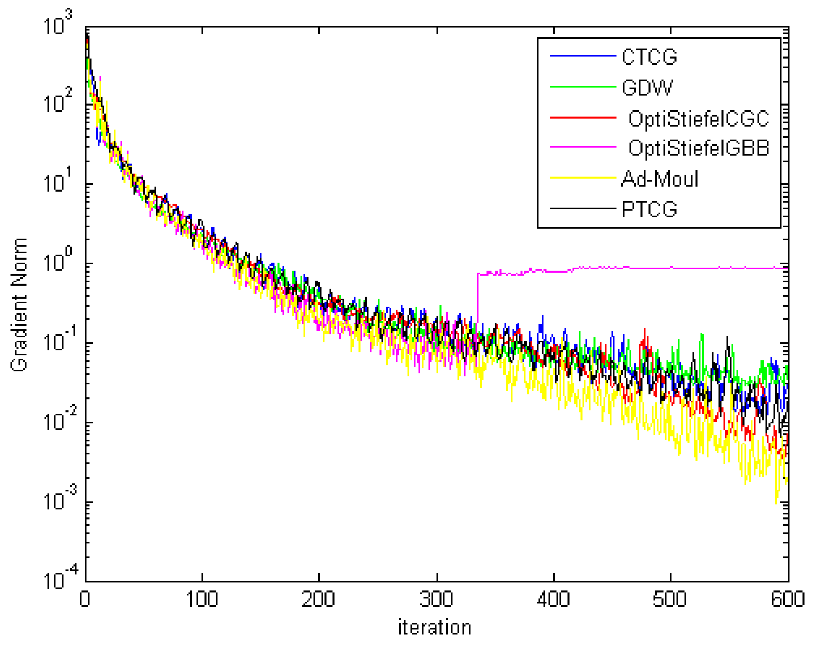

Figure 5.27: HQM 4: Average gradient norm

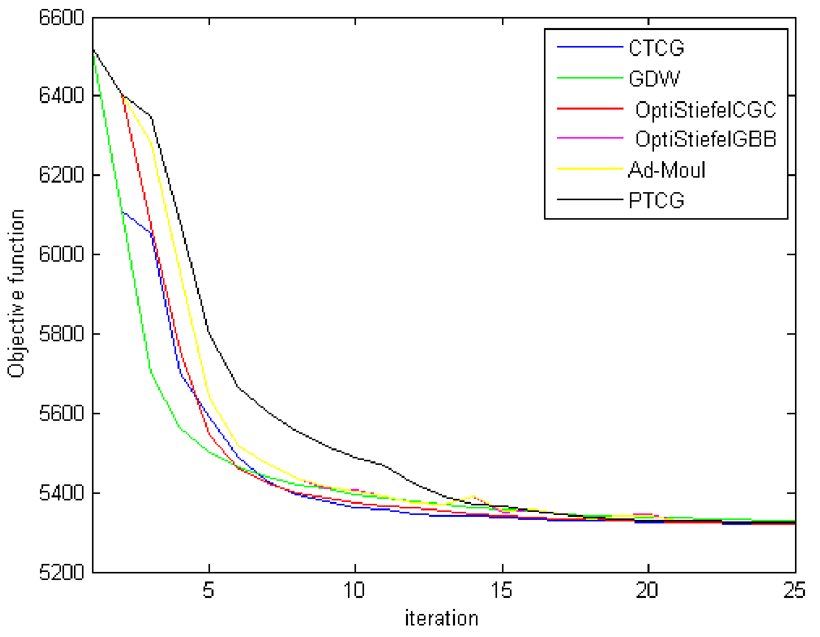

Figure 5.28: HQM 4: Average objective function

5.5 Joint Diagonalization Problem

In this section we solve some simulated instances of the Joint Diagonalization Problem, JDP, formulated as

$$\max_{X\in\mathbb{R}^{n\times p}} \sum_{j=1}^{N} \|\mathrm{diag}(X^\top A_j X)\|_2^2 \quad \text{s.t.} \quad X^\top X = I_p,$$

where A_j, $j = 1, ..., N$, are symmetric matrices.

Objective function and gradient corresponding to this problem are

$$\mathcal{F}(X) = -\sum_{j=1}^{N} \|\mathrm{diag}(X^\top A_j X)\|_2^2 \quad \text{and} \quad G = \left[-4\sum_{j=1}^{N}(X_{[i]}^\top A_j X_{[i]})A_j X_{[i]}\right]. \tag{5.8}$$

5.5.1 JDP: experiment 1

Data matrices A_js for this problem are

$$A_j = \mathrm{diag}(\sqrt{n+1}, \sqrt{n+2}, \cdots, \sqrt{2n}) + B_j + B_j^\top, \tag{5.9}$$

where B_js are random matrices generated by $B_j = \mathrm{randn}(n)$.

For the first experiment we consider $n = 500$, $p = 3$ and $N = 3$.

From the results in Table 5.15 we can deduce that this problem requires more computational effort than the former ones, since we are considering a small problem size and the smallest minimum CPU time is 1.94 seconds, for Ad-Moul.

Moreover, the Fval column shows that algorithms failed to reach the optimum value of the objective function in all repetitions. Actually, variance is above one for all algorithms, which never happened on past experiments.

We can also note that mean gradient norm is in the order of 1e-1 which can be considered a large value for this feature. The smallest mean gradient norm are obtained by GDW and OptiStiefelGBB with 9.65e-2 and 9.23e-2, respectively.

Regarding number of iterations, OptiStiefelGBB and Ad-Moul have the smallest mean values with 237.0 and 236.1, respectively. Nevertheless, oue CTCG method has a mean number of 247.5 iterations which is not far from those of OptiStiefelGBB and Ad-Moul, actually, is less than 328.2, the corresponding value for OptiStiefelCGC.

Moreover, the mean number of function evaluations is 385.2 for CTCG and 521.4 for OptiStiefelCGC, that is, our CTCG method have fine performance in this experiment except for the CPU time.

Regarding Time, OptiStiefelGBB and Ad-Moul are the fastest algorithms. Time difference between these algorithms and the others is noticeable. For example, the fastest of our methods, PTCG, have a mean time value of 6.11 while Ad-Moul have a mean time value of 3.05 seconds.

In Figure 5.29 it can be seen that Ad-Moul and OPtiStiefelGBB have better descending rates than other algorithms. However, for this experiment our CTCG method have a better descending rate than OptiStiefelCGC and is close to OptiStiefelGBB and Ad-Moul for the first iterations.

Results summary: JDP Experiment 1

Our CTCG method is competitive with OptiStiefelCGC if computational time is not considered. Considering computational time only our PTCG method is comparable with state-of-the-art algorithms.

Ad-Moul and OptiStiefelGBB are again the best algorithms, regarding all features.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
	mean	-3.5832e+04	1.78e-15	1.83e-01	363.7	580.4	6.11
PTCG	min	-3.5832e+04	5.05e-16	2.79e-02	214.0	328.0	3.50
	max	-3.5830e+04	3.78e-15	5.15e-01	503.0	820.0	8.48
	var	1.4172e+00	5.27e-31	1.59e-02	6304.7	18245.8	1.98
	mean	-3.5831e+04	1.75e-15	1.30e-01	247.5	385.2	9.19
CTCG	min	-3.5832e+04	9.36e-16	1.82e-02	173.0	265.0	6.31
	max	-3.5830e+04	3.22e-15	3.71e-01	327.0	527.0	12.46
	var	1.5671e+00	3.88e-31	6.28e-03	1318.2	3751.7	2.04
	mean	-3.5832e+04	1.02e-15	9.65e-02	342.3	540.4	6.41
GDW	min	-3.5832e+04	3.15e-16	2.65e-02	215.0	336.0	4.00
	max	-3.5830e+04	2.17e-15	1.63e-01	466.0	768.0	9.06
	var	1.4172e+00	2.66e-31	1.43e-03	4406.0	11788.0	1.64
	mean	-3.5832e+04	9.65e-16	9.23e-02	237.0	256.3	2.70
OptiStiefelGBB	min	-3.5832e+04	3.84e-17	2.08e-02	173.0	188.0	1.99
	max	-3.5830e+04	2.52e-15	3.14e-01	341.0	374.0	3.88
	var	1.4997e+00	2.89e-31	4.16e-03	2283.9	2660.1	0.29
	mean	-3.5832e+04	1.23e-15	1.08e-01	236.1	254.2	3.05
Ad-Moul	min	-3.5832e+04	1.27e-16	2.89e-02	148.0	159.0	1.94
	max	-3.5830e+04	2.53e-15	3.23e-01	306.0	333.0	3.97
	var	1.4172e+00	2.99e-31	6.37e-03	1650.6	1985.2	0.28
	mean	-3.5832e+04	2.31e-15	1.53e-01	328.2	521.4	5.50
OptiStiefelCGC	min	-3.5832e+04	6.78e-16	2.13e-02	216.0	334.0	3.51
	max	-3.5830e+04	3.87e-15	4.33e-01	495.0	790.0	8.38
	var	1.2072e+00	7.09e-31	1.19e-02	5479.6	15114.2	1.63

Table 5.15: Results of experiment 1 for Joint Diagonalization Problem

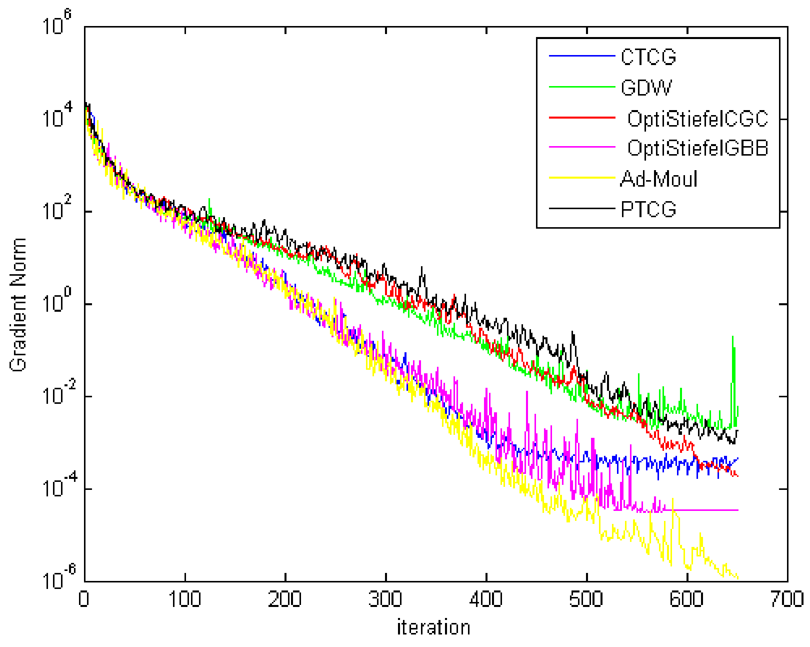

Figure 5.29: JDP 1: Average gradient norm

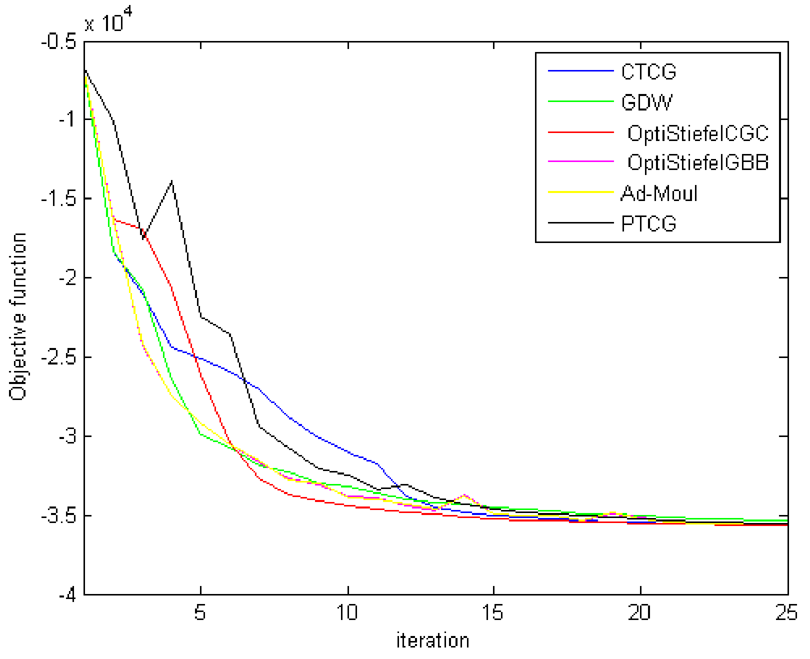

Figure 5.30: JDP 1: Average objective function

5.5.2 JDP: experiment 2

The second experiment considers $n = 500$, $p = 5$ and $N = 5$. Matrices A_is are defined as in (5.9).

Note that the only change with respect to the last experiment is that two more matrices A_j are added to the problem. Nonetheless, this change was enough to reduce the variance of the Fval column for all algorithms, as seen in Table 5.16. In this experiment variance of Fval is in the order of 1e-10 or less, which is much lesser than 1e0 corresponding to the last experiments.

On the other hand, mean value for gradient norm continues to be in the order of 1e-1. The smallest value corresponds to Ad-Moul with 8.00e-2.

Regarding number of iterations, the smallest mean value corresponds to our CTCG method with 218.1. However, OptiStiefelGBB and Ad-Moul are close with 223.6 and 227.2 iterations, respectively.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-4.6044e+04	1.57e-15	1.95e-01	291.6	459.0	8.06
	min	-4.6044e+04	4.25e-16	5.42e-02	231.0	363.0	6.31
	max	-4.6044e+04	3.02e-15	4.73e-01	393.0	641.0	11.29
	var	7.0750e-11	3.95e-31	9.20e-03	1868.7	5365.0	1.73
CTCG	mean	-4.6044e+04	1.76e-15	1.75e-01	218.1	335.6	10.46
	min	-4.6044e+04	6.78e-16	4.11e-02	153.0	230.0	7.16
	max	-4.6044e+04	4.04e-15	5.10e-01	333.0	525.0	16.27
	var	3.7886e-11	5.92e-31	1.23e-02	1723.6	4628.2	4.38
GDW	mean	-4.6044e+04	8.16e-16	1.08e-01	302.0	473.9	9.01
	min	-4.6044e+04	2.24e-16	1.33e-02	192.0	289.0	5.48
	max	-4.6044e+04	1.81e-15	5.30e-01	545.0	860.0	16.31
	var	1.8825e-10	1.87e-31	9.18e-03	7247.1	19017.0	6.81
OptiStiefelGBB	mean	-4.6044e+04	8.24e-16	1.31e-01	223.6	241.5	4.26
	min	-4.6044e+04	1.20e-16	2.14e-02	179.0	191.0	3.32
	max	-4.6044e+04	1.41e-15	4.13e-01	310.0	337.0	5.95
	var	2.6045e-10	9.95e-32	1.38e-02	1420.4	1632.4	0.51
Ad-Moul	mean	-4.6044e+04	1.17e-15	8.00e-02	227.2	243.0	4.60
	min	-4.6044e+04	4.02e-16	8.59e-03	168.0	180.0	3.41
	max	-4.6044e+04	1.84e-15	4.35e-01	342.0	366.0	6.89
	var	1.0527e-10	1.14e-31	6.18e-03	2022.7	2291.6	0.83
OptiStiefelCGC	mean	-4.6044e+04	2.92e-15	1.60e-01	265.2	409.4	7.23
	min	-4.6044e+04	1.12e-15	2.21e-02	214.0	326.0	5.73
	max	-4.6044e+04	6.32e-15	6.78e-01	433.0	678.0	11.90
	var	4.0553e-11	1.36e-30	1.74e-02	2120.8	5619.9	1.71

Table 5.16: Results of experiment 2 for Joint Diagonalization Problem

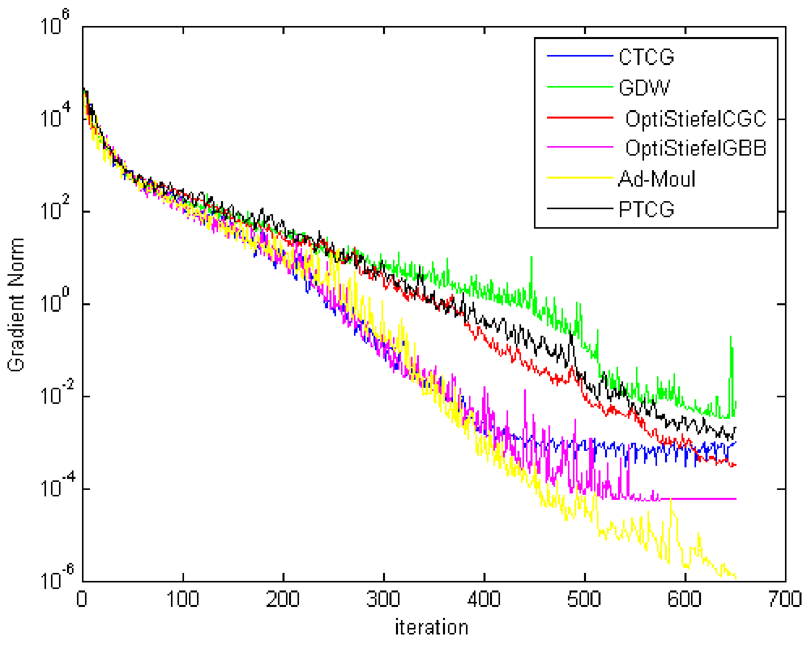

Figure 5.31: JDP 2: Average gradient norm

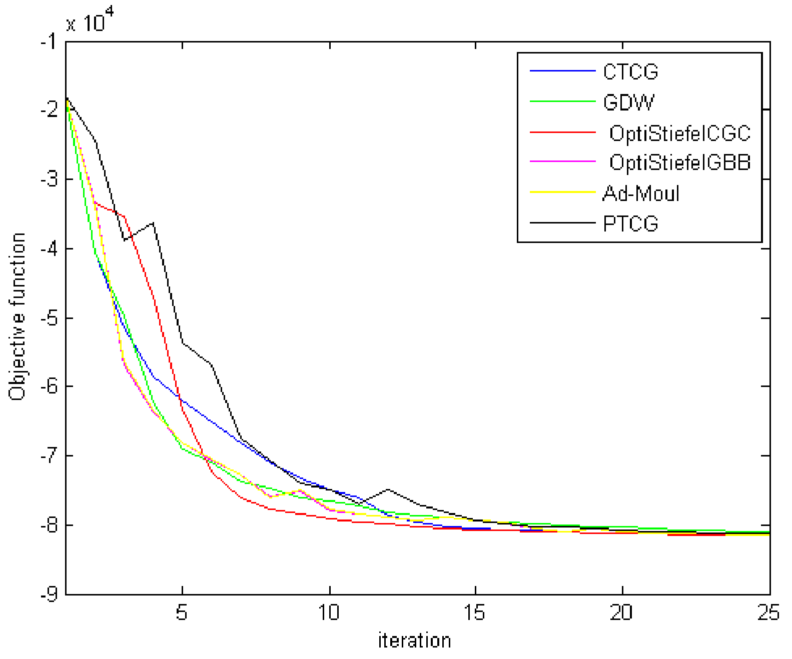

Figure 5.32: JDP 2: Average objective function

Mean number of function evaluations is lesser for CTCG than for OptiStiefelCGC, the corresponding values are 335.6 and 409.4. However, the corresponding values for OptiStiefelGBB and Ad-Moul are 241.5 and 243.0, respectively. These values show that CTCG and OptiStiefelCGC have worst performance than OptiStiefelGBB and Ad-Moul in this regard.

As always, the main issue for our algorithms is computational time. CTCG has a mean value of 10.46 seconds which is more than double the time of Ad-Moul, 4.60 seconds. The fastest of our algorithms is PTCG and it has a mean time of 8.06 seconds, which is still far from Ad-Moul or OptiStiefelGBB.

Added to the small number of iterations and function evaluations, CTCG also has a fine descending rate as shown in Figure 5.31. However, OptiStiefelGBB and Ad-Moul have better descending rates than all other algorithms.

Results summary: JDP Experiment 2

Ad-Moul and OptiStiefelGBB outperformed all other methods. These algorithms have the small mean values for number of iterations, function evaluations and computational time. If computational time is not considered then our CTCG method has a fine performance since it has a small number of iterations.

Considering computational time, PTCG is our fastest algorithm and its computational time is comparable with that of OptiStiefelCGC.

5.5.3 JDP: experiment 3

This experiment also considers the matrices A_is to be defined as in (5.9). We now increase the size of the problem considering $n = 1000$, $p = 3$ and $N = 3$.

Results for this experiment are displayed in Table 5.17. Fval column have small variance for all algorithms, in the order of 1e-8 or less.

There is one important detail in the NrmG column. For PTCG, OptiStiefelGBB, Ad-Moul and OptiStiefelCGC the maximum value is in the order of 1e0 which is large value for this feature. Our CTCG and GDW have a maximum value in the order of 1e-1, which is one order less.

Number of iterations for CTCG is actually comparable with that of OptiStiefelGBB and Ad-Moul. Although, OptiStiefelGBB and Ad-Moul have lesser number of function evaluations.

Nonetheless, the mean time for CTCG is 41.12 seconds which is almost four times the 12.32 seconds corresponding to OPtiStiefelGBB. This seems to be the main weakness of our method.

Our fastest algorithm is PTCG which has a mean time of 27.47, this value is still far from 13.89, the mean time corresponding to Ad-Moul.

Behavior of gradient norm for our CTCG method shows faster descending rate than that of OptiStiefelCGC but both algorithms end at a similar value, this is seen in figure 5.33. OptiStiefelGBB and Ad-Moul are the algorithms with better descending properties.

Results summary: JDP Experiment 3

Ad-Moul and OptiStiefelGBB outperformed all other methods. However, the performance of CTCG is fine without regarding computational time. Main features of CTCG in this experiment are small number of iterations and maximum gradient norm in the order of 1e-1. Our PTCG method is the fastest of our algorithms but is still far from being competitive.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-7.2687e+04	1.53e-15	3.41e-01	408.3	655.2	27.47
	min	-7.2687e+04	4.05e-16	5.30e-02	263.0	413.0	17.15
	max	-7.2687e+04	3.61e-15	1.14e+00	639.0	1043.0	44.21
	var	1.6491e-09	5.37e-31	4.32e-02	6653.2	18744.8	33.53
CTCG	mean	-7.2687e+04	1.66e-15	2.87e-01	274.8	434.2	41.12
	min	-7.2687e+04	3.92e-16	1.16e-01	190.0	296.0	28.09
	max	-7.2687e+04	2.81e-15	7.04e-01	384.0	621.0	58.59
	var	3.0476e-10	3.97e-31	2.76e-02	2411.1	6648.2	58.06
GDW	mean	-7.2687e+04	1.26e-15	2.57e-01	397.8	635.7	30.81
	min	-7.2687e+04	5.56e-16	2.17e-02	212.0	325.0	15.83
	max	-7.2687e+04	2.23e-15	9.56e-01	633.0	1009.0	48.95
	var	3.2030e-09	2.16e-31	4.93e-02	9087.1	24183.9	56.62
OptiStiefelGBB	mean	-7.2687e+04	1.35e-15	1.91e-01	270.8	295.1	12.32
	min	-7.2687e+04	3.34e-16	2.53e-02	214.0	235.0	9.67
	max	-7.2687e+04	2.64e-15	1.30e+00	358.0	387.0	16.24
	var	1.3347e-09	3.11e-31	4.90e-02	1717.4	1989.4	3.46
Ad-Moul	mean	-7.2687e+04	1.59e-15	2.81e-01	266.9	288.7	13.89
	min	-7.2687e+04	6.25e-16	6.24e-02	183.0	204.0	9.62
	max	-7.2687e+04	3.10e-15	1.10e+00	385.0	420.0	20.33
	var	7.4544e-09	4.87e-31	5.57e-02	2088.8	2445.9	5.82
OptiStiefelCGC	mean	-7.2687e+04	2.96e-15	4.50e-01	361.3	578.3	24.18
	min	-7.2687e+04	1.46e-15	8.03e-02	236.0	371.0	15.85
	max	-7.2687e+04	4.86e-15	1.44e+00	488.0	786.0	32.84
	var	3.0539e-08	1.14e-30	1.01e-01	3470.4	9863.7	17.32

Table 5.17: Results of experiment 3 for Joint Diagonalization Problem

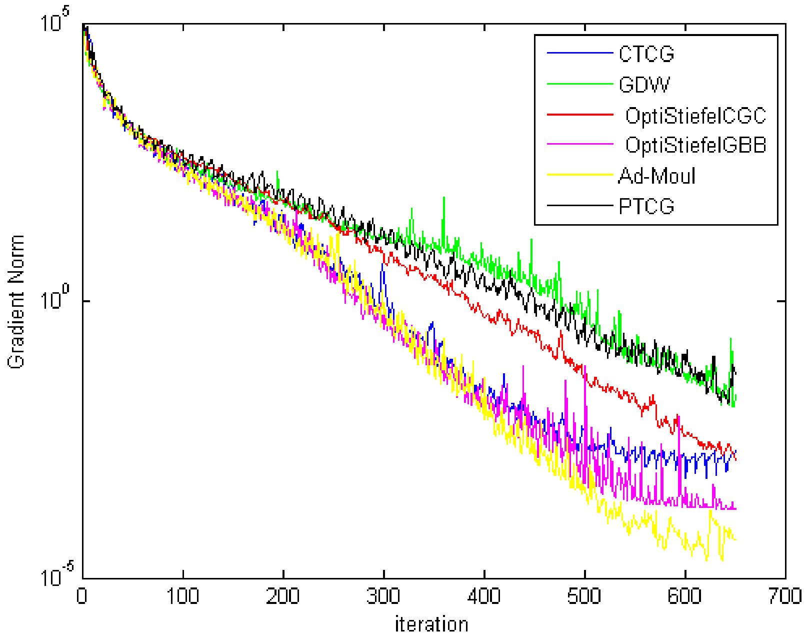

Figure 5.33: JDP 3: Average gradient norm

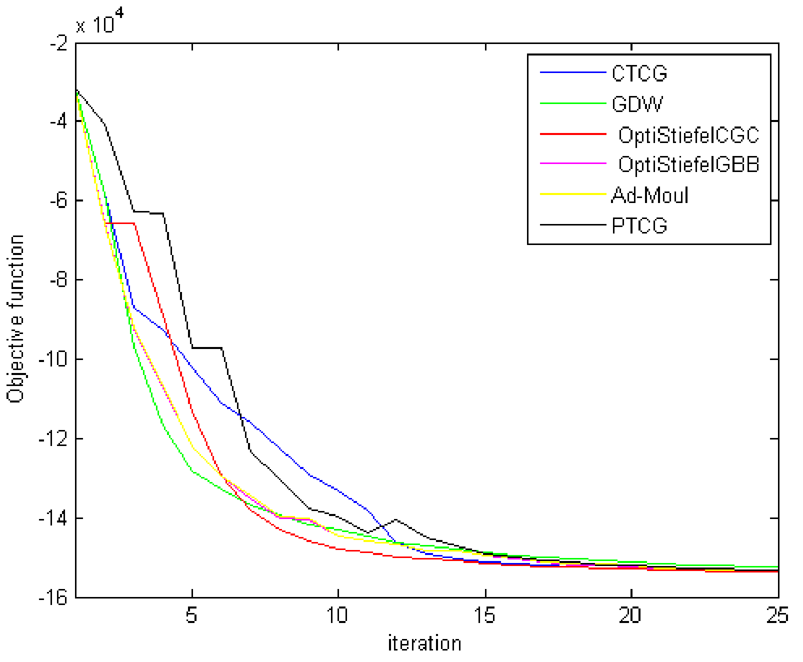

Figure 5.34: JDP 3: Average objective function

5.5.4 JDP: experiment 4

For our last experiment we consider the same matrices A_is, that is, defined as in (5.9). Main change is the augmentation of N from 3 to 5, i.e., the size of the problem is $n = 1000$, $p = 3$ and $N = 5$.

For this experiment, none of the algorithms had a maximum gradient norm below 1e0, which is a large value for this feature. Although, the mean value for the gradient norm is in the order of 1e-1 for all algorithms.

Note that mean number of iterations is smaller for our CTCG method than for OptiStiefelCGC. Actually, this value for CTCG is not far from the corresponding value for Ad-Moul and OptiStiefelGBB. OptiStiefelGBB have the smallest mean value with 284.5 iterations.

Algorithm		Fval	Feasi	NrmG	Nitr	Nfe	Time
PTCG	mean	-9.2089e+04	1.60e-15	3.99e-01	438.8	711.2	51.89
	min	-9.2089e+04	3.72e-16	1.01e-01	258.0	402.0	29.22
	max	-9.2089e+04	3.82e-15	1.60e+00	607.0	980.0	71.32
	var	1.2475e-08	6.29e-31	1.23e-01	6801.5	19219.5	101.26
CTCG	mean	-9.2089e+04	1.82e-15	3.72e-01	301.6	476.2	59.10
	min	-9.2089e+04	4.16e-16	1.40e-01	208.0	321.0	40.11
	max	-9.2089e+04	3.55e-15	1.06e+00	418.0	669.0	82.80
	var	2.2443e-09	6.25e-31	6.74e-02	2488.9	6840.1	103.66
GDW	mean	-9.2089e+04	1.37e-15	2.89e-01	410.8	657.2	51.91
	min	-9.2089e+04	2.23e-16	2.52e-02	273.0	433.0	34.26
	max	-9.2089e+04	2.55e-15	1.35e+00	525.0	850.0	66.94
	var	3.1953e-09	2.31e-31	6.90e-02	4561.2	12711.5	78.04
OptiStiefelGBB	mean	-9.2089e+04	1.52e-15	3.30e-01	284.5	307.3	21.83
	min	-9.2089e+04	4.59e-16	6.64e-02	224.0	238.0	16.74
	max	-9.2089e+04	2.50e-15	3.78e+00	421.0	459.0	32.71
	var	1.3276e-09	2.70e-31	4.37e-01	1713.7	2115.7	11.05
Ad-Moul	mean	-9.2089e+04	1.69e-15	2.70e-01	293.9	317.8	25.08
	min	-9.2089e+04	7.13e-16	6.33e-02	190.0	208.0	16.17
	max	-9.2089e+04	3.52e-15	1.31e+00	378.0	403.0	31.92
	var	8.0600e-09	5.35e-31	6.57e-02	2177.8	2628.7	16.11
OptiStiefelCGC	mean	-9.2089e+04	3.66e-15	5.16e-01	385.8	617.9	43.89
	min	-9.2089e+04	1.47e-15	1.19e-01	275.0	429.0	30.55
	max	-9.2089e+04	8.63e-15	1.30e+00	518.0	863.0	61.32
	var	5.3951e-09	2.76e-30	9.64e-02	3602.0	10069.7	50.53

Table 5.18: Results of experiment 4 for Joint Diagonalization Problem

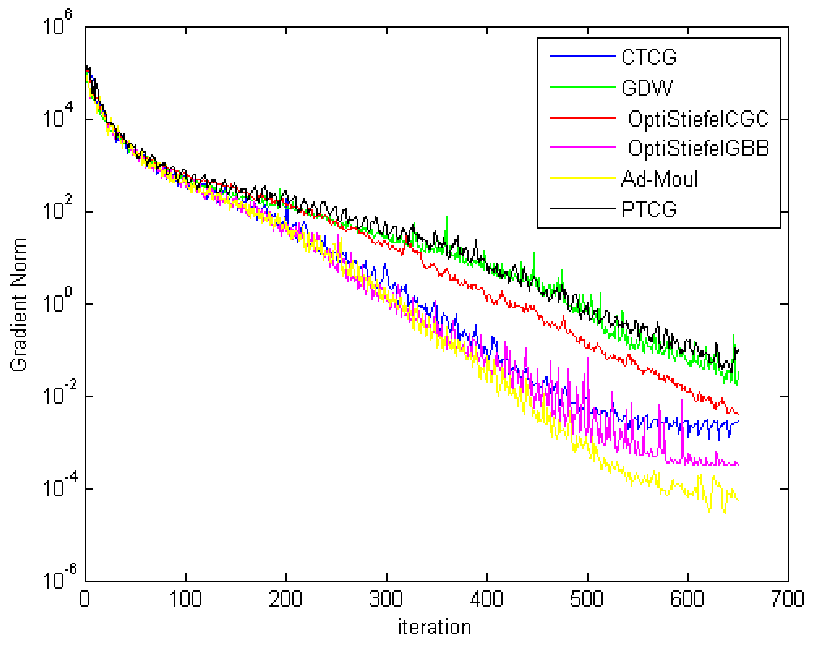

Figure 5.35: JDP 4: Average gradient norm

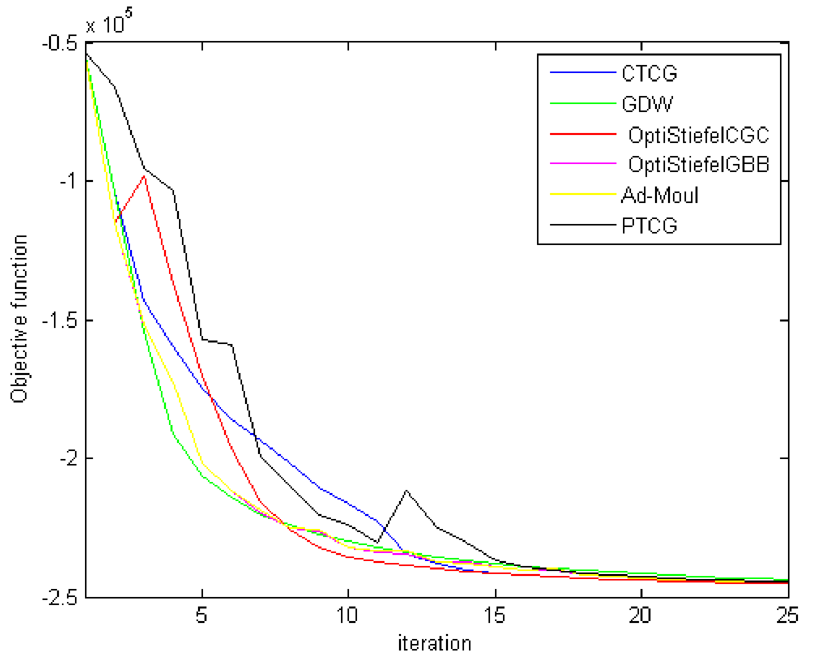

Figure 5.36: JDP 4: Average objective function

Note that our PTCG method has the largest mean value of function evaluations with 711.2 which slowed down its performance. Actually, its mean computational time is almost equal to that of GDW, which is a more expensive algorithm. In this experiment none of our algorithms have a value of CPU time comparable with state-of-the-art algorithms.

From Figure 5.35 it is seen that CTCG have a comparable descending rate with those of Ad-Moul and OptiStiefelGBB.

Results summary: JDP Experiment 4

All of our algorithms had a large computational time in this experiment, therefore, none of them performed satisfactorily. Nevertheless, CTCG has a fine performance when Time is not considered. Actually, its number of iterations and descending rate is close to those of OptiStiefelGBB and Ad-Moul, the best algorithms for this experiment.

5.6 Conclusions for numerical experiments

Experiments 2 and 4 of the Linear Eigenvalue Problem showed that a diagonal well-conditioned matrix can affect the performance of algorithms. From our methods, only CTCG performed satisfactorily in this experiments. However, its high computational cost makes it not competitive. Moreover, experiments for the Joint Diagonalization problem proved our CTCG method to have a fine performance on this challenging problem if computational time is not considered.

Our PTCG algorithm is fast and performed satisfactorily in most experiments but it is seen that the most difficult problems can affect its performance and slow it down because the number of iterations and function evaluations increases considerably.

For all experiments OptiStiefelGBB and Ad-Moul have a fine performance and are fastest in general. Our methods are still far from having this level of performance.

Chapter 6

Ideas for future work

In this work we studied problem (1.1) from a different non-Riemannian perspective. As a consequence, we have learn some interesting things about optimization over Stiefel manifold.

In this chapter we present ideas generated from our investigation that hold potential but due to lack of time had no further development.

These ideas comprehend a non-Riemannian Quasi-Newton method for optimization on $\mathrm{St}(n,p)$, deduction of a retraction using Givens rotation matrices, use of Householder transformation matrices for optimization on $\mathrm{St}(n,p)$ and a non- Riemannian optimization method for optimization over the manifold of rank-k matrices.

6.1 Non-Riemannian Quasi-Newton Method for Optimization on St(n,p)

In Chapter 4 we used a composite function $\mathcal{H}_X(W)$ and computed its gradient w.r.t. a skew-symmetric matrix $W \in \mathbb{R}^{n\times n}$. However, in order to obtain this gradient we used the *vech* form of W and differentiated only w.r.t. the $n(n-1)/2$ variables below diagonal of W. See Appendix B) for details.

Actually, $\mathcal{H}_X(W)$ can be seen as function of $vech(W) \in \mathbb{R}^{n(n-1)/2}$ instead of a function of W. Then, the gradient would be

$$\overset{vec}{\nabla}\, \mathcal{H}_X(W) = vech((I_n - W)^{-\top} G (X+Y)^\top - (X+Y) G^\top (I_n - W)^{-1}) \tag{6.1}$$

and the Quasi-Newton equation would yield the search direction

$$p_k = -H_k \overset{vec}{\nabla}\, \mathcal{H}_X(W_k), \tag{6.2}$$

and the iterative scheme

$$vech(W_{k+1}) = vech(W_k) + \alpha_k p_k. \tag{6.3}$$

As usual, the matrices H_k are symmetric and satisfy the secant equation

$$H_{k+1}y_k = s_k, \tag{6.4}$$

where $s_k = vech(W_{k+1} - W_k)$ and $y_k = \overset{vec}{\nabla} \mathcal{H}_X(W_{k+1}) - \overset{vec}{\nabla} \mathcal{H}_X(W_k)$.

The proposed Quasi-Newton method has some disadvantages that have to be commented. First, in order to evaluate $Y(W_k) = (I - W_k)^{-1}(I + W_k)X$ we need to generate W_k from $vech(W_k)$. Also, computing $\overset{vec}{\nabla} \mathcal{H}_X(W_k)$ implies computing $vech(W_k)$. So that, computational cost is increased by the use of the $vech$ operator.

On the other hand, the matrices H_k are squared so they are $n(n-1)/2$-by-$n(n-1)/2$ matrices. Hence, for large n this method is impracticable for the needed memory is larger than the available in a standard PC. As an example, in the experiments of Chapter 5 we have $n = 1000$ which would require a 499500-by-499500 matrix H_k , approximately 1858.9GB.

So we write these ideas as a possible research topic such as using of large scale techniques or modifications of H_k to make it a sparse matrix. However, this research is beyond the scope of this work.

6.2 Givens retraction for $\mathbf{St}(n, p)$

Wen and Yin used the Crank-Nicolson scheme to find a new trial point of the iterative process [48]. The result is a feasible scheme that generates a rotation matrix using the Cayley Transform $(I - W)^{-1}(I + W)$. However, W is given as a definition with no further explanation on how to choose said matrix.

We have notice that our approach is equivalent to Wen and Yin's under some conditions, i.e., using the Cayley retraction is equivalent to use the chain rule and taking the derivative of $\mathcal{F}(Y(W))$ and, thus, the selection of W corresponds to a line-search in the direction of the gradient w.r.t. W. We explain ourselves in the next paragraph.

From Appendix B we have that

$$\nabla\mathcal{H}_X(W) = (I - W)^{-\top}G(X + Y)^\top - (X + Y)G^\top(I - W)^{-1},$$

which evaluated at $W = 0$ yields

$$\nabla\mathcal{H}_X(0) = 2(GX^\top - XG^\top). \tag{6.5}$$

Note that (6.5) is the same search direction defined by Wen and Yin. Hence, instead of thinking of a retraction we can think of a composite function evaluated at $W_k = 0$ in each iteration and generating a rotation of Xk_k as next iterate.

Actually, a new method can be defined regarding the above.

Suppose that $Q := Q(M)$, is a rotation matrix that depends on the parameter M, matrix or vector, such that $Q(0) = I$. We define $Y(M) := QX$, so that $Y(0) = X$.

Notice that in the case of the Cayley retraction Q is the Cayley transform and W is the matrix parameter such that when equaling zero we have $Y(0) = X$ and $Q(0) = I$.

A possible choice for Q is to use Givens rotations. For an n-by-n skew-symmetric matrix W , let $\mathrm{Giv}(W) = \prod_{1\leq i\leq j\leq n} \mathcal{R}(i,j,W_{ij})$, where the order of multiplication is any fixed order and where $\mathcal{R}(i,j,\theta)$ is the Givens rotation of angle θ in the (i,j) plane, namely, $\mathcal{R}(i,j,\theta)$ is the identity matrix with the substitutions $e_i^\top \mathcal{R}(i,j,\theta)e_i = e_j^\top \mathcal{R}(i,j,\theta)e_j = \cos(\theta)$ and $e_i^\top \mathcal{R}(i,j,\theta)e_j = -e_j^\top \mathcal{R}(i,j,\theta)e_i = \sin(\theta)$.

A retraction based on Givens rotations has already been defined for the orthogonal group [4], $\mathrm{St}(n,n)$, as

$$R_X(XW) = X\mathrm{Giv}(W),$$

where $\eta = XW \in T_X\mathrm{St}(n,n)$. Nevertheless, we define a new Givens retraction for St(n, p) as

$$R_X(\eta) = \mathrm{Giv}(W)X, \tag{6.6}$$

where $W = W(\eta)$ is chosen as follows.

Note that

$$\sum_{1\leq i\leq j\leq n} \left.\frac{\mathrm{d}vec(\mathcal{R}(i,j,W_{ij}))}{\mathrm{d}vech(W)}\right|_{W=0} = S_n, \tag{6.7}$$

where S_n is defined as in (4.5). On the other hand, $\mathcal{R}(i,j,\theta) = I$ for $1 \leq i \leq j \leq n$. Now, defining $Y_G := Y_G(W) = \mathrm{Giv}(W)X$ yields

$$\left.\frac{\mathrm{d}vec(Y_G(W))}{\mathrm{d}vech(W)}\right|_{W=0} = (X^\top \otimes I)S_n, \tag{6.8}$$

and therefore for $\mathcal{H}_X(W) = \mathcal{F}(Y(W))$ we have

- $\mathcal{H}_X(0) = \mathcal{F}(X)$
- $\nabla\mathcal{H}_X(0) = GX^\top - XG^\top$.

These last two properties allow us to define a very similar method to that of Wen and Yin's using Y_G . The iterative process would be defined by

$$X_{k+1} = \mathrm{Giv}(-\alpha_k(G_kX_k^\top - X_kG_k^\top))X_k. \tag{6.9}$$

The problem about using Given rotation matrices is that we have to compute the product of $n(n-1)/2$ Givens matrices, although this can be programmed efficiently, for large n it becomes expensive.

Other important detail is that there is no need to invert a matrix like in the Cayley transform but the sine and cosine of $n(n-1)/2$ variables has to be calculated at each iteration. Again, the line-search procedure can elevate the cost of these calculations.

6.3 Housholder reflections for optimization on $\mathbf{St}(n,p)$

The ideas in the last two sections are theoretically fine but impractical. We now present some ideas that are practical but need further theoretical development.

Considering that$X_{k+1} = Q_k X_k$ is a feasible scheme, where Q_k is a rotation matrix, we propose to use the product of two Householder reflectors to get a rotation matrix.

Define the following curve on $\mathrm{St}(n,p)$ for a given non-null $u \in \mathbb{R}^n$, $t > 0$ and $v \in \mathbb{R}^n$ such that $u + tv \neq 0$,

$$Y_{v;u}(t) = \left(I - 2\frac{(u+tv)(u+tv)^\top}{(u+tv)^\top(u+tv)}\right)\left(I - 2\frac{uu^\top}{u^\top u}\right)X. \tag{6.10}$$

The curve (6.10) has the property $Y_{v;u}(0) = X$ for any u and v. However, choosing u, v is a difficult problem. We proceed using the following manner to obtain a feasible iterative scheme.

Suppose $u \neq 0$ is given and make

$$Y_u(v) = \left(I - 2\frac{(u+v)(u+v)^\top}{(u+v)^\top(u+v)}\right)\left(I - 2\frac{uu^\top}{u^\top u}\right)X. \tag{6.11}$$

Compute $G_v := \nabla\mathcal{F}(Y_u(v))|_{v=0}$, then, make $v = 0 - tG_v$ in (6.11) so that a line-search procedure can be done over t and next feasible iterate is obtained. Obviously, there are a lot of other choices for v that yield feasible optimization schemes. Nevertheless, there are also infinite choices for u and our numerical experiments have shown that a good selection of u is critical to obtain an effective optimization procedure.

We have not found a solution to the problem of choosing u. A first approach could be to differentiate (6.10) w.r.t. t and consider $\dot{Y}_{v;u}(0) \in T_X\mathrm{St}(n,p)$, the velocity of the curve at X, to choose u as the solution of

$$\min_u \langle \nabla\mathcal{F}(X), \dot{Y}_{v;u}(0)\rangle \quad \text{s.t.} \quad u^\top u = 1,$$

which can be a difficult problem.

We think that further research in the use of Householder reflectors can be worth the effort since the corresponding iterative scheme is feasible and does not require the inversion of any matrix. Moreover, applying a Householder reflection to an n-by-p matrix when $p \ll n$ can be done in a efficient not-expensive manner.

6.4 Simplified Cayley Transform

The advantage of Cayley Transform based Retraction is that any tangent vector can be decomposed as $\eta = UV^\top$ where $U, V \in \mathbb{R}^{n\times 2p}$ and the use of SMW formula (2.1) only requires the inversion of a $2p$-by-$2p$ matrix. Nevertheless, this is only practical when $p < n/2$ and the cost is dependent on p.

We have tried to generate a scheme where the SMW formula (2.1) is always practical since only the inversion of a 2-by-2 matrix is needed, independent of the value of p. In order to accomplish this, we consider $r, l \in \mathbb{R}^n$ and define

$$Y := Y(r, l) = (I - (rl^\top - lr^\top))^{-1}(I + rl^\top - lr^\top)X. \tag{6.12}$$

In this manner, we are forcing $W = rl^\top - lr^\top$ to have the decomposition

$$W = UV^\top = \begin{bmatrix} l & -r \end{bmatrix} \begin{bmatrix} r & l \end{bmatrix}^\top,$$

and $V^\top U$ is a 2-by-2 matrix, so that inverting $(I - (rl^\top - lr^\top))$ can be done efficiently using the SMW formula.

Naturally, there are a few details to be considered. First, if we want to use Y in a similar way to the Cayley retraction we should begin the iteration at $W_k = 0$ and then perform a line-search to find $W_{k+1} = 0 - \alpha_k(r_k l_k^\top - r_k l_k^\top)$ but suitable r_k, l_k have to be found.

Our first try consisted in setting a fixed r_k and taking $l_k = r_k$ so that $Y = X$. Then, the gradient w.r.t. l, G_l , is computed and evaluated at $l_k = r_k$. Taking $l_{k+1} = l_k - \alpha_k G_l$, for some $\alpha_k > 0$ and $Y(u_k, l_{k+1}) = X_{k+1}$ we get a feasible iterate X_{k+1} with a lower value of the objective function. However, this approach generates rotation matrices near to the identity, $(I - (rl^\top - lr^\top))^{-1}(I + rl^\top - lr^\top) \approx I$, and, therefore, the descending process is slow.

Our best proposal is to select $u \in \mathbb{R}^n$ such that $u \notin null(GX^\top - XG^\top)$ and $l = (GX^\top - XG^\top)r$. The following arguments justify our proposal

. We have shown that $\nabla\mathcal{H}_X(0) = G_k X_k^\top - X_k G_k^\top$ can be regarded as the gradient of a composite function w.r.t. W, evaluated at $W_k = 0$. Then, $W_{k+1} = W_k - \alpha_k(G_k k X_k^\top - X_k G_k^\top)$ generates a new iterate X_{k+1} via the Cayley Transform. Note that

$$\begin{aligned} \langle \nabla\mathcal{H}_X(0), rr^\top\nabla\mathcal{H}_X(0)^\top - \nabla\mathcal{H}_X(0)rr^\top \rangle &= \mathrm{Tr}[\nabla\mathcal{H}_X(0)^\top rr^\top \nabla\mathcal{H}_X(0)^\top] \\ &\quad - \mathrm{Tr}[\nabla\mathcal{H}_X(0)^\top \nabla\mathcal{H}_X(0) rr^\top] \\ &= -\mathrm{Tr}[r\nabla\mathcal{H}_X(0)^\top \nabla\mathcal{H}_X(0) r] \\ &\quad - \mathrm{Tr}[r\nabla\mathcal{H}_X(0)^\top \nabla\mathcal{H}_X(0) r] \\ &= -2\|\nabla\mathcal{H}_X(0) r^2\|_2^2 \\ &< 0, \end{aligned}$$

since $\nabla\mathcal{H}_X(0)$ is a skew-symmetric matrix. Hence, $rr^\top\nabla\mathcal{H}_X(0) - \nabla\mathcal{H}_X(0)rr^\top$ is a descent direction for any r such that $\nabla\mathcal{H}_X(0)r \neq 0$.

Once again, the problem is to select r in a clever way. One option to begin the study on r is to consider the curve

$$Y(t) = (I - t(rr^\top\nabla\mathcal{H}_X(0)^\top - \nabla\mathcal{H}_X(0)rr^\top))^{-1}(I + t(rr^\top\nabla\mathcal{H}_X(0)^\top - \nabla\mathcal{H}_X(0)rr^\top))X \tag{6.13}$$

and its velocity $\dot{Y}(0) \in T_X\mathrm{St}(n, p)$ in order to find a fine choice for r.

6.5 Optimization over the manifold of fixed-rank matrices

Building the composite function $\mathcal{H}_X(W)$ an take the derivative w.r.t. W is an idea that can be used to perform optimization over the manifold of rank-k matrices.

Consider the problem

$$\min \mathcal{F}(Z) \quad \text{s.t.} \quad Z \in \mathcal{M}_k \tag{6.14}$$

where

$$\begin{aligned}\mathcal{M}_k &= \{M \in \mathbb{R}^{m\times n} | \text{rank}(M) = k\} &(6.15)\\ &= \{U\Sigma V^\top : U \in \text{St}(m,k), V \in \text{St}(n,k), \Sigma = \text{diag}(\sigma_i), \sigma_1 \geq \cdots \geq \sigma_k > 0, \} &(6.16)\end{aligned}$$

be the manifold of fixed-rank matrices. Since $\mathcal{M}_k$ is a Riemannian manifold, Riemannian methods can be used to solve optimization problems over $\mathcal{M}_k$, such as low-rank matrix completion [50].

Similar to the case of $\text{St}(n,p)$, feasibility is maintained using a retraction for tangent vectors. Actually, a projection operator can be used as a retraction, since the truncated SVD of a tangent vector gives the desired projection. This means that at each iteration at least one SVD has to be computed. We propose a method that does not need computation of any SVD besides the one to generate the first iterate.

Let

$$Z(W_U, W_V) = (I - W_U)^{-1}(I + W_U)U\Sigma[(I - W_V)^{-1}(I + W_V)V]^\top \in \mathcal{M}_k, \tag{6.17}$$

so that

$$Z(0,0) = U\Sigma V^\top \in \mathcal{M}_k.$$

The proposed method considers alternate descending as follows.

1. $W_U^{k+1} \approx \arg\min_{W_U} \mathcal{H}_{U_k}(W_U)|_{W_U=0}$
2. $U_{k+1} = (I - W_U^{k+1})^{-1}(I + W_U^{k+1})U_k$
3. $\Sigma_{k+1} \approx \arg\min_\Sigma \mathcal{F}(U_{k+1}\Sigma V_k^\top)|_{\Sigma=\Sigma_k}$
4. $W_V^{k+1} \approx \arg\min_{W_V} \mathcal{H}_{V_k}(W_V)|_{W_V=0}$
5. $V_{k+1} = (I - W_V^{k+1})^{-1}(I + W_{V^{k+1}})V_k$

Chapter 7

Conclusions

We have shown, theoretically and experimentally, that classical conjugate gradient method can be used to solve (1.1) while maintaining a feasible iterative scheme. The Cayley Transform was successfully applied to find minimizers of functions defined over the space of skew-symmetric matrices. In order to have an equivalent problem, the minimizers of $\mathcal{F}$ should not be a rotation of $X \in \mathrm{St}(n, p)$ with some eigenvalue equal to -1. This situation was avoided using composite rotations.

We are conscious that CTCG is an expensive algorithm and its main weakness is its elevated execution time. Nevertheless, we remark that, as far as we know, this is the first effort made to solve optimization problems over Stiefel manifold using this technique. We believe in the possibility of this work leading to further investigation where more efficient schemes are developed. Particularly, use of the Polar Decomposition in order to formulate the PTCG algorithm is a first idea inspired by CTCG which achieved the reduction of computational cost.

This work did not comprehend experiments where $p > 10$ because of the following reason. It was important to prove that CTCG algorithm could be competitive regarding all aspects but complexity, that is, a reasonable number of iterations was needed in order to obtain convergence. Since this was proven by the numerical experiments in Chapter 5, we think that it is worth the effort to continue this investigation and search for more efficient schemes and, then, experiment with larger problems.

An immediate topic of research is step-size estimation since the Barzilai-Borwein step-size produced a large number of function evaluations for our algorithms. Moreover, each function evaluation needs the computation of a LU decomposition which is expensive. Reducing this cost could be the first step to produce a more efficient algorithm.

Performance of CTCG method through numerical experiments showed an expensive yet fine algorithm that is susceptible of improvement. For starters, we think that this work opens an investigation line where Riemannian concepts are combined with this kind of parametric generation of rotation matrices where classical derivatives can be used. The very first examples are presented in Chapter 6, where practical ideas are given but need further theoretical development. We treasure these and all the ideas contained in this document for they hold potential.

Appendix A

Cayley Transform

Proposition A.1 (Cayley Transform). *Let $W \in \mathbb{R}^{n\times n}$ be a skew-symmetric matrix, i.e., $W^\top = -W$. Then $I-W$ is nonsingular and the matrix $(I-W)^{-1}(I+W)$ is orthogonal. This is known as the Cayley Transform of W.*

Proof. Let $v \in \mathbb{R}^n$, $v \neq 0$. Note that $\mathbb{R} \ni v^\top W v = 0$ since

$$\begin{aligned} v^\top W v &= (v^\top W v)^\top \\ &= v^\top W^\top v \\ &= -v^\top W v. \end{aligned}$$

Moreover,

$$v^\top (I-W)v = v^\top v - v^\top W v = v^\top v = \|v\|_2^2 > 0.$$

Since v is arbitrary, $I-W$ is a positive definite matrix and, therefore, it is nonsingular.

To see that $Q = (I-W)^{-1}(I+W)$ is an orthonormal matrix is enough to prove that $Q^\top Q = QQ^\top = I$. We have

$$\begin{aligned} [(I-W)^{-1}(I+W)]^\top (I-W)^{-1}(I+W) &= (I+W)^\top (I-W)^{-\top}(I-W)^{-1}(I+W) \\ &= (I-W)(I+W)^{-1}(I-W)^{-1}(I+W) \\ &= (I+W)^{-1}(I-W)(I-W)^{-1}(I+W) \\ &= I \end{aligned}$$

□

Appendix B

Matrix calculus

In Section 2.4 we defined the matrix S_n as

$$S_n := \frac{\mathrm{d}vec(W)}{\mathrm{d}vech(W)}.$$

We now present an example of the computation of S_n and its use for computation of derivatives w.r.t. skew-symmetric matrices.

Let $w_{ji} = -w_{ij} \in \mathbb{R}$ for $i, j = 1, 2, 3, 4$. Hence, we have the skew-symmetric matrix

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} & w_{14} \\ w_{21} & 0 & w_{23} & w_{24} \\ w_{31} & w_{32} & 0 & w_{34} \\ w_{41} & w_{42} & w_{43} & 0 \end{bmatrix} = \begin{bmatrix} 0 & -w_{21} & -w_{31} & -w_{141} \\ w_{21} & 0 & -w_{32} & -w_{42} \\ w_{31} & w_{32} & 0 & -w_{43} \\ w_{41} & w_{42} & w_{43} & 0 \end{bmatrix},$$

the *vec* and *vech* forms of W are

$$vec(W) = \begin{bmatrix} 0 \\ w_{21} \\ w_{31} \\ w_{41} \\ w_{12} \\ 0 \\ w_{32} \\ w_{42} \\ w_{13} \\ w_{23} \\ 0 \\ w_{43} \\ w_{14} \\ w_{24} \\ w_{34} \\ 0 \end{bmatrix} \quad \text{and} \quad vech(W) = \begin{bmatrix} w_{21} \\ w_{31} \\ w_{41} \\ w_{32} \\ w_{42} \\ w_{43} \end{bmatrix}.$$

Then

$$S_4 = \frac{\mathrm{d}vec(W)}{\mathrm{d}vech(W)} = \begin{bmatrix}
\frac{\partial 0}{\partial w_{21}} & \frac{\partial 0}{\partial w_{31}} & \frac{\partial 0}{\partial w_{41}} & \frac{\partial 0}{\partial w_{32}} & \frac{\partial 0}{\partial w_{42}} & \frac{\partial 0}{\partial w_{43}} \\
\frac{\partial w_{21}}{\partial w_{21}} & \frac{\partial w_{21}}{\partial w_{31}} & \frac{\partial w_{21}}{\partial w_{41}} & \frac{\partial w_{21}}{\partial w_{32}} & \frac{\partial w_{21}}{\partial w_{42}} & \frac{\partial w_{21}}{\partial w_{43}} \\
\frac{\partial w_{31}}{\partial w_{21}} & \frac{\partial w_{31}}{\partial w_{31}} & \frac{\partial w_{31}}{\partial w_{41}} & \frac{\partial w_{31}}{\partial w_{32}} & \frac{\partial w_{31}}{\partial w_{42}} & \frac{\partial w_{31}}{\partial w_{43}} \\
\frac{\partial w_{41}}{\partial w_{21}} & \frac{\partial w_{41}}{\partial w_{31}} & \frac{\partial w_{41}}{\partial w_{41}} & \frac{\partial w_{41}}{\partial w_{32}} & \frac{\partial w_{41}}{\partial w_{42}} & \frac{\partial w_{41}}{\partial w_{43}} \\
\frac{\partial w_{12}}{\partial w_{21}} & \frac{\partial w_{12}}{\partial w_{31}} & \frac{\partial w_{12}}{\partial w_{41}} & \frac{\partial w_{12}}{\partial w_{32}} & \frac{\partial w_{12}}{\partial w_{42}} & \frac{\partial w_{12}}{\partial w_{43}} \\
\frac{\partial 0}{\partial w_{21}} & \frac{\partial 0}{\partial w_{31}} & \frac{\partial 0}{\partial w_{41}} & \frac{\partial 0}{\partial w_{32}} & \frac{\partial 0}{\partial w_{42}} & \frac{\partial 0}{\partial w_{43}} \\
\frac{\partial w_{32}}{\partial w_{21}} & \frac{\partial w_{32}}{\partial w_{31}} & \frac{\partial w_{32}}{\partial w_{41}} & \frac{\partial w_{32}}{\partial w_{32}} & \frac{\partial w_{32}}{\partial w_{42}} & \frac{\partial w_{32}}{\partial w_{43}} \\
\frac{\partial w_{42}}{\partial w_{21}} & \frac{\partial w_{42}}{\partial w_{31}} & \frac{\partial w_{42}}{\partial w_{41}} & \frac{\partial w_{42}}{\partial w_{32}} & \frac{\partial w_{42}}{\partial w_{42}} & \frac{\partial w_{42}}{\partial w_{43}} \\
\frac{\partial w_{13}}{\partial w_{21}} & \frac{\partial w_{13}}{\partial w_{31}} & \frac{\partial w_{13}}{\partial w_{41}} & \frac{\partial w_{13}}{\partial w_{32}} & \frac{\partial w_{13}}{\partial w_{42}} & \frac{\partial w_{13}}{\partial w_{43}} \\
\frac{\partial w_{23}}{\partial w_{21}} & \frac{\partial w_{23}}{\partial w_{31}} & \frac{\partial w_{23}}{\partial w_{41}} & \frac{\partial w_{23}}{\partial w_{32}} & \frac{\partial w_{23}}{\partial w_{42}} & \frac{\partial w_{23}}{\partial w_{43}} \\
\frac{\partial 0}{\partial w_{21}} & \frac{\partial 0}{\partial w_{31}} & \frac{\partial 0}{\partial w_{41}} & \frac{\partial 0}{\partial w_{32}} & \frac{\partial 0}{\partial w_{42}} & \frac{\partial 0}{\partial w_{43}} \\
\frac{\partial w_{43}}{\partial w_{21}} & \frac{\partial w_{43}}{\partial w_{31}} & \frac{\partial w_{43}}{\partial w_{41}} & \frac{\partial w_{43}}{\partial w_{32}} & \frac{\partial w_{43}}{\partial w_{42}} & \frac{\partial w_{43}}{\partial w_{43}} \\
\frac{\partial w_{14}}{\partial w_{21}} & \frac{\partial w_{14}}{\partial w_{31}} & \frac{\partial w_{14}}{\partial w_{41}} & \frac{\partial w_{14}}{\partial w_{32}} & \frac{\partial w_{14}}{\partial w_{42}} & \frac{\partial w_{14}}{\partial w_{43}} \\
\frac{\partial w_{24}}{\partial w_{21}} & \frac{\partial w_{24}}{\partial w_{31}} & \frac{\partial w_{24}}{\partial w_{41}} & \frac{\partial w_{24}}{\partial w_{32}} & \frac{\partial w_{24}}{\partial w_{42}} & \frac{\partial w_{24}}{\partial w_{43}} \\
\frac{\partial w_{34}}{\partial w_{21}} & \frac{\partial w_{34}}{\partial w_{31}} & \frac{\partial w_{34}}{\partial w_{41}} & \frac{\partial w_{34}}{\partial w_{32}} & \frac{\partial w_{34}}{\partial w_{42}} & \frac{\partial w_{34}}{\partial w_{43}} \\
\frac{\partial 0}{\partial w_{21}} & \frac{\partial 0}{\partial w_{31}} & \frac{\partial 0}{\partial w_{41}} & \frac{\partial 0}{\partial w_{32}} & \frac{\partial 0}{\partial w_{42}} & \frac{\partial 0}{\partial w_{43}}
\end{bmatrix}$$

$$= \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Note that for $M \in \mathbb{R}^{4\times 4}$ we have

$$
\begin{aligned}
S_4^\top vec(M) &= \begin{bmatrix} 0&1&0&0&-1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&-1&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&-1&0&0&0\\ 0&0&0&0&0&0&1&0&0&-1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&-1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&-1&0 \end{bmatrix} \begin{bmatrix} m_{11}\\ m_{21}\\ m_{31}\\ m_{41}\\ m_{12}\\ m_{22}\\ m_{32}\\ m_{42}\\ m_{13}\\ m_{23}\\ m_{33}\\ m_{43}\\ m_{14}\\ m_{24}\\ m_{34}\\ m_{44} \end{bmatrix}\\
&= \begin{bmatrix} m_{21}-m_{12}\\ m_{31}-m_{13}\\ m_{41}-m_{14}\\ m_{32}-m_{23}\\ m_{42}-m_{24}\\ m_{43}-m_{34} \end{bmatrix}\\
&= vech(M - M^\top).
\end{aligned}
$$

From above example and the definition of S_n, we can easily deduce that

$$S_n^\top vec(M) = vech(M - M^\top), \quad M \in \mathbb{R}^{n\times n}.$$

B.1 Derivatives of the objective functions used in the experiments

Details about the computation of gradient for the objective functions used in the numerical experiments are given in this section.

We start by taking the derivative of

$$\mathcal{F}(X) = \mathrm{Tr}[X^\top M], \quad X \in \mathrm{St}(n,p), M \in \mathbb{R}^{n\times p},$$

since it will be useful to compute other derivatives. Note that

$$\mathcal{F}(X) = \mathrm{Tr}[X^\top M] = \sum_i [X^\top M]_{ii} = \sum_{i=1}^{p}\sum_{j=1}^{n} X^\top_{ij} M_{ji} = \sum_{i=1}^{p}\sum_{j=1}^{n} X_{ji} M_{ji}, \tag{B.1}$$

and

$$\frac{\partial \mathcal{F}(X)}{\partial X_{uv}} = M_{uv}.$$

Hence, the gradient of $\mathcal{F}(X) = \mathrm{Tr}[X^\top M]$ is

$$G := \nabla \mathcal{F}(X) = M. \tag{B.2}$$

Linear Eigenvalue Problem. The objective function of this problem is

$$\mathcal{F}(X) =_{\mathrm{Tr}} [X^\top A X], \quad X \in \mathrm{St}(n,p), A \in \mathbb{R}^{n\times n}, A^\top = A. \tag{B.3}$$

The gradient of (B.3) can be easily calculated using property 3 in Theorem 2.1 and applying (B.2) twice with $M = AX$. Therefore, we have

$$G := \nabla \mathcal{F}(X) = -2AX. \tag{B.4}$$

Orthogonal Procrutes Problem. The objective function of this problem is

$$\mathcal{F}(X) = \mathrm{Tr}[X^\top A^\top A X - 2B^\top A X], \quad X \in \mathrm{St}(n,p), A \in \mathbb{R}^{l\times n}, B \in \mathbb{R}^{l\times p}. \tag{B.5}$$

In this case, we use properties in Theorem 2.1 to write

$$\mathcal{F}(X) = \mathrm{Tr}[X^\top A^\top A X - 2A^\top B] \tag{B.6}$$

and use (B.2) to obtain

$$G := \mathcal{F}(X) = 2A^\top A X - 2A^\top B. \tag{B.7}$$

Heterogeneous Quadratics Minimization. The objective function of this problem is

$$\mathcal{F}(X) = \sum_{i=1}^{p} X_{[i]}^\top A_i X_{[i]}, \quad X \in \mathrm{St}(n,p), A_i \in \mathbb{R}^{n\times n}, \tag{B.8}$$

where $X_{[i]}$ denotes the i-th column of X. Note that this derivative can also be calculated using (B.2) for each i and considering $M = A_i X_{[i]} \in \mathbb{R}^n$. Therefore, the gradient is

$$G := \nabla \mathcal{F}(X) = [2A_i X_{[i]}]. \tag{B.9}$$

Joint Diagonalization Problem. The objective function of this problem is

$$\mathcal{F}(X) = -\sum_{j=1}^{N} \|\mathrm{diag}(X^\top A_j X)\|_F^2, \quad X \in \mathrm{St}(n,p), A_j \in \mathbb{R}^{n\times n}. \tag{B.10}$$

First note that for $A \in \mathbb{R}^{n\times n}$ we have

$$\operatorname{diag}(X^\top AX) = \begin{bmatrix} X_{[1]}^\top AX_{[1]} \\ X_{[2]}^\top AX_{[2]} \\ \vdots \\ X_{[p]}^\top AX_{[p]} \end{bmatrix},$$

so that

$$\begin{aligned} \|\operatorname{diag}(X^\top AX)\|_2^2 &= \operatorname{diag}(X^\top AX)^\top \operatorname{diag}(X^\top AX) \\ &= \begin{bmatrix} X_{[1]}^\top AX_{[1]} & X_{[2]}^\top AX_{[2]} & \cdots & X_{[p]}^\top AX_{[p]} \end{bmatrix} \begin{bmatrix} X_{[1]}^\top AX_{[1]} \\ X_{[2]}^\top AX_{[2]} \\ \vdots \\ X_{[p]}^\top AX_{[p]} \end{bmatrix} \\ &= \sum_{i=1}^{p} (X_{[i]}^\top AX_{[i]})^2. \end{aligned}$$

So that (B.10) can be written as

$$\mathcal{F}(X) = -\sum_{j=1}^{N}\sum_{i=1}^{p} (X_{[i]}^\top A_j X_{[i]})^2, \tag{B.11}$$

which gradient is

$$G := \nabla\mathcal{F} = \left[-4\sum_{j=1}^{N} (X_{[i]}^\top A_j X_{[i]}) A_j X_{[i]} \right]. \tag{B.12}$$

Appendix C

Details on the iterative scheme of steepest descent method

Consider $W_{k+1} = W_k - \alpha_k \nabla \mathcal{H}(W_k)$ and the following decomposition of the gradient

$$\begin{aligned}\nabla \mathcal{H}_X(W_k) &= (I - W_k)^{-\top} G_k (X + Y_k)^\top - (X + Y_k) G_k^\top (I - W_k)^{-1} \\ &= \begin{bmatrix}(I - W_k)^{-\top} G_k & -(X + Y_k)\end{bmatrix} \begin{bmatrix}(X + Y_k) & (I - W_k)^{-\top} G_k\end{bmatrix}^\top \\ &= U_k V_k^\top,\end{aligned}$$

where $Y_k := (I - W_k)^{-1}(I + W_k)X$. Define $C_k := (I - W_k)^{-1}$. we have

$$\begin{aligned}Y_{k+1} &= (I - W_{k+1})^{-1}(I + W_{k+1})X \\ &= (I - (W_k - \alpha_k \nabla \mathcal{H}_X(W_k)))^{-1}(I + W_k - \alpha_k \nabla \mathcal{H}_X(W_k))X \\ &= (I - W_k + \alpha_k U_k V_k^\top)^{-1}(I + W_k - \alpha_k U_k V_k^\top)X \\ &= [C_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top C_k](I + W_k - \alpha_k U_k V_k^\top)X \\ &= Y_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top Y_k - \alpha_k C_k U_k V_k^\top X \\ &\quad + \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top C_k (\alpha_k U_k V_k^\top) X \\ &= Y_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top Y_k \\ &\quad - \alpha_k C_k U_k [I_{2p} + (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top C_k (\alpha_k U_k)] V_k^\top X \\ &= Y_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top Y_k \\ &\quad - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} [I_{2p} + \alpha_k V_k^\top C_k U_k - \alpha_k V_k^\top C_k U_k] V_k^\top X \\ &= Y_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top Y_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top X \\ &= Y_k - \alpha_k C_k U_k (I_{2p} + \alpha_k V_k^\top C_k U_k)^{-1} V_k^\top (X + Y_k).\end{aligned}$$

References

[1] Traian Abrudan, Jan Eriksson, and Visa Koivunen. Conjugate gradient algorithm for optimization under unitary matrix constraint. *Signal Processing*, 89(9):1704–1714, 2009.

[2] Traian E Abrudan, Jan Eriksson, and Visa Koivunen. Steepest descent algorithms for optimization under unitary matrix constraint. *IEEE Transactions on Signal Processing*, 56(3):1134–1147, 2008.

[3] P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

[4] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds.* Princeton University Press, 2009.

[5] Roy L Adler, Jean-Pierre Dedieu, Joseph Y Margulies, Marco Martens, and Mike Shub. Newton's method on riemannian manifolds and a geometric model for the human spine. *IMA Journal of Numerical Analysis*, 22(3):359–390, 2002.

[6] Christopher G Baker, P-A Absil, and Kyle A Gallivan. An implicit trust-region method on riemannian manifolds. *IMA journal of numerical analysis*, 28(4):665–689, 2008.

[7] János Balogh, Tibor Csendes, and Tamás Rapcsák. Some global optimization problems on stiefel manifolds. *Journal of Global Optimization*, 30(1):91–101, 2004.

[8] Robert G Bartle. The elements of real analysis. 1976.

[9] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

[10] Dimitri P Bertsekas. *Nonlinear programming.* Athena scientific Belmont, 1999.

[11] João R Cardoso and F Silva Leite. Exponentials of skew-symmetric matrices and logarithms of orthogonal matrices. *Journal of computational and applied mathematics*, 233(11):2867–2875, 2010.

[12] Oscar Susano Dalmau Cedeño and Harry Fernando Oviedo Leon. Projected nonmonotone search methods for optimization with orthogonality constraints. *Computational and Applied Mathematics*, pages 1–27, 2017.

[13] Yuhong Dai. Nonmonotone conjugate gradient algorithm for unconstrained optimization. *Journal of Systems Science and Complexity*, 15(2):139–145, 2002.

[14] Alexandre d'Aspremont, Laurent E Ghaoui, Michael I Jordan, and Gert R Lanckriet. A direct formulation for sparse pca using semidefinite programming. In *Advances in neural information processing systems*, pages 41–48, 2005.

[15] Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[16] Lars Eldén and Haesun Park. A procrustes problem on the stiefel manifold. *Numerische Mathematik*, 82(4):599–619, 1999.

[17] Paul L Fackler. Notes on matrix calculus. *North Carolina State University*, 2005.

[18] Reeves Fletcher and Colin M Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[19] JB Francisco and Tiara Martini. Spectral projected gradient method for the procrustes problem. *TEMA (São Carlos)*, 15(1):83–96, 2014.

[20] Daniel Gabay. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37(2):177–219, 1982.

[21] M. Giaquinta and G. Modica. *Mathematical Analysis: An Introduction to Functions of Several Variables*. Mathematical analysis. Birkhäuser Boston, 2009.

[22] Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[23] John C Gower and Garmt B Dijksterhuis. *Procrustes problems*, volume 30. Oxford University Press on Demand, 2004.

[24] Igor Grubišić and Raoul Pietersz. Efficient rank reduction of correlation matrices. *Linear algebra and its applications*, 422(2-3):629–653, 2007.

[25] Wen Huang. *Optimization algorithms on Riemannian manifolds with applications*. PhD thesis, The Florida State University, 2013.

[26] Wen Huang, P-A Absil, and Kyle A Gallivan. Intrinsic representation of tangent vectors and vector transports on matrix manifolds. *Numerische Mathematik*, 136(2):523–543, 2017.

[27] Wen Huang, Kyle A Gallivan, and P-A Absil. A broyden class of quasi-newton methods for riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

[28] Bo Jiang and Yu-Hong Dai. A framework of constraint preserving update schemes for optimization on stiefel manifold. *Mathematical Programming*, 153(2):535–575, 2015.

[29] Marcel Joho and Heinz Mathis. Joint diagonalization of correlation matrices by using gradient methods with application to blind signal separation. In *Sensor Array and Multichannel Signal Processing Workshop Proceedings, 2002*, pages 273–277. IEEE, 2002.

[30] Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(Feb):517–553, 2010.

[31] Steven E Koonin and Dawn C Meredith. *Computational physics: Fortran version*. Addison-Wesley, 1990.

[32] Erwin Kreyszig. *Introductory functional analysis with applications*, volume 1. wiley New York, 1978.

[33] John M Lee. Introduction to smooth manifolds, 2001.

[34] David G Luenberger. Introduction to linear and nonlinear programming. 1973.

[35] Jonathan H Manton. Optimization algorithms exploiting unitary constraints. *IEEE Transactions on Signal Processing*, 50(3):635–650, 2002.

[36] Yasunori Nishimori and Shotaro Akaho. Learning algorithms utilizing quasi-geodesic flows on the stiefel manifold. *Neurocomputing*, 67:106–135, 2005.

[37] Jorge Nocedal and Stephen J Wright. Numerical optimization 2nd, 2006.

[38] Raoul Pietersz 4 and Patrick JF Groenen. Rank reduction of correlation matrices by majorization. *Quantitative Finance*, 4(6):649–662, 2004.

[39] Wolfgang Ring and Benedikt Wirth. Optimization methods on riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*, 22(2):596–627, 2012.

[40] Youcef Saad. *Numerical methods for large eigenvalue problems*. Manchester University Press, 1992.

[41] Hiroyuki Sato and Toshihiro Iwai. A riemannian optimization approach to the matrix singular value decomposition. *SIAM Journal on Optimization*, 23(1):188–212, 2013.

[42] Hiroyuki Sato and Toshihiro Iwai. A new, globally convergent riemannian conjugate gradient method. *Optimization*, 64(4):1011–1031, 2015.

[43] Berkant Savas and Lek-Heng Lim. Quasi-newton methods on grassmannians and multilinear approximations of tensors. *SIAM Journal on Scientific Computing*, 32(6):3352–3393, 2010.

[44] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[45] Steven T Smith. Optimization techniques on riemannian manifolds. *Fields institute communications*, 3(3):113–135, 1994.

[46] Eduard Stiefel. Richtungsfelder und fernparallelismus in n-dimensionalen mannigfaltigkeiten. *Commentarii Mathematici Helvetici*, 8(1):305–353, 1935.

[47] Fabian J Theis, Thomas P Cason, and P-A Absil. Soft dimension reduction for ica by joint diagonalization on the stiefel manifold. In *International Conference on Independent Component Analysis and Signal Separation*, pages 354–361. Springer, 2009.

[48] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[49] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[50] Chao Yang, Juan C Meza, and Lin-Wang Wang. A trust region direct constrained minimization algorithm for the kohn–sham equation. *SIAM Journal on Scientific Computing*, 29(5):1854–1875, 2007.

[51] Hongchao Zhang and William W Hager. A nonmonotone line search technique and its application to unconstrained optimization. *SIAM journal on Optimization*, 14(4):1043–1056, 2004.

[52] Xiaojing Zhu. A riemannian conjugate gradient method for optimization on the stiefel manifold. *Computational Optimization and Applications*, 67(1):73–110, 2017.

[53] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.