Optimization Algorithms on Tensor Manifolds using Charts

Author:

Mohammad Mohsen Abedin Nejad

Advisors:

Dr. Marie Billaud-Friess and Prof. Anthony Nouy

A thesis submitted in partial fulfillment for the degree of Erasmus Mundus Master of Science in Computational Mechanics

Location: **Nantes, France** Date: **22 August 2017**

Ecole Centrale de Nantes (ECN)

Institut de Recherche en Génie Civil et Mécanique (GeM)

The International Center for Numerical Methods in Engineering (CIMNE)

Acknowledgement

I would like to express my sincere gratitude to my advisors Dr. Marie Billaud Friess and Prof. Anthony Nouy and for their support, patience and immense knowledge and more importantly for sharing their great idea and work with me by which I could build my thesis.

I would also like to thank my parents for their continuous supporting me spiritually throughout working on this thesis and my life in general.

Contents

List of Figures

Chapter 1

Introduction

The purpose of this report is to present some algorithms to optimize a smooth objective functional defined on the space of fixed low-rank matrices denoted by $\mathcal{M}_r(\mathbb{R}^{n\times m})$ where $r < \min(n, m)$ is the rank of $n \times m$-matrices. In [1], it is shown that these sets of fixed low-rank matrices are Riemannian manifolds embedded in $\mathbb{R}^{n\times m}$. Therefore, the differentiable structure and inner product of $\mathbb{R}^{n\times m}$ induced on $\mathcal{M}_r(\mathbb{R}^{n\times m})$ let us to define gradient and Hessian of the objective functional and finally obtain sensible optimization algorithms (see [1, 2]). But in this report, we use manifold structure of $\mathcal{M}_r(\mathbb{R}^{n\times m})$ described by local charts. Local charts allow us to map manifolds to parameter spaces and provide optimization algorithms on parameter spaces.

The optimization of the smooth objective functions where its search domains belongs to $\mathcal{M}_r(\mathbb{R}^{n\times m})$ are considered as non-linear optimisation problems. These kind of problems are current research topics as one can see in [3, 4] for matrix completion, in [5] for phaseless reconstruction, in [6] for independent component analysis and in many other research works. In this report, we apply the optimization algorithms to three problems: matrix completion, model reduction for parameter dependent partial differential equations and dimension reduction for 2-dimensional PDEs.

In the first chapter, we present a brief introduction to the theory of manifolds and describe the geometry of the sets of fixed rank matrices $\mathcal{M}_r(\mathbb{R}^{n\times m})$. In the second chapter, we present Riemannian optimization algorithms on $\mathcal{M}_r(\mathbb{R}^{n\times m})$ and in particular, present a steepest descent algorithm using Riemannian structure. Then, we provide optimization algorithms based on the manifold structure of $\mathcal{M}_r(\mathbb{R}^{n\times m})$ using local charts that are the steepest descent and the Newton algorithms. In the last chapter, we see the numerical results obtained by implementing the Newton algorithm to solve partial differential equations and matrix completion problem.

Chapter 2

Low-rank matrix manifolds

In this chapter, we first briefly discuss about theoretical background of the geometrical objects called manifolds. Then, we see that the sets of fixed low-rank matrices possess the manifold structure.

2.1 Elements of geometry

In this section, we recall definition of smooth manifolds, differentiable functions on manifolds, tangent spaces and submanifolds. One can see more and deeper discussions about manifolds in [7] and in particular about matrix manifolds in [1].

2.1.1 Smooth manifolds

Manifolds are the sets of points which locally look like Euclidean spaces. These identifications are called charts. Therefore, charts help us to consider manifolds locally as linear spaces while globally these sets may be non-linear spaces. When we say 'locally', we mean the special subsets. These special subsets are open sets belonging to the topology defined on the manifold. Thus, the manifolds are in fact the topological spaces. Moreover, most practical problems need the sensible concept of differentiation and integration of some function defined on the manifold. To do this, we should be able to define smoothness when we pass from one open set to another one. This lead to the concept of compatible charts letting us to do calculus on manifolds. In fact, smooth manifolds are manifolds equipped with compatible charts. The following definitions present the formal description of manifolds.

Definition 2.1.1 (n-dimensional charts). *Let M be a set of points. The pair (U, φ) is a n-dimensional chart of M if U is an open subset of M and $\varphi : U \to \varphi(U) \subseteq \mathbb{R}^n$ is a bijection where $\varphi(U)$ is an open subset of $\mathbb{R}^n$ (see Figure 2.1). Given $x \in U$, the elements of $\varphi(x) = (x_1, \dots, x_n)$ are called local coordinates of x related to the chart (U, φ). Moreover, we denote neighbourhood of x as a open set U_x including x such that $\varphi(x) = 0$.*

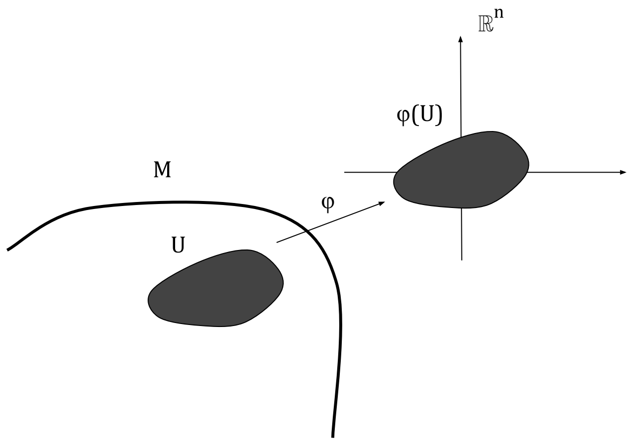

Figure 2.1: Coordinate chart

Definition 2.1.2 (Compatible charts). *Let (U_1, φ_1) and (U_2, φ_2) be two n-dimensional charts on the set of M such that $U_1 \cap U_2 \neq \emptyset$. The pairs (U_1, φ_1) and (U_2, φ_2) are compatible charts (see Figure 2.2) if* *i . $\varphi_1(U_1 \cap U_2)$ and $\varphi_2(U_1 \cap U_2)$ are open subset of $\mathbb{R}^n$.*

ii . $\varphi_1 \circ \varphi_2^{-1} : \varphi_2(U_1 \cap U_2) \to \mathbb{R}^n$ is a smooth invertible function with smooth inverse. This mapping is called change of coordinates.

Note that if $U_1 \cap U_2 = \emptyset$, the charts (U_1, φ_1) and (U_2, φ_2) are compatible.

Definition 2.1.3 (Smooth atlas)**.** *The set $\mathcal{A} = \{(\mathcal{U}_i, \varphi_i) | i \in I\}$ of compatible charts such that $\cup_{i \in I} U_i = M$ is called a smooth atlas of manifold M.*

One can generate a maximal atlas $\mathcal{A}^+$ from a given smooth atlas $\mathcal{A}$ by adding all possible compatible charts to $\mathcal{A}$.
We say that $(M, \mathcal{A}^+)$ is a smooth manifold where M is an arbitrary set and $\mathcal{A}^+$ is its maximal atlas. Henceforward, we mean smooth manifolds, compatible charts and maximal atlas when we mention manifolds, charts and atlas.

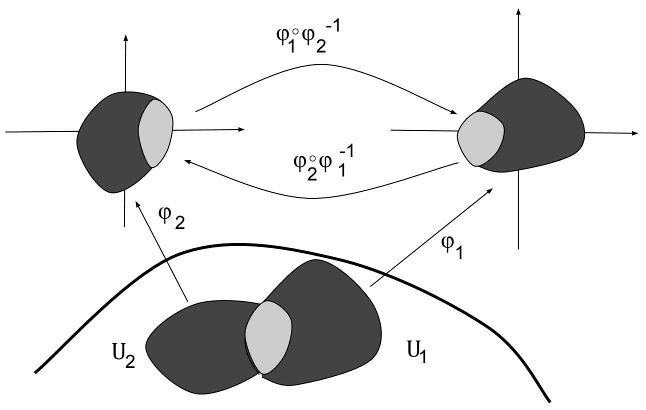

Figure 2.2: Two compatible charts

Example 2.1.4 (Linear spaces as manifolds)**.** *The simplest example of a manifold is* $\mathbb{R}$ *whose atlas can be chosen the identity map on* $\mathbb{R}$ *to itself. Similarly,* $\mathbb{R}^n$ *can be viewed as a manifold. More generally, each* n*-dimensional linear space* $\mathcal{V}$ *and* $\varphi : \mathcal{V} \to \mathbb{R}^n : v \to [v_1 \dots v_n]^T$ *such that* $v = \sum_{i=1}^{n} v_i e_i$ *form a linear manifold (*e_i *is denoted a basis vector of* $\mathcal{V}$*).*

Example 2.1.5 (Matrix manifolds $\mathbb{R}^{n\times m}$)**.** *The space of* $n \times m$ *of real matrices owns a linear manifold structure since it is isomorphic with* $\mathbb{R}^{nr}$ *using* vec *operator. For any* $Z \in \mathbb{R}^{n\times m}$*,* $\mathrm{vec}(Z) \in \mathbb{R}^{nm}$ *denotes a vector obtained by stacking the columns of* Z*. In the next chapter, we talk about the geometry of all fixed rank* $n \times m$ *matrices and show that they admit a manifold structure.*

2.1.2 Differentiable functions on manifolds

Now, we consider the functions defined on manifolds. From classical calculus, we know how to define differentiable functions between euclidean spaces. The derivative of a smooth function $f : \mathbb{R}^n \to \mathbb{R}^m$ at point $x \in \mathbb{R}^n$, is linear operator $Df(x) : \mathbb{R}^n \to \mathbb{R}^m$. This fact can be extended to smooth functions defined on manifolds using local charts.

Definition 2.1.6 (Smooth functions)**.** *Let* $f : M \to N$ *be a function between* m*-manifold* M *and* n*-manifold* N *and let* $x \in M$*. Let* (U_1, φ_1) *and* (U_2, φ_2) *be charts such that* $x \in U_1 \cap U_2$*. Then,* f *is a smooth function at* x *if* $\hat{f} = \varphi_2 \circ f \circ \varphi_1^{-1} : \varphi_1(U_1) \subseteq \mathbb{R}^m \to \mathbb{R}^n$ *is smooth at* $\varphi_1(x)$*. The function* $\hat{f}$ *is called coordinate representation of* f*.*

2.1.3 Tangent spaces

Here, we provide a common definition of tangent spaces on manifolds. Let $x \in M$ and $\gamma : (-\epsilon, \epsilon) \to M$ where M is a n-manifold and γ is a smooth function defining a curve on M such that $\gamma(0) = x$. Let (U, φ) be an arbitrary chart such that $x \in U$. Then, the map $\varphi \circ \gamma : (-\epsilon, \epsilon) \to \mathbb{R}^n$ defines a smooth curve in $\mathbb{R}^n$ and $v = (\varphi \circ \gamma)'(0)$ is a tangent vector in $\mathbb{R}^n$. But there exist many curves like γ producing the same vector v. Using the following definition, we consider all curves with the same tangent vector are equivalent.

Definition 2.1.7 (Equivalent curves on manifolds)**.** *Let M be a n-manifold and $x \in M$. Two smooth curves $\gamma_1 : (-\epsilon, \epsilon) \to \mathbb{R}^n$ and $\gamma_2 : (-\epsilon, \epsilon) \to \mathbb{R}^n$ such that $\gamma_1(0) = \gamma_2(0) = x$, are equivalent if and only if there exists a chart (U, φ) including x so that,*

$$(\varphi \circ \gamma_1)'(0) = (\varphi \circ \gamma_2)'(0).$$

The equivalence relation defined in the Definition 2.1.7 is independent of the choice of charts. To show this, let ψ be another chart including $x \in M$. Then,

$$(\psi \circ \gamma_1)'(0) = ((\psi \circ \varphi^{-1}) \circ (\varphi \circ \gamma_1))'(0)$$

and similarly for γ_2, so we get $(\psi \circ \gamma_1)'(0) = (\psi \circ \gamma_2)'(0)$. Considering the Definition 2.1.7, we are able to define a tangent vector on manifolds.

Definition 2.1.8 (Tangent vectors on manifolds)**.** *Let M be a n-manifold and $x \in M$. A tangent vector on M at x is any equivalence class of curves on M including x modulo the relation given in the Definition 2.1.7.*

The tangent space at $x \in M$ is the set of all tangent vectors at x and is denoted by T_xM. Moreover, $\bigcup_{x \in M} T_xM$ is called tangent bundle and is denoted by TM.
To see the components of a tangent vector $\xi \in T_xM$ where M is a m-manifold, suppose that U_x is the neighbourhood of x and (U_x, φ) is the chart related to this neighbourhood. So we have the associated coordinates $(x_1, \ldots, x_m)$ in U_x. Then, the components of ξ are $(\xi_1, \ldots, \xi_m)$ such that,

$$u_i = \frac{d}{dt}(\varphi \circ \gamma)_i\Big|_{t=0}, \qquad (i = 1, \ldots, m)$$

where γ is the representative of the equivalent class defined in 2.1.7. It is also useful to describe the vector fields on manifolds.

Definition 2.1.9 (Vector fields). *Let M be a manifold. A vector field on M is the following continuous map*

$$\xi : M \to TM, \qquad x \to \xi_x$$

where $x \in M$ and $\xi_x \in T_xM$. In other words, a vector field maps each point x of the manifold to a vector in the tangent space T_xM.

Let $(x_1, \ldots, x_n)$ be the local coordinates of the open set $U \subseteq M$. The value of the vector field ξ at any point $x \in U$ can be written as,

$$\xi_x = \sum_{i=1}^{n} \xi_i(x) \frac{\partial}{\partial x_i}\Big|_x \tag{2.1}$$

where $\xi_i : U \to \mathbb{R}$ are component functions of ξ with respect to the given chart.
Now, we can define the differential of a map $f : M \to N$ between two manifolds M and N by the concept of tangent space. The differential of f at $x \in M$ is the following linear mapping,

$$Df(x) : T_xM \to T_{f(x)}N, \qquad \xi \to Df(x)[\xi] \tag{2.2}$$

Note that if M and N are linear manifolds, then

$$Df(x)[\xi] = \lim_{h \to 0} \frac{f(x + h\xi) - f(x)}{h}$$

that is the classical definition of directional derivative. Therefore the derivative defined in (2.2) is a generalised directional derivative. According to the equation (2.1), if $f : M \to \mathbb{R}$ and (U, φ) is the local coordinate charts around x then,

$$Df(x)[\xi] = \sum_{i=1}^{n} \xi_i(\varphi(x)) \frac{\partial}{\partial x_i}(f \circ \varphi^{-1})\Big|_{\varphi(x)}$$

2.1.4 Embedded submanifolds

In general, submanifolds are the subsets with a structure of manifolds. More formally, we have the following definition.

Definition 2.1.10 (Embedded submanifolds)**.** *Let M be a m-manifold and N be a n-manifold such that $N \subseteq M$. For $n < m$, one can write $\mathbb{R}^m = \mathbb{R}^n \times \mathbb{R}^{(m-n)}$. Then N is a embedded n-dimensional submanifold of M, if for any $x \in N$, there is a coordinate chart (U_x, φ) such that U_x is the neighbourhood of x and*

$$U_x \cap N = \varphi^{-1}(\mathbb{R}^n \times (0, \dots, 0)).$$

$U_x \cap N$ is called a n-slice of neighbourhood U_x, the coordinate chart (U_x, φ) is called a slice chart for N in M, and the corresponding coordinates $(x_1, \dots, x_n)$ are called slice coordinates.

The identity map $i : N \hookrightarrow M$ is called an inclusion map if any element of N under i treated as an element of M. The inclusion map is nothing but a fancy way to say that any element of N can be viewed as an element of M. According to the Definition 2.1.10, it is proved that if N is an embedded n-dimensional submanifold of M, then N is a n-manifold with the topology induced by M and the unique smooth structure inherited by M (see for example [7]). In other words, the inclusion map $i : N \hookrightarrow M$ should be a smooth embedding.

2.1.5 Riemannian metrics

In this section, we introduce metric structure on smooth manifolds. These metric structures let us measure the lengths of tangent vectors or the angles between them. How to measure the length of tangent vector allow us to define the gradient of a given real-valued function.

Definition 2.1.11 (Riemannian metric)**.** *Let $x \in M$ with M a manifold. A Riemannian metric on M is the inner product (a bilinear, symmetric positive-definite form) $g_x : T_xM \times T_xM \to \mathbb{R}^+$ such that for any smooth vector fields ξ_x and ζ_x the function $x \to g_x(\xi_x, \zeta_x)$ is smooth.*

In this report, we use one of the following notations for Riemannian metric,

$$g(\xi_x, \zeta_x) = \langle \xi_x, \zeta_x \rangle$$

A manifold whose tangent space equipped with a Riemannian metric is called a Riemannian manifold. Now let $x = (x_1, \ldots, x_n)$ be the local coordinates related to the chart (U, φ) of M. Then, the Riemannian metric g is represented by a symmetric, positive-definite matrix

$$[g_{ij}(x)]_{i,j=1,\ldots,n}$$

such that

$$g_{ij} = \langle \frac{\partial}{\partial x_i}, \frac{\partial}{\partial x_j} \rangle$$

The Riemannian metric induces the following norm that compute the length of tangent vector

$$\|\xi\| = \sqrt{\langle \xi, \xi \rangle}.$$

Let $f : M \to \mathbb{R}$ be a smooth function. Then, the gradient of f at $x \in M$, denoted by $\nabla f(x)$, is defined as the unique vector in T_xM such that,

$$\langle \nabla f(x), \xi \rangle = Df(x)[\xi], \qquad \forall \xi \in T_xM. \tag{2.3}$$

2.2 Geometric description of the set of fixed rank matrices

As the search space to optimize a given cost functional in this report, is the space of fixed low-rank matrices denoted by $\mathcal{M}_r(\mathbb{R}^{n\times m})$ where $r < \min(n, m)$, we give a brief geometric description of these sets. Without loss of generality we assume that $n > m$. To introduce the local charts for $\mathcal{M}_r(\mathbb{R}^{n\times m})$, we need to know the geometry of Grassman and Stiefel manifolds. The Grassman manifold $\mathbb{G}_r(\mathbb{R}^n)$ consists all r-dimensional linear subspaces of $\mathbb{R}^n$ and the Stiefel manifold $\mathcal{M}_r(\mathbb{R}^{n\times r})$ is the set of full rank $n \times r$- matrices. For more details one can refer to [8].

2.2.1 The Grassman manifold $\mathbb{G}_r(\mathbb{R}^n)$

In this section, we present a n-atlas for the set $\mathbb{G}_r(\mathbb{R}^n)$ of all r-dimensional linear subspaces of $\mathbb{R}^n$. We define the surjective map

$$\mathrm{col}_{n,r} : \mathcal{M}_r(\mathbb{R}^{n\times r}) \mapsto \mathbb{G}_r(\mathbb{R}^n), \qquad Z \to \mathrm{col}_{n,r}(Z) \tag{2.4}$$

where $\mathrm{col}_{n,r}(Z)$ is a r-dimensional subspace of $\mathbb{R}^n$ generated by column vectors of Z. There is an infinite number of Z's whose column space represents a particular r-dimensional linear subspace of $\mathbb{R}^n$. For instance, all elements of

$$\{(a,0)^T | 0 \neq a \in \mathbb{R}\}$$

generate the same subspace in $\mathbb{R}^2$. In fact, each element of $\mathbb{G}_r(\mathbb{R}^n)$ corresponds to an equivalence class $Z\mathrm{GL}_r$ where GL_r is the set of square invertible matrices of size r. Therefore, we need to choose a representative of this equivalence class corresponding an element of $\mathbb{G}_r(\mathbb{R}^n)$.

Definition 2.2.1 (Affine cross section). *Let $Z \in \mathcal{M}_r(\mathbb{R}^{n\times r})$ and $Z_\perp \in \mathcal{M}_r(\mathbb{R}^{n\times(n-r)})$ such that $Z^T Z_\perp = 0$. Then matrix Z defines the following affine cross section,*

$$\mathcal{S}_Z = \{W \in \mathcal{M}_r(\mathbb{R}^{n\times r}) \mid Z^T(W - Z) = 0\} \tag{2.5}$$

Lemma 2.2.2. *The affine cross section $\mathcal{S}_Z$ is characterized by*

$$\mathcal{S}_Z = \{Z + Z_\perp X \mid X \in \mathbb{R}^{(n-r)\times r}\} \tag{2.6}$$

and map

$$\eta_Z : \mathbb{R}^{(n-r)\times r} \mapsto \mathcal{S}_Z, \qquad X \to Z + Z_\perp X$$

is bijective.

Proposition 2.2.3. *For each $W \in \mathcal{M}_r(\mathbb{R}^{n\times r})$ such that det$(Z^TW) \neq 0$, there exists a unique $G_W \in \mathrm{GL}_r$ such that*

$$W\mathrm{GL}_r \cap \mathcal{S}_Z = \{WG_W^{-1}\}$$

One can see the proofs of lemma 2.2.2 and proposition 2.2.3 in [8]. One straightforward result of proposition 2.2.3 is that $\mathrm{col}_{n,r} : \mathcal{S}_Z \to \mathbb{G}_r(\mathbb{R}^n)$ is a bijective map. Hence, for each $Z \in \mathcal{M}_r(\mathbb{R}^{n\times r})$, one can define a neighbourhood of $\mathrm{col}_{n,r}(Z)$ as follows,

$$\mathfrak{U}_Z = \mathrm{col}_{n,r}(\mathcal{S}) = \{\mathrm{col}_{n,r}(W) \mid W \in \mathcal{S}_Z\} \tag{2.7}$$

The open sets in (2.7) together with the bijective maps

$$\varphi_Z = (\mathrm{col}_{n,r} \circ \eta_Z)^{-1} : \mathfrak{U}_Z \to \mathbb{R}^{(n-r)\times r}, \tag{2.8}$$

such that

$$\varphi_Z^{-1}(X) = \mathrm{col}_{n,r}(Z + Z_\perp X)$$

form an atlas for $\mathbb{G}_r(\mathbb{R}^n)$. The charts defined in (2.8), clearly cover $\mathbb{G}_r(\mathbb{R}^n)$ and are proved to be compatible.

2.2.2 The Stiefel manifold $\mathcal{M}_r(\mathbb{R}^{n\times r})$

To build an atlas for a Stiefel manifold $\mathcal{M}_r(\mathbb{R}^{n\times r})$, one can define neighbourhoods of $Z \in \mathcal{M}_r(\mathbb{R}^{n\times r})$ as follows,

$$\mathcal{U}_Z = \{W \in \mathcal{M}_r(\mathbb{R}^{n\times r}) \mid \det(Z^TW) \neq 0\} \tag{2.9}$$

Clearly, the affine cross section $\mathcal{S}_Z$ is included in the open set $\mathcal{U}_Z$. Proposition 2.2.3 states that there exists a unique pair of $(X, G) \in \mathbb{R}^{(n-r)\times r} \times \mathrm{GL}_r$ such that for any $W \in \mathcal{S}_Z$, $WG^{-1} = Z + Z_\perp X$. Therefore, one can define the charts β_Z as follows,

$$\beta_Z : \mathcal{U}_Z \to \mathbb{R}^{(n-r)\times r} \times \mathrm{GL}_r \tag{2.10}$$

such that,

$$\beta_Z^{-1}(X, G) = (Z + Z_\perp X)G.$$

In particular,

$$\beta_Z^{-1}(0, id_r) = Z.$$

Clearly $\mathcal{U}_Z$ cover $\mathcal{M}_r(\mathbb{R}^{n\times r})$. Besides, in [8], one can see that $\{(\mathcal{U}_Z, \beta_Z)\}_{Z\in\mathcal{M}_r(\mathbb{R}^{n\times r})}$ are compatible charts and as a result, they form an atlas for $\mathcal{M}_r(\mathbb{R}^{n\times r})$. Moreover, it is proved that inclusion map $i : \mathcal{M}_r(\mathbb{R}^{n\times r}) \to \mathbb{R}^{n\times r}$ is an embedding so the non-compact Stiefel manifold is an embedded submanifold of $\mathbb{R}^{n\times r}$. Also, by restricting inclusion map to each neighbourhood $\mathcal{U}_Z$,

$$i \circ \beta_Z^{-1} : \mathbb{R}^{(n-r)\times r} \times \mathrm{GL}_r \mapsto \mathbb{R}^{n\times r}, \qquad (X, G) \to (Z + Z_\perp X)G,$$

the tangent map $T_Z i$ at $Z = \beta_Z^{-1}(0, id_r)$, defined by $T_Z i = D(i\circ\beta_Z^{-1})(0, id_r)$, is as follows,

$$T_Z i : \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{r\times r} \mapsto \mathbb{R}^{n\times r}, \qquad (\dot{X}, \dot{G}) \to (Z_\perp \dot{X} + Z\dot{G}) \tag{2.11}$$

with inverse given by,

$$(T_Z i)^{-1}(\dot{Z}) = (Z_\perp^+ \dot{Z}, Z^+ \dot{Z}),$$

where $Z^+ = (Z^T Z)^{-1} Z^T \in \mathcal{M}_r(\mathbb{R}^{r\times n})$ is the Moore-Penrose pseudo-inverse of Z.

2.2.3 The geometry of $\mathcal{M}_r(\mathbb{R}^{n\times m})$

One can decompose $Z \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ as follows,

$$Z = UGV^T, \tag{2.12}$$

where $U \in \mathcal{M}_r(\mathbb{R}^{n\times r})$, $V \in \mathcal{M}_r(\mathbb{R}^{m\times r})$ and $G \in \mathrm{GL}_r$. Note that the column space of Z is $\mathrm{col}_{n,r}(U)$ and the row space of Z is $\mathrm{col}_{m,r}(V)$. Now, we define the following surjective map,

$$\alpha_r : \mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathbb{G}_r(\mathbb{R}^n) \times \mathbb{G}_r(\mathbb{R}^m), \quad \alpha_r(Z) = (\mathrm{col}_{n,r}(U), \mathrm{col}_{m,r}(V)).$$

The set

$$\alpha_r^{-1}(\mathrm{col}_{n,r}(U), \mathrm{col}_{m,r}(V)) = \{UHV^T \mid H \in \mathrm{GL}_r\}$$

is identified with GL_r. Then, one can define a neighbourhood of $Z = UGV^T$,

$$\mathcal{U}_Z = \alpha_r^{-1}(\mathfrak{U}_U \times \mathfrak{U}_V)$$

where $\mathfrak{U}_U$ and $\mathfrak{U}_V$ are open sets of Grassman manifolds $\mathbb{G}_r(\mathbb{R}^n)$ and $\mathbb{G}_r(\mathbb{R}^m)$ containing U and V. According to (2.7) and (2.8), the neighbourhood of $Z = UGV^T \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ can be written as follows,

$$\mathcal{U}_Z = \{(U + U_\perp X)H(V + V_\perp Y)^T \mid (X, Y, H) \in \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathrm{GL}_r\}.$$

Then, one can define,

$$\varphi_Z : \mathcal{U}_Z \to \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathrm{GL}_r$$

such that

$$\varphi_Z^{-1}(X, Y, H) = (U + U_\perp X)H(V + V_\perp Y)^T. \tag{2.13}$$

In particular, $\varphi_Z^{-1}(0, 0, G) = Z$. In [8], it is proved that $\{(\mathcal{U}_Z, \varphi_Z) | (U, V, G) \in \mathcal{M}_r(\mathbb{R}^{n\times r}) \times \mathcal{M}_r(\mathbb{R}^{m\times r}) \times \mathrm{GL}_r\}$ is an atlas for $\mathcal{M}_r(\mathbb{R}^{n\times m})$. Therefore, $\mathcal{M}_r(\mathbb{R}^{n\times m})$ is a $(n + m - r)r$-dimensional manifold. It is also shown that $\mathcal{M}_r(\mathbb{R}^{n\times m})$ is an embedded submanifold of $\mathbb{R}^{n\times m}$. Then, by restricting inclusion map $i : \mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathbb{R}^{n\times m}$ to the neighbourhood $\mathcal{U}_Z$, we get,

$$\begin{aligned} i \circ \varphi_Z^{-1} : \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathrm{GL}_r &\to \mathbb{R}^{n\times m} \\ (X, Y, H) &\to (U + U_\perp X)H(V + V_\perp Y)^T. \end{aligned} \tag{2.14}$$

The tangent map $T_Z i$ at $Z = \varphi_Z^{-1}(0, 0, G)$, defined by $T_Z i = D(i \circ \varphi_Z^{-1})(0, 0, G)$ is,

$$\begin{aligned} T_Z i : \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r} &\to \mathbb{R}^{n\times m} \\ (\dot{X}, \dot{Y}, \dot{H}) &\to U_\perp \dot{X} G V^T + UG(V_\perp \dot{Y})^T + U\dot{H}V^T, \end{aligned} \tag{2.15}$$

with inverse given by,

$$(T_Z i)^{-1}(\dot{Z}) = (U_\perp^+ \dot{Z}(V^+)^T G^{-1}, V_\perp^+ \dot{Z}^T (U^+)^T G^{-T}, U^+ \dot{Z}(V^+)^T). \qquad (2.16)$$

Chapter 3

Optimization on matrix manifolds

Optimization on matrix manifolds consists of minimizing a real-valued cost function defined on a matrix manifold. Many optimization algorithms need to compute gradient and Hessian of the cost function. To define sensible gradient and Hessian, we need to have information about the geometry of matrix manifolds and their tangent spaces. As we are interested in optimizing objective functional $J : \mathbb{R}^{n\times m} \to \mathbb{R}$ the problem can be expressed as follows,

$$\min_{Z\in\mathcal{M}_r(\mathbb{R}^{n\times m})} J(Z).$$

In the section 2.2.2, it is shown that $\mathcal{M}_r(\mathbb{R}^{n\times m})$ admits a manifold structure induced by the local charts $\{(\mathcal{U}_Z, \varphi_Z)\}$ where $Z \in \mathcal{M}_r(\mathbb{R}^{n\times m})$. In this chapter, we discuss about two different algorithms on manifolds: the Riemannian algorithms and the chart based algorithms.

3.1 Riemannian algorithms

The goal of this section is presenting a steepest descent Riemannian algorithm. The Riemannian algorithms are provided such that the set $\mathcal{M}_r(\mathbb{R}^{n\times m})$ is treated as a embedded $(n + m - r)r$-submanifold of $\mathbb{R}^{n\times m}$. In the literature, many algorithms use Riemannian optimization. For instance, one can refer to [9] for a conjugate gradient algorithm, [10] for the Newton algorithm and [11] for trust-region methods.

Considering $\mathcal{M}_r(\mathbb{R}^{n\times m})$ as a submanifold of $\mathbb{R}^{n\times m}$, the tangent space $T_Z\mathcal{M}_r(\mathbb{R}^{n\times m})$ at $Z \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ is equipped with the induced Frobenius inner product on $\mathbb{R}^{n\times m}$ denoted by $\langle .,. \rangle_F$. Note that for any $A, B \in \mathbb{R}^{n\times m}$,

$$\langle A, B\rangle_F = \operatorname{tr}(A^T B) = \sum_{ij} a_{ij} b_{ij}$$

where $A = [a_{ij}]$ and $B = [b_{ij}]$. Hence, $\mathcal{M}_r(\mathbb{R}^{n\times m})$ together with $\langle .,. \rangle_F$ on $T\mathcal{M}_r(\mathbb{R}^{n\times m})$ form a Riemannian manifold. Now we are able to define the gradient of an objective function as the Rimannian metric is chosen.

Given a cost function $J : \mathbb{R}^{n\times m} \to \mathbb{R}$, the Riemannian gradient of J at $Z \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ is the unique vector in $T_Z\mathcal{M}_r(\mathbb{R}^{n\times m})$ denoted by $\nabla^R J(Z)$ such that,

$$\langle \nabla^R J(Z), \dot{Z}\rangle_F = DJ(Z)[\dot{Z}], \qquad \forall \dot{Z} \in T_Z\mathcal{M}_r(\mathbb{R}^{n\times m}). \tag{3.1}$$

Note that the gradient of J at Z treated in the set $\mathbb{R}^{n\times m}$ is the Euclidean gradient of J. Therefore, we get

$$\nabla^R J(Z) = P_{T_Z\mathcal{M}_r} \nabla J(Z)$$

where $\nabla J(Z)$ is the Euclidean gradient at Z viewed in $\mathbb{R}^{n\times m}$ and $P_{T_Z\mathcal{M}_r} : \mathbb{R}^{n\times m} \to T_Z\mathcal{M}_r(\mathbb{R}^{n\times m})$ is the orthogonal projection onto the tangent space. The orthogonal projection $P_{T_Z\mathcal{M}_r}$ of any $A \in \mathbb{R}^{n\times m}$ is defined as follows,

$$P_{T_Z\mathcal{M}_r} A = (id_n - UU^+)A(VV^+)^T + UU^+A(id_m - VV^+)^T + UU^+A(VV^+)^T,$$

where U and V are those in the equation (2.12) (see [9]).

Using the Riemannian gradient of the cost function defined in (3.1), we are able to find the direction in which the cost function decreases the most but we need a mapping that takes the points from tangent space to the manifold. Such a map is called a retraction and is defined on the tangent bundle $T\mathcal{M}_r(\mathbb{R}^{n\times m})$ as follows,

$$R : T\mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathcal{M}_r(\mathbb{R}^{n\times m}), \qquad (Z, \dot{Z}) \mapsto R(Z, \dot{Z})$$

One can refer to [12] to see the assumption that makes R a retraction. A shorthand notation for the retraction R is

$$R_Z : T_Z\mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathcal{M}_r(\mathbb{R}^{n\times m}), \qquad \dot{Z} \mapsto R(Z, \dot{Z})$$

In [9], The retraction is chosen as the metric projection,

$$R_Z : \mathcal{U}_Z \to \mathcal{M}_r(\mathbb{R}^{n\times m}), \qquad \dot{Z} \mapsto \arg \min_{W\in\mathcal{M}_r(\mathbb{R}^{n\times m})} \|Z + \dot{Z} - W\|_F$$

where $\mathcal{U}_Z \subseteq T_Z\mathcal{M}_r(\mathbb{R}^{n\times m})$ is a small enough neighbourhood around zero vector. This retraction can be obtained by the r-terms truncated singular value decomposition of $Z + \dot{Z}$.

according to what discussed in this section, we provide a Riemannian steepest descent algorithm to minimize a given cost function $J : \mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathbb{R}$.

Algorithm 3.1.1 (Riemannian steepest descent algorithm)**.**

```
1. Initialize $Z_0 \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ and the tolerance $\epsilon$.
2. For $k = 0, 1, \dots$ do
3. Compute $\nabla^R J(Z_k)$.
4. If $\|\nabla^R J(Z_k)\|_F < \epsilon$, then stop.
5. Determine $\rho_k$ by solving $\rho_k = \arg\min_\rho J(R_{Z_k}(-\rho\nabla^R J(Z_k)))$.
6. Set $Z_{k+1} = R_{Z_k}(-\rho_k\nabla^R J(Z_k))$.
```

3.2 Chart based algorithms

In this section, we provide optimization algorithms exploiting local coordinate charts of manifold $\mathcal{M}_r(\mathbb{R}^{n\times m})$. Given the cost function $J : \mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathbb{R}$, a natural optimization algorithm to minimize J is finding the sequence $\{Z_k\} \subset \mathcal{M}_r(\mathbb{R}^{n\times m})$ such that Z_{k+1}

is obtained by minimizing the cost function on the neighbourhood $\mathcal{U}_{Z_k}$ of Z_k. In other words,

$$Z_{k+1} = \arg \min_{W \in \mathcal{U}_{Z_k}} J(W). \tag{3.2}$$

According to the definition of local charts of $\mathcal{M}_r(\mathbb{R}^{n\times m})$ given in equations (2.13) and (2.14), the problem (3.2) is equivalent to $Z_{k+1} = \varphi_{Z_k}^{-1}(X_{k+1}, Y_{k+1}, H_{k+1})$, such that

$$(X_{k+1}, Y_{k+1}, H_{k+1}) = \arg \min_{(X,Y,H)\in\mathbb{R}^{(n-r)\times r}\times\mathbb{R}^{(m-r)\times r}\times \mathrm{GL}_r} \hat{J}_{Z_k}(X, Y, H), \tag{3.3}$$

where for any $Z \in \mathcal{M}_r(\mathbb{R}^{n\times m})$,

$$\hat{J}_Z = J \circ i \circ \varphi_Z^{-1} : \varphi_Z(\mathcal{U}_Z) \subset \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathrm{GL}_r \to \mathbb{R}.$$

3.2.1 Steepest descent algorithm

To define a steepest descent algorithm for the problem (3.3), we need to define the gradient of $\hat{J}_Z$ which is a vector in the parametric space $\mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r}$ according to the relation (2.15). Therefore, we should define a inner product $\langle ., . \rangle$ on $\mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r}$. A natural choice for inner product on the parametric space is

$$\langle (\dot{X}_1, \dot{Y}_1, \dot{H}_1), (\dot{X}_2, \dot{Y}_2, \dot{H}_2) \rangle = \langle \dot{X}_1, \dot{X}_2 \rangle_F + \langle \dot{Y}_1, \dot{Y}_2 \rangle_F + \langle \dot{H}_1, \dot{H}_2 \rangle_F \tag{3.4}$$

where $(\dot{X}_1, \dot{Y}_1, \dot{H}_1), (\dot{X}_2, \dot{Y}_2, \dot{H}_2) \in \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r}$.
With the given inner product in (3.4), the gradient of $\hat{Z}$ at $\varphi_Z(Z) = (0, 0, G)$ denoted by $\nabla \hat{J}_Z(0, 0, G)$ is such that for every $(\dot{X}, \dot{Y}, \dot{H}) \in \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r}$,

$$D\hat{J}_Z(0, 0, G)(\dot{X}, \dot{Y}, \dot{H}) = \langle \nabla \hat{J}_Z(0, 0, G), (\dot{X}, \dot{Y}, \dot{H}) \rangle. \tag{3.5}$$

Applying chain rule to the right hand side of equation (3.5) and according to (2.15) we get,

$$\begin{aligned}
D\hat{J}_Z(0,0,G)(\dot{X},\dot{Y},\dot{H}) &= DJ((i\circ\varphi_Z^{-1})(0,0,G))\big((D(i\circ\varphi_Z^{-1})(0,0,G))(\dot{X},\dot{Y},\dot{H})\big)\\
&= DJ(Z)(T_Z i(\dot{X},\dot{Y},\dot{H})) = \langle \nabla J(Z), T_Z i(\dot{X},\dot{Y},\dot{H})\rangle_F\\
&= \langle \nabla J(Z), U_\perp \dot{X} G V^T\rangle_F + \langle \nabla J(Z), UG(V_\perp \dot{Y})^T\rangle_F + \langle \nabla J(Z), U\dot{H}V^T\rangle_F\\
&= \langle U_\perp^T \nabla J(Z) V G^T, \dot{X}\rangle_F + \langle V_\perp^T \nabla J(Z)^T U G, \dot{Y}\rangle_F + \langle U^T \nabla J(Z) V, \dot{H}\rangle_F
\end{aligned}$$

where $\nabla J(Z)$ is the Euclidean gradient of J at Z. According to identifications (3.4) and (3.5), we obtain

$$\nabla \hat{J}(0,0,G) = (U_\perp^T \nabla J(Z) V G^T, V_\perp^T \nabla J(Z)^T U G, U^T \nabla J(Z) V). \tag{3.6}$$

The gradient of $\hat{J}_Z$ at $(0,0,G)$ obtained in (3.6), is based on the inner product presented in the equation (3.4). However, if we define the following inner product on $\mathbb{R}^{(n-r)\times r}\times\mathbb{R}^{(m-r)\times r}\times\mathbb{R}^{r\times r}$ for a given $Z\in\mathcal{M}_r(\mathbb{R}^{n\times m})$,

$$\langle(\dot{X}_1,\dot{Y}_1,\dot{H}_1),(\dot{X}_2,\dot{Y}_2,\dot{H}_2)\rangle = \langle T_Z i(\dot{X}_1,\dot{Y}_1,\dot{H}_1), T_Z i(\dot{X}_2,\dot{Y}_2,\dot{H}_2)\rangle_F,$$

where $(\dot{X}_1,\dot{Y}_1,\dot{H}_1),(\dot{X}_2,\dot{Y}_2,\dot{H}_2)\in\mathbb{R}^{(n-r)\times r}\times\mathbb{R}^{(m-r)\times r}\times\mathbb{R}^{r\times r}$, then according to the definition of Riemmannian gradient given in (3.1) we have,

$$\begin{aligned}
D\hat{J}_Z(0,0,G)(\dot{X},\dot{Y},\dot{H}) &= DJ(Z)(T_Z i(\dot{X},\dot{Y},\dot{H}))\\
&= \langle \nabla^R J(Z), T_Z i(\dot{X},\dot{Y},\dot{H})\rangle_F \quad = \langle P_{T_Z\mathcal{M}_r}\nabla J(Z), T_Z i(\dot{X},\dot{Y},\dot{H})\rangle_F.
\end{aligned}$$

Therefore, by identification (3.5),

$$\nabla \hat{J}_Z(0,0,G) = (T_Z i)^{-1} P_{T_Z\mathcal{M}_r}\nabla J(Z) = (T_Z i)^{-1}\nabla^R J(Z).$$

According to the definition of $(T_Z i)^{-1}$ given in (2.16) and the Riemannian gradient $\nabla^R J(Z)$, we obtain,

$$\nabla \hat{J}(0,0,G) = (U_\perp^+ \nabla J(Z)(V^+)^T G^{-1}, V_\perp^+ \nabla J(Z)^T (U^+)^T G^{-T}, U^+ \nabla J(Z)(V^+)^T). \tag{3.7}$$

The gradient of $\hat{J}_Z$ at $(0,0,G)$ defined in (3.7) corresponds to the Riemannian gradient defined in (3.1) but in the parametric space. A chart based steepest descent algorithm using $\nabla\hat{J}_Z(0,0,G)$ defined either in (3.6) or (3.7), is as follows,

Algorithm 3.2.1 (Chart based steepest descent algorithm)**.**

```
1. Initialize $Z_0 = U_0 G_0 V_0^T$ and the tolerance $\epsilon$.
2. For $k = 0, 1, \ldots$ do
3. Set $(\dot{X}_k, \dot{Y}_k, \dot{H}_k) = \nabla\hat{J}_{Z_k}(0,0,G_k)$.
4. Determine $\rho_k$ by solving $\rho_k = \arg\min_\rho \hat{J}_{Z_k}(-\rho\dot{X}_k, -\rho\dot{Y}_k, G_k - \rho\dot{H}_k)$.
5. Set $(X_{k+1}, Y_{k+1}, H_{k+1}) = (-\rho_k\dot{X}_k, -\rho_k\dot{Y}_k, G_k - \rho_k\dot{H}_k)$.
6. Set $Z_{k+1} = \varphi_{Z_k}^{-1}(X_{k+1}, Y_{k+1}, H_{k+1})$.
7. If $\frac{\|U_{k+1}-U_k\|}{\|U_{k+1}\|} + \frac{\|V_{k+1}-V_k\|}{\|V_{k+1}\|} + \frac{\|H_{k+1}-H_k\|}{\|H_{k+1}\|} \leq \epsilon$, then stop.
```

3.2.2 Newton algorithm on matrix manifolds

For a cost functional defined on a Euclidean space like $J : \mathbb{R}^n \to \mathbb{R}$, the Newton iteration function $N_J(x) : \mathbb{R}^n \to \mathbb{R}^n$ is defined as follow,

$$N_J(x) = x - [HessJ(x)]^{-1}\nabla J(x), \qquad (x \in \mathbb{R}^n)$$

where $HessJ(x)$ is the Hessian of J at x and $\nabla J(x)$ is the gradient of J at x, respectively. The function N_J does not depend on the inner product needed to define Hessian and gradient since the domain is Euclidean space. This iteration function generates the Newton iterates $x_{k+1} = N_J(x_k)$ where k is iteration number. It is proved that under some conditions (see for example [13]), the sequence $\{x_k\}$ is well-defined and convergent to a non-degenerate critical point $x \in \mathbb{R}^n$, that means $\nabla J(x) = 0$ and the inverse of the Hessian of J at all x_k exist.

To extend the Newton algorithm to optimize the functional $J : M \to \mathbb{R}$ defined on a manifold M, in [14], Manton proves that under some conditions on local parametrisations $\varphi_p : T_pM \to M$ and $\psi_p : T_pM \to M$, $p \in M$, the generalised Newton function $E_f : M \to M$ defined as

$$E_J(p) = \psi_p \circ N_{J\circ\varphi_p}(0_p) \tag{3.8}$$

converges locally quadratically to a non-degenerate critical point $\bar{p} \in M$. Here, the role of the second parametrisation ψ_p is to return to the manifold. In [14], it is also shown that parametrizations based on projections guarantee that choosing parametrisation φ_p instead of ψ_p in the map (3.8) is enough to get locally quadratic convergence.
Let $f : \mathcal{V} \to \mathbb{R}$ be a smooth functional on the linear space $\mathcal{V}$. The second derivative of f at $x \in \mathcal{V}$ is a bilinear mapping on $\mathcal{V} \times \mathcal{V}$ into $\mathbb{R}$ and denoted by $D^2f(x)$. The second derivative satisfies the symmetric property

$$D^2f(x)(v_1, v_2) = D^2f(x)(v_2, v_1), \qquad \forall v_1, v_2 \in \mathcal{V}$$

Then, The Hessian of f at $x \in \mathcal{V}$ is the unique symmetric operator $Hessf(x) : \mathcal{V} \to \mathcal{V}$ such that

$$\langle Hessf(x)(v_1), V_2\rangle = D^2f(x)(v_1, v_2), \qquad \forall v_1, v_2 \in \mathcal{V}$$

where $\langle ., .\rangle$ is the inner product defined on $\mathcal{V}$.
In order to apply the Newton algorithm to the optimization problem given in (3.3), we need to define $Hess\hat{J}_Z$ at $\varphi_Z(Z) = (0, 0, G) \in \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r}$. Let the inner product on $\mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r}$ be the same as the inner product defined in (3.4), so we can write,

$$\begin{aligned} D^2\hat{J}_Z(0, 0, G)((\dot{X}_1, \dot{Y}_1, \dot{H}_1), (\dot{X}_2, \dot{Y}_2, \dot{H}_2)) = \\ \langle Hess\hat{J}_Z(0, 0, G)(\dot{X}_1, \dot{Y}_1, \dot{H}_1), (\dot{X}_2, \dot{Y}_2, \dot{H}_2)\rangle. \end{aligned} \tag{3.9}$$

For convenience in writing, let $v_1 = (\dot{X}_1, \dot{Y}_1, \dot{H}_1)$ and $v_2 = (\dot{X}_2, \dot{Y}_2, \dot{H}_2)$. Applying the chain rule for the second derivative of $\hat{J}$ at $(0, 0, G)$ (see [15]) and according to the identification (2.15), we get

$$\begin{aligned}
&D^2\hat{J}_Z(0,0,G)(v_1,v_2) = \\
&\qquad D^2J((i\circ\varphi_Z^{-1})(0,0,G)\big((D(i\circ\varphi_Z^{-1})(0,0,G)(v_1),(D(i\circ\varphi_Z^{-1})(0,0,G)(v_2)\big) \\
&\qquad + DJ((D(i\circ\varphi_Z^{-1})(0,0,G))\big((D^2(i\circ\varphi_Z^{-1})(0,0,G)(v_1,v_2) \\
&\qquad = D^2J(Z)(T_Zi(v_1),T_Zi(v_2)) + DJ(Z)\big(D^2(i\circ\varphi_Z^{-1})(0,0,G)(v_1,v_2)\big) \\
&\qquad = \langle HessJ(Z)(T_Zi(v_1)),T_Zi(v_2)\rangle_F + \langle\nabla J(Z),D^2(i\circ\varphi_Z^{-1})(0,0,G)(v_1,v_2)\rangle_F
\end{aligned} \tag{3.10}$$

where $HessJ(Z)$ and $\nabla J(Z)$ are the Euclidean Hessian and gradient of J at $Z \in \mathbb{R}^{n\times m}$ respectively. For convenience, let $\mathcal{H} = HessJ(Z)$. Then, according to the equation (2.15), the first term in the right hand side of the equation (3.10) can be written as follows,

$$\begin{aligned}
&\langle\mathcal{H}(T_Zi(\dot{X}_1,\dot{Y}_1,\dot{H}_1)),T_Zi(\dot{X}_2,\dot{Y}_2,\dot{H}_2)\rangle_F = \\
&\langle U_\perp^T\mathcal{H}(U_\perp\dot{X}_1GV^T)VG^T,\dot{X}_2\rangle_F + \langle V_\perp^T\mathcal{H}(U_\perp\dot{X}_1GV^T)^TUG,\dot{Y}_2\rangle_F + \langle U^T\mathcal{H}(U_\perp\dot{X}_1GV^T)V,\dot{H}_2\rangle_F \\
&+\langle U_\perp^T\mathcal{H}(UG\dot{Y}_1^TV_\perp^T)VG^T,\dot{X}_2\rangle_F + \langle V_\perp^T\mathcal{H}(UG\dot{Y}_1^TV_\perp^T)^TUG,\dot{Y}_2^T\rangle_F + \langle U^T\mathcal{H}(UG\dot{Y}_1^TV_\perp^T)V,\dot{H}_2\rangle_F \\
&+\langle U_\perp^T\mathcal{H}(U\dot{H}_1V^T)VG^T,\dot{X}_2\rangle_F + \langle V_\perp^T\mathcal{H}(U\dot{H}_1V^T)^TUG,\dot{Y}_2^T\rangle_F + \langle U^T\mathcal{H}(U\dot{H}_1V^T)V,\dot{H}_2\rangle_F
\end{aligned}$$

In the equation (3.10), The bilinear mapping $D^2(i\circ\varphi_Z^{-1})(0,0,G) : T_Z\mathcal{M}_r(\mathbb{R}^{n\times m}) \times T_Z\mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathbb{R}^{n\times m}$ is defined as follows,

$$\begin{aligned}
&D^2(i\circ\varphi_Z^{-1})(0,0,G)((\dot{X}_1,\dot{Y}_1,\dot{H}_1),(\dot{X}_2,\dot{Y}_2,\dot{H}_2)) = \\
&\qquad\qquad U_\perp\dot{X}_1G(V_\perp\dot{Y}_2)^T + U_\perp\dot{X}_1\dot{H}_2V^T + U_\perp\dot{X}_2G(V_\perp\dot{Y}_1)^T \\
&\qquad\qquad + U\dot{H}_2(V_\perp\dot{Y}_1)^T + U_\perp\dot{X}_2\dot{H}_1V^T + U\dot{H}_1(V_\perp\dot{Y}_2)^T.
\end{aligned}$$

Therefore, the second product in the right hand side of identification (3.10) can be written as follows,

$$\begin{aligned}\langle \nabla J(Z), D^2(i\circ\varphi_Z^{-1})(0,0,G)((\dot X_1,\dot Y_1,\dot H_1),(\dot X_2,\dot Y_2,\dot H_2))\rangle_F = \\ \langle V_\perp^T\nabla J(Z)^T U_\perp \dot X_1 G, \dot Y_2\rangle_F + \langle \dot X_1^T U_\perp^T \nabla J(Z) V, \dot H_2\rangle_F \\ + \langle U_\perp^T \nabla J(Z) V_\perp \dot Y_1 G^T, \dot X_2\rangle_F + \langle U^T\nabla J(Z) V_\perp \dot Y_1, \dot H_2\rangle_F \\ + \langle U_\perp^T\nabla J(Z) V \dot H_1^T, \dot X_2\rangle_F + \langle V_\perp^T \nabla J(Z)^T U \dot H_1, \dot Y_2\rangle_F\end{aligned}$$

Therefore, according to the equality (3.9), for any vector $(\dot X,\dot Y,\dot H)$ in the parametric space, we have

$$Hess\hat J_Z(0,0,G)(\dot X,\dot Y,\dot H) = (h_1,h_2,h_3)\in\mathbb{R}^{(n-r)\times r}\times\mathbb{R}^{(m-r)\times r}\times\mathbb{R}^{r\times r} \tag{3.11}$$

where

$$\begin{aligned}h_1 &= U_\perp^T[\mathcal{H}(U_\perp\dot X_1 G V^T)+\mathcal{H}(UG\dot Y_1^T V_\perp^T)+\mathcal{H}(U\dot H_1 V^T)]VG^T \\ &\quad + U_\perp^T\nabla J(Z)V_\perp \dot Y_1 G^T + U_\perp^T \nabla J(Z) V\dot H_1^T,\end{aligned}$$

$$\begin{aligned}h_2 &= V_\perp^T[\mathcal{H}(U_\perp\dot X_1 G V^T)^T+\mathcal{H}(UG\dot Y_1^T V_\perp^T)^T+\mathcal{H}(U\dot H_1 V^T)^T]UG \\ &\quad + V_\perp^T\nabla J(Z)^T U_\perp \dot X_1 G + V_\perp^T \nabla J(Z)^T U\dot H_1,\end{aligned}$$

$$\begin{aligned}h_3 &= U^T[\mathcal{H}(U_\perp\dot X_1 G V^T)+\mathcal{H}(UG\dot Y_1^T V_\perp^T)+\mathcal{H}(U\dot H_1 V^T)]V \\ &\quad + \dot X_1^T U_\perp^T\nabla J(Z)V + U^T \nabla J(Z) V_\perp\dot Y_1.\end{aligned}$$

The definition of Hessian operator of $\hat J$ given in (3.11) is not practically useful since we need the inverse of the operator $Hess\hat J_Z(0,0,G)$ when implementing Newton method not the value of this operator acting on an arbitrary vector in the parametric space. To find a matrix expression for the operator $Hess\hat J_Z(0,0,G)$, firstly, for any matrices $A\in\mathbb{R}^{n\times l}$, $B\in\mathbb{R}^{m\times p}$ and $C\in\mathbb{R}^{l\times p}$, we define,

$$ACB^T = (A\otimes B)C$$

where the operator $A \otimes B : \mathbb{R}^{l \times p} \to \mathbb{R}^{n \times m}$ is the tensor product of A and B. Then, the first inner product of $\langle HessJ(Z)(T_Z i(v_1)), T_Z i(v_2)\rangle_F$ in the equality (3.10) can be written as follows,

$$\langle \mathcal{H}(U_\perp \dot{X}_1 G V^T), U_\perp \dot{X}_2 G V^T\rangle_F = \langle (U_\perp \otimes V G^T)^T \mathcal{H}(U_\perp \otimes V G^T)\dot{X}_1, \dot{X}_2\rangle_F,$$

where the operator $(U_\perp \otimes V G^T)^T \mathcal{H}(U_\perp \otimes V G^T)$ is the operator defined on $\mathbb{R}^{(n-r)\times r}$ onto $\mathbb{R}^{(n-r)\times r}$. Similarly we can find all the operators related to each term of the equation (3.10). Finally, the matrix representation of the Hessian operator of $\hat{J}_Z$ at $(0,0,G)$ for any vector $(\dot{X}, \dot{Y}, \dot{H}) \in \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r \times r}$ is written as follows,

$$Hess\hat{J}_Z(0,0,G)(\dot{X}, \dot{Y}, \dot{H}) = H_{1Z}\begin{bmatrix}\text{vec}(\dot{X})\\ \text{vec}(\dot{Y})\\ \text{vec}(\dot{H})\end{bmatrix} + H_{2Z}\begin{bmatrix}\text{vec}(\dot{X}^T)\\ \text{vec}(\dot{Y}^T)\\ \text{vec}(\dot{H}^T)\end{bmatrix}, \tag{3.12}$$

where 'vec' is the vector operator explained in the Example 2.1.5, and H_{1Z} and H_{2Z} are two 3×3-block matrices defined as follow,

$$H_{1Z} = \begin{bmatrix} (U_\perp \otimes V G^T)^T \mathcal{H}(U_\perp \otimes V G^T) & U_\perp^T \nabla J(Z) V_\perp \otimes G & (U_\perp \otimes V G^T)^T \mathcal{H}(U \otimes V) \\ V_\perp^T \nabla J(Z)^T U_\perp \otimes G^T & (V_\perp \otimes U G)^T \mathcal{H}^T (V_\perp \otimes U G) & V_\perp^T \nabla J(Z)^T U \otimes id_r \\ (U \otimes V)^T \mathcal{H}(U_\perp \otimes V G^T) & U^T \nabla J(Z) V_\perp \otimes id_r & (U \otimes V)^T \mathcal{H}(U \otimes V) \end{bmatrix}$$

$$H_{2Z} = \begin{bmatrix} 0 & (U_\perp \otimes V G^T)^T \mathcal{H}(U G \otimes V_\perp) & U_\perp^T \nabla J(Z) V \otimes id_r \\ (V_\perp \otimes U G)^T \mathcal{H}^T (V G^T \otimes U_\perp) & 0 & (V_\perp \otimes U G)^T \mathcal{H}^T (V \otimes U) \\ id_r \otimes V^T \nabla J(Z)^T U_\perp & (U \otimes V)^T \mathcal{H}(U G \otimes V_\perp) & 0 \end{bmatrix}$$

Now, we can provide a Newton algorithm to optimize the cost functional $J : \mathcal{M}_r(\mathbb{R}^{n\times m}) \to \mathbb{R}$ using manifold structure of $\mathcal{M}_r(\mathbb{R}^{n\times m})$.

Algorithm 3.2.2 (Newton algorithm on matrix manifolds)**.**

1. *Initialize $Z_0 = U_0 G_0 V_0^T$ and set the tolerance ϵ.*

2. *For $k = 0, 1, \ldots$ do*

3. *Set $(\dot{X}_k, \dot{Y}_k, \dot{H}_k) = \nabla \hat{J}_{Z_k}(0, 0, G_k)$.*

4. *Set* $(X_{k+1}, Y_{k+1}, H_{k+1}) = (0, 0, G_k) - [Hess\hat{J}_{Z_k}(0, 0, G_k)]^{-1}(\dot{X}_k, \dot{Y}_k, \dot{H}_k)$.

5. *Set* $Z_{k+1} = \varphi_Z^{-1}(X_{k+1}, Y_{k+1}, H_{k+1})$.

6. *If* $\frac{\|U_{k+1}-U_k\|}{\|U_{k+1}\|} + \frac{\|V_{k+1}-V_k\|}{\|V_{k+1}\|} + \frac{\|H_{k+1}-H_k\|}{\|H_{k+1}\|} \leq \epsilon$, *then stop.*

Chapter 4

Numerical experiments

In this chapter, we implement the Newton Algorithm 3.2.2 for three problems. These problems are dimension reduction for partial differential equations, model reduction for parametric PDEs and matrix completion. Moreover, we compare the steepest descent Algorithm 3.1.1 with the Newton Algorithm 3.2.2 for the matrix completion problem.

4.1 Dimension reduction for PDEs

We consider the Poisson's equation

$$\begin{cases} \Delta u(x,y) = 1 & \text{on } (0,1)\times(0,1) \\ u(x,y) = 0 & \text{on the boundary} \end{cases} \tag{4.1}$$

where Δ is the Laplace operator. We discretize the equation (4.1) using a finite difference scheme on a regular grid which yields an equation

$$(K \otimes id_n + id_n \otimes K)Z = f \tag{4.2}$$

where n is the number of nodes in each dimension, $Z \in \mathbb{R}^{n\times n}$ is the matrix corresponding to the values of $u(x,y)$ at the nodes, K is the matrix corresponding to the 1-dimensional Laplace operator discretized with finite difference scheme, and f is a $n \times n$-matrix whose all entries are equal to 1. The problem can be interpreted as the minimizing the following functional,

$$J(Z) = \frac{1}{2}\langle Z, AZ\rangle_F - \langle Z, f\rangle_F \tag{4.3}$$

where the operator A is equal to $(K \otimes id_n + id_n \otimes K)$. Clearly,

$$\begin{aligned} \nabla J &= AZ - f, \\ Hess J &= A. \end{aligned} \tag{4.4}$$

We look for an approximation of the solution, still denoted Z in the manifold $\mathcal{M}_r(\mathbb{R}^{n\times n})$, and the Newton Algorithm 3.2.2 is implemented in the parametric space $\mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{r\times r}$ where we need to compute both Hessian and gradient of $\hat{J}_Z = J \circ i \circ \varphi_Z^{-1}$ given in the sections 3.2.1 and 3.2.2. Note that $\{(\mathcal{U}_Z, \varphi_Z)\}$ are the local charts defined in the section 2.2.3.

The implementation of Algorithm 3.2.2 is done in MATLAB R2013a on a computer with a 3.20 GHz CPU.

Initial guess. Using a random initial guess $Z \in \mathcal{M}_r(\mathbb{R}^{n\times n})$ where r is fixed does not show any convergence. On the other hand, since the goal of this implementation is testing the Newton algorithm in the parametric space, we use an almost optimized solution of (4.4) obtained by a greedy algorithm (see [16]). The initial guess obtained by greedy algorithm for the equation (4.2), has the rank $r = 7$ at most.

Stopping criterion. Let $Z_k = U_k G_k V_k^T$ where k is the iteration number, $U_k, V_k \in \mathcal{M}_r(\mathbb{R}^{n\times r})$ and $G_k \in \mathrm{GL}_r$. We define the stagnation criterion as follow,

$$\text{Stagnation}(k) = \frac{\|U_{k+1} - U_k\|_F}{\|U_{k+1}\|_F} + \frac{\|V_{k+1} - V_k\|_F}{\|V_{k+1}\|_F} + \frac{\|G_{k+1} - G_k\|_F}{\|G_{k+1}\|_F}. \tag{4.5}$$

The tolerance of stagnation is fixed at 10^{-12} for all implementations. Besides, the error is considered as follows,

$$\text{Error}(k) = \frac{\|AZ_k - f\|_F}{\|f\|_F} = \frac{\|\nabla J(Z_k)\|_F}{\|\nabla J(0)\|_F}. \tag{4.6}$$

Solvers. Since in each iteration, we need to solve the linear system,

$$[Hess\hat{J}_{Z_k}(0, 0, G_k)](X_{k+1}, Y_{k+1}, H_{k+1}) = [\nabla \hat{J}_{Z_k}(0, 0, G_k)]$$

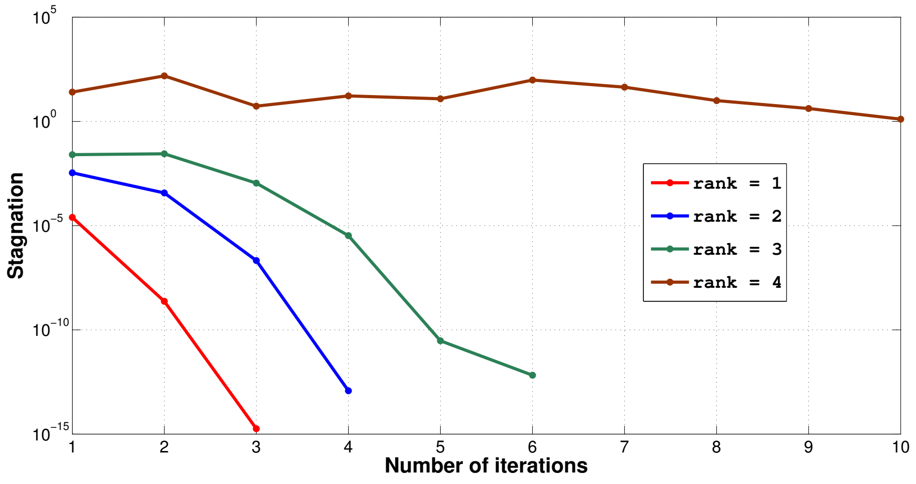

Figure 4.1: Convergence plot of Algorithm 3.2.2 for different ranks (using direct solver).

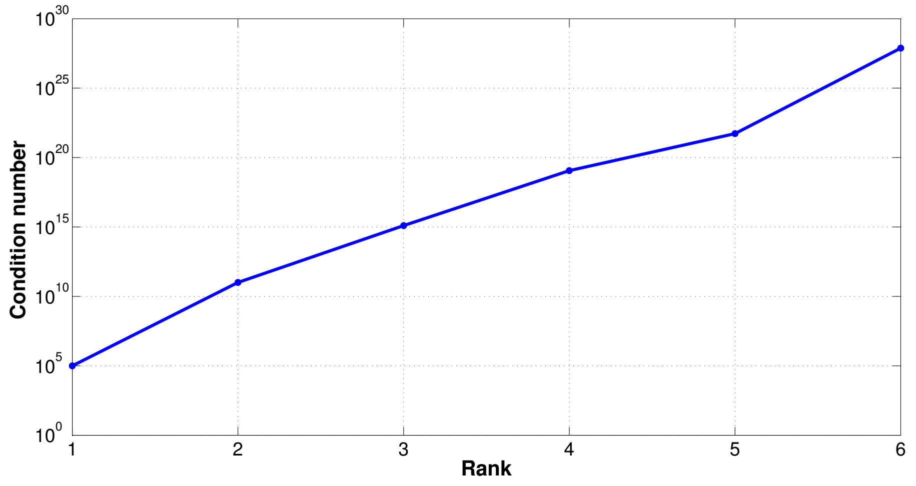

Figure 4.2: Condition number of $Hess\hat{J}_{Z_0}(0, 0, G_0)$

where $Hess\hat{J}_k(0, 0, G_k)$ is given in the equation (3.12), we can use either a direct solver computing the inverse of $Hess\hat{J}_{Z_k}(0, 0, G_k)$ directly or an iterative solver. The iterative solver that we use in this work, is called preconditioned conjugate gradient (PCG). For Poisson's problem, there were no improvement using PCG rather than using direct solver.

The number of nodes is fixed to $n = 100$ and the maximum iteration is set to 10. The

number of iterations to reach the tolerance for different ranks $r = 1, \ldots, 4$ is shown in the Figure 4.1. For $r = 4$, there is no convergence due to the increase of condition number of matrix $Hess\hat{J}$ when the rank increases. For $r = 1$ also, the solution obtained in the first iteration is very optimal due to the optimal initial guess.
The Figure 4.2 shows how $Hess\hat{J}_{Z_0}(0, 0, G_0)$ becomes ill-conditioned when we increase the rank.

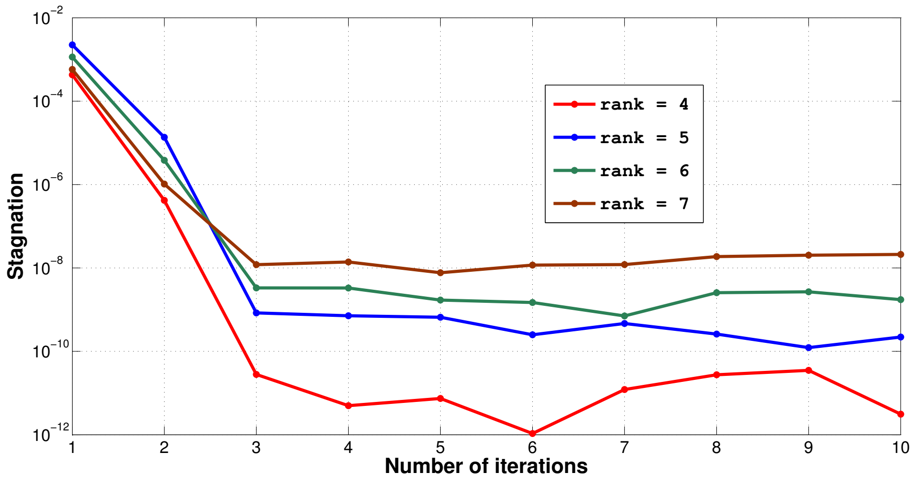

Figure 4.3: Convergence plot of Algorithm 3.2.2 for different ranks (using direct solver).

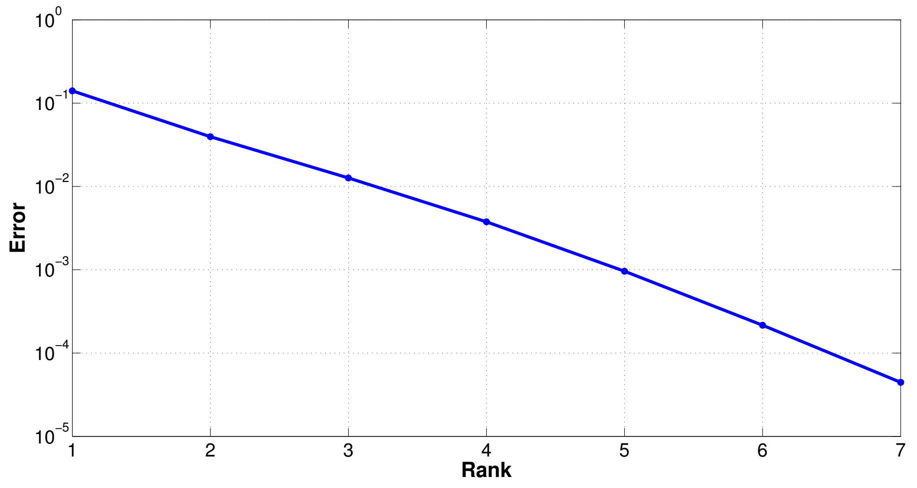

Figure 4.4: Error for different ranks.

Using iterative solver PCG does not improve convergence in this case. Therefore, we used a more optimal initial guess to overcome this problem. One can see the convergence for higher ranks in Figure 4.3. In this case, the stagnation criterion does not reach the selected tolerance after maximum number of iterations but at least we observe improvement compared to the initial guess.

Finally, Figure 4.4 shows the error defined in the equation (4.6) with respect to the rank. The higher fixed rank, the less error.

4.2 Model reduction for parameter-dependent equations

We consider the following reaction-diffusion equation,

$$\begin{cases} -\Delta u + \xi u = 1 & \text{on } (0,1)\times(0,1) \\ u = 0 & \text{on the boundary} \end{cases} \tag{4.7}$$

where ξ is the reaction parameter. Let the solution $u(x,y)$ be written as a separated function. After discretization in space and parameter domain, an approximation of u is defined by

$$(K_x \otimes id_m + id_n \otimes K_\xi)Z = f \tag{4.8}$$

where n is the number of nodes in the spacial domain, m is the number of reaction parameter values chosen randomly between 0 and 50, K_x is the matrix corresponding to the 2-dimensional Laplace operator discretized with finite difference scheme, K_ξ is a diagonal matrix filled with reaction parameters and f is a $n \times m$-matrix whose all entries are all equal to 1.

The functional $J : \mathbb{R}^{n\times m} \to \mathbb{R}$, ∇J and $HessJ$ are defined similar to those defined in the section 4.1, but the operator A is replaced with $K_x \otimes id_m + id_n \otimes K_\xi$. The Newton method is implemented in parametric space $\mathbb{R}^{(n-r)\times r} \times \mathbb{R}^{(m-r)\times r} \times \mathbb{R}^{r\times r}$ and the local functional $\hat{J}_Z$ and its gradient and Hessian are defined as in section 4.1.

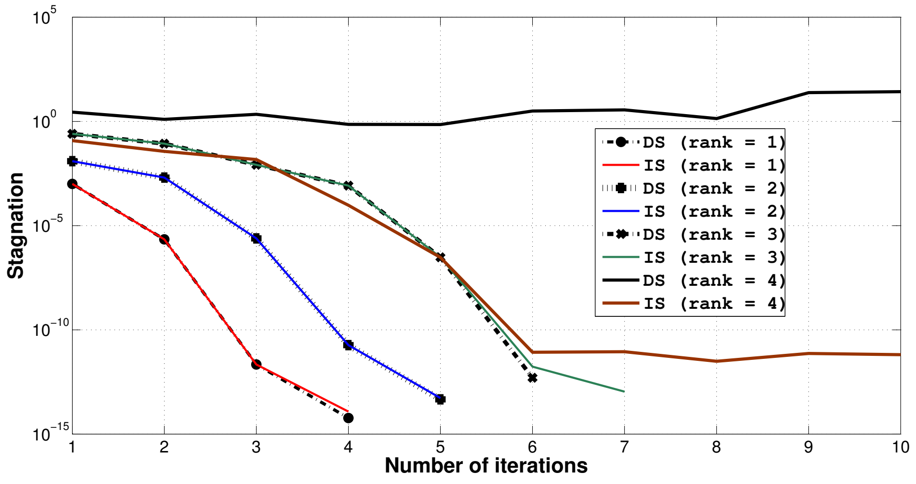

Figure 4.5: Convergence plot of Algorithm 3.2.2 for different ranks using both direct solver (DS) and IS-PCG (IS).

The initial guess, stagnation criterion, error and solvers are chosen similarly those in the section 4.1. The only difference is now $Z \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ is a rectangular matrix. All tests except the last one, is done for $n = 400$ and $m = 100$ while we change r.

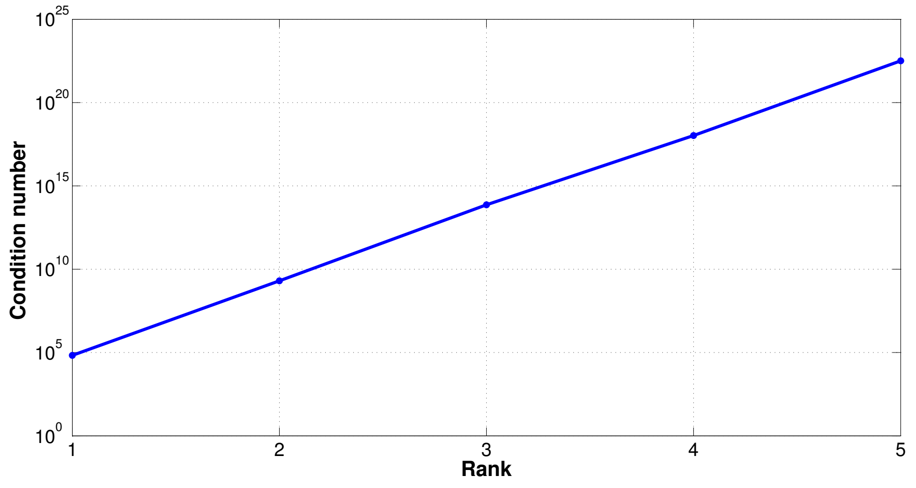

Figure 4.6: Condition number of $Hess\hat{J}_{Z_0}(0,0,G_0)$

The Figure 4.5 shows the convergence plot for different ranks for both direct solver (DS) and iterative solver (IS-PCG). For $r = 1, 2, 3$, both direct solver and IS-PCG show the same convergence behaviour while for $r = 4$, the solution obtained by DS is divergent.

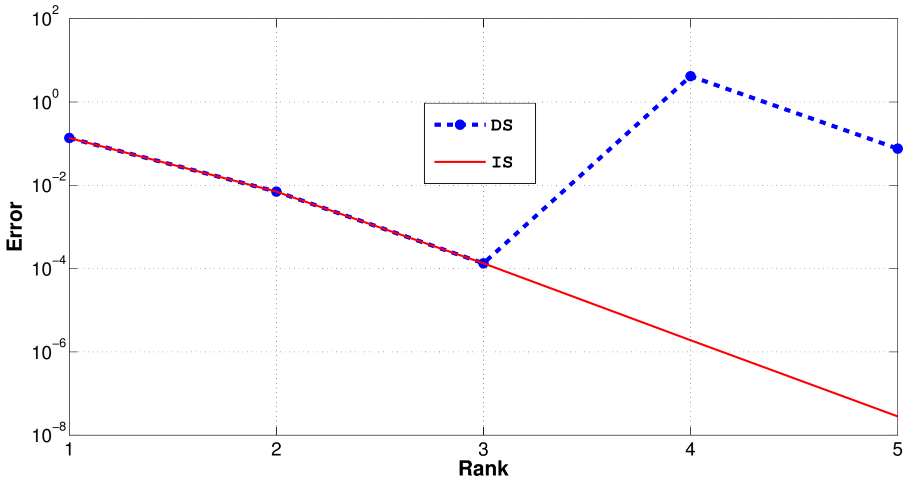

Figure 4.7: The least error reached for different ranks with both DS and IS.

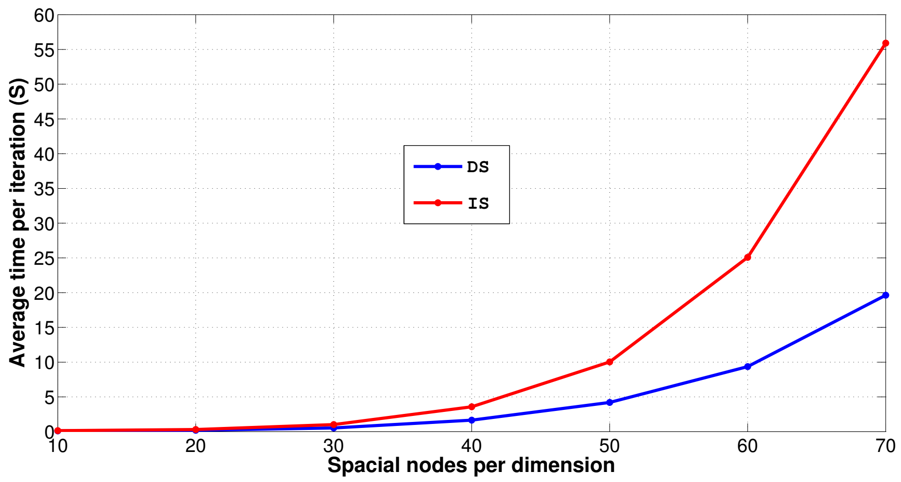

Figure 4.8: Computation time per iteration of matrices with different sizes for a fixed rank $r = 2$.

The Figure 4.7 shows the error with respect to each rank. For $r > 3$, DS is divergent. Finally we test the run time of the Algorithm 3.2.2 for different sizes of Z. To do this, we fix $r = 2$, $m = 100$ and increase n from 100 to 4900. The Figure 4.8 shows the run time plot for both DS and IS-PCG. The run time of IS-PCG is higher since we use an inner iteration loop for preconditioning $Hess\hat{J}_{Z_k}(0, 0, G_k)$.

4.3 Matrix completion

The matrix completion problem refers to filling unknown entries of a matrix partially observed. In this problem, $\Omega \subseteq \{1, \dots, n\} \times \{1, \dots, m\}$ represents the set of indices of known entries called the sample set. In practice, the sample set usually consists of many entries of the matrix. Let $A \in \mathbb{R}^{n\times m}$. Then, the low-rank matrix completion is finding a matrix with the lowest possible rank that is equal to A on the sample set, solution of

$$\begin{aligned} &\min_{Z\in\mathbb{R}^{n\times m}} \operatorname{rank}(Z) \\ &\text{such that} \qquad P_\Omega(Z) = P_\Omega(A), \end{aligned} \tag{4.9}$$

where,

$$P_\Omega(Z) = \begin{cases} A_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{if } (i,j) \notin \Omega \end{cases} \tag{4.10}$$

is the orthogonal projection of Z onto Ω. Unfortunately, this problem is an NP-hard non-convex optimization problem. In [17], Candes and Tao prove that replacing rank(Z) by the nuclear norm (sum of the singular values) i.e. considering relaxed problem,

$$\begin{aligned} &\min_{Z\in\mathbb{R}^{n\times m}} \|Z\|_* \\ &\text{such that} \qquad P_\Omega(Z) = P_\Omega(A), \end{aligned}$$

can allow us to recover with high probability any low rank matrix A satisfying certain conditions if Ω is sampled uniformly at random and $|\Omega|$ is large enough. In [18, 19, 20], one can find the algorithms for solving the relaxed problem by methods that exploit the low-rank structure. Moreover, one can see in [21, 22, 23] the other approaches like optimization on Grassmann manifolds or see in [24] the atomic decomposition.
As an alternative to the problem (4.9), the matrix completion problem can be considered as follows (see for example [9]),

$$\min_{Z\in\mathcal{M}_r(\mathbb{R}^{n\times m})} J(Z) = \frac{1}{2}\|P_\Omega(Z - A)\|_F^2. \tag{4.11}$$

We should minimize the functional J in (4.11) over all fixed-rank matrices Z such that $r \ll m < n$. Implementing Algorithm 3.2.2 to this problem, we need the Hessian and gradient of the functional $\hat{J}_Z$ at $(0, 0, G)$ in the parametric space. Therefore, we need to know $\nabla J(Z)$ and $\mathit{Hess}J(Z)$. Clearly, the Euclidean gradient is as follows,

$$\nabla J(Z) = P_\Omega(Z - A).$$

Moreover, one can write the projection $P_\Omega : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$, identified with an element of $\mathbb{R}^{n\times n} \otimes \mathbb{R}^{m\times m}$, based on the basis matrices of $\mathbb{R}^{n\times n}$ and $\mathbb{R}^{m\times m}$ as follows,

$$P_\Omega = \sum_{(i,j)\in\Omega} e_{ii} \otimes e_{jj}. \tag{4.12}$$

Therefore, we can write,

$$\text{Hess } J(Z) = \sum_{(i,j)\in\Omega} e_{ii} \otimes e_{jj}. \tag{4.13}$$

Guessing rank. There are some strategies to guess the rank in the fixed-rank optimization algorithms, for example, one can see [25, 26]. A standard strategy is starting from $r = 1$ and increasing the rank till there is no improvement detected.

The matrix $A \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ in the equation (4.11) has constructed by two full-rank matrices $A_l \in \mathbb{R}^{n\times r}$, $A_r \in \mathbb{R}^{m\times r}$ with i.i.d standard Gaussian entries such that $A = A_l A_r^T$.

Oversampling factor. As the dimension of the manifold $\mathcal{M}_r(\mathbb{R}^{n\times m})$ is equal to $(n + m - r)r$, we need to know at least $(n + m - r)r$ entries of A to recover it. We define the oversampling factor as the following ratio,

$$mboxOS = \frac{|\Omega|}{(n + m - r)r}$$

Clearly, to find missing entries, OS ≥ 1. On the other hand, at least one entry of each row and each column should be sampled to make sure of recovering A. Therefore, we consider OS sufficiently large to make sure that each column and row sampled at least once.

Initial guess. We try to construct the initial matrix $Z_0 \in \mathcal{M}_r(\mathbb{R}^{n\times m})$ randomly in the same way as matrix A but there is no convergence observed. Thus, we choose the initial guess Z_0 obtained by 15 to 20 iterations of Algorithm 3.1.1 to make sure that the Newton algorithm converges.

The stagnation and error is defined the same way as those given in the section 4.1. The maximum number of iterations and the tolerance of stagnation is set to 10 and 10^{-12}, respectively. Moreover, we use DS and IS-PCG for our experiments. We consider four different cases to test the Algorithm 3.2.2. In the all cases, the sample set is chosen uniformly at random except for the last case where we consider an biased sample set.

Case 1 (Comparing the Algorithm 3.2.2 to the Algorithm 3.1.1 on a rectangular matrix). The first test is to compare the error obtained by the Newton Algorithm 3.2.2 using direct solver (Newton Al (DS)) and the steepest descent Algorithm 3.1.1 obtained by Riemannian optimization (SD Riemannian). The parameters is set $n = 1000$, $m = 100$, $r = 8$ and OS $= 3$. Figure 4.9 shows the result for 30 iterations. The Newton Algorithm 3.2.2 needs just 6 iterations to reach the error 10^{-9} while after 30 iterations the algorithm 3.1.1 have not even reach to error 10^{-3}. Although the number of iterations using Algorithm 3.2.2 is incredibly less than the Algorithm 3.1.1, the cost of computations for each iteration of the Newton algorithm is very high.

Case 2 (Square matrix). We set $n = 1000$, $m = 1000$, $r = 15$ and OS $= 3$. The Figures 4.10 and 4.11 visualise the residual and error convergence plot of the Newton algorithm for both DS and IS. Both solvers convergence behaviour is the same. Besides, we see that the error does not decrease less than 10^{-9} though the residual decreasing.

Case 3 (Square matrix with low oversampling factor). In this case, we consider the case 2 but with OS $= 2$. The goal of this test is to check if the Algorithm 3.2.2 works well knowing much fewer entries. Figures 4.12 and 4.13 show the residual and error convergence plot of the Newton algorithm for both DS and IS. Like the case 2, there is no difference between convergence behaviour using DS and IS but we see a

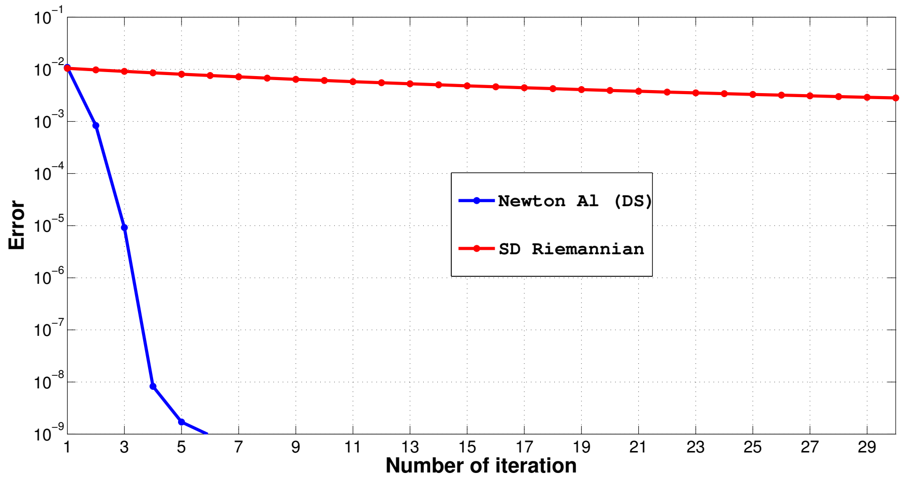

Figure 4.9: Error Comparison between the Newton algorithm using direct solver (Newton Al(DS)) and Riemannian steepest descent method (SD Riemannian) for the first 30 iterations

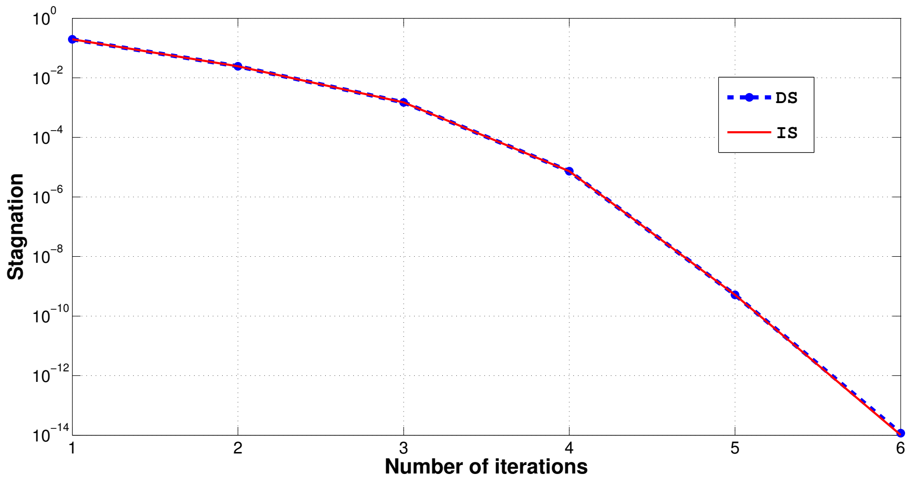

Figure 4.10: Stagnation convergence plot of Newton algorithm using direct solver(DS) and iterative solver PCG(IS)(OS = 3)

reduction of the number of iterations from 6 to 5 when we reduce oversampling factor while we could have expected increasing. Decreasing iterations in this case is due to using a more optimal initial guess obtained by the Algorithm 3.1.1 to make sure that the Newton algorithm converges with lower oversampling factor.

Case 4 (Biased sampling). For the last test, we set $n = 1500$, $m = 300$, $r = 6$ and

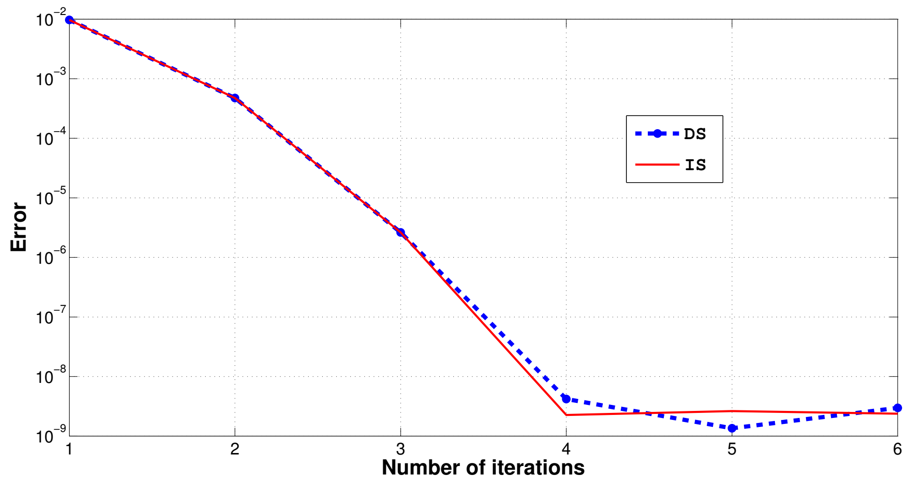

Figure 4.11: Error convergence plot of Newton algorithm using direct solver(DS) and iterative solver PCG(IS)(OS = 3)

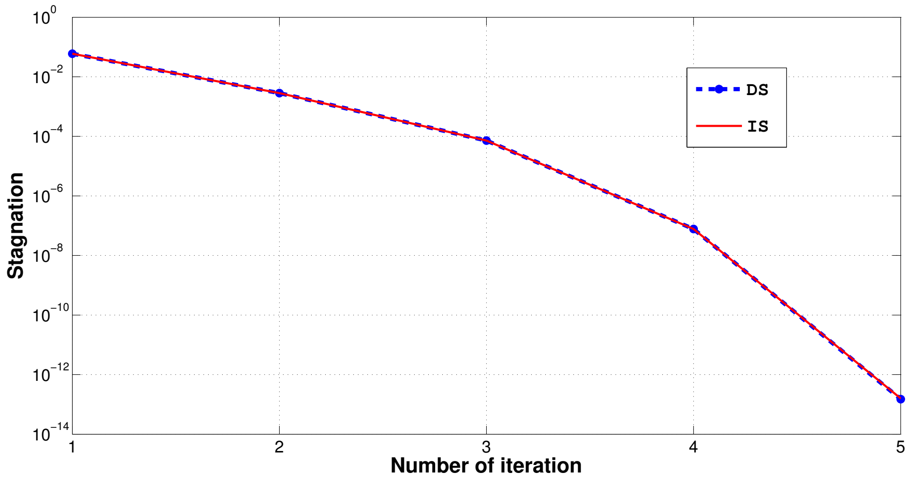

Figure 4.12: Stagnation convergence plot of Newton algorithm using direct solver(DS) and iterative solver PCG(IS)(OS = 2)

use an artificial sampling. This test imitates the well-known Netflix problem. Netflix problem, refers to a matrix whose the (i, j)-th entry is the rate given by a costumer i to a movie j. Since this matrix is incomplete, the goal is to predict all other missing entries. Consider m corresponds to the movies and n corresponds to the viewers. The sample set is chosen such that the first 30 movies are rated with the lowest probability by viewers, then the middle 240 movies are rated with 3 times higher probability of the

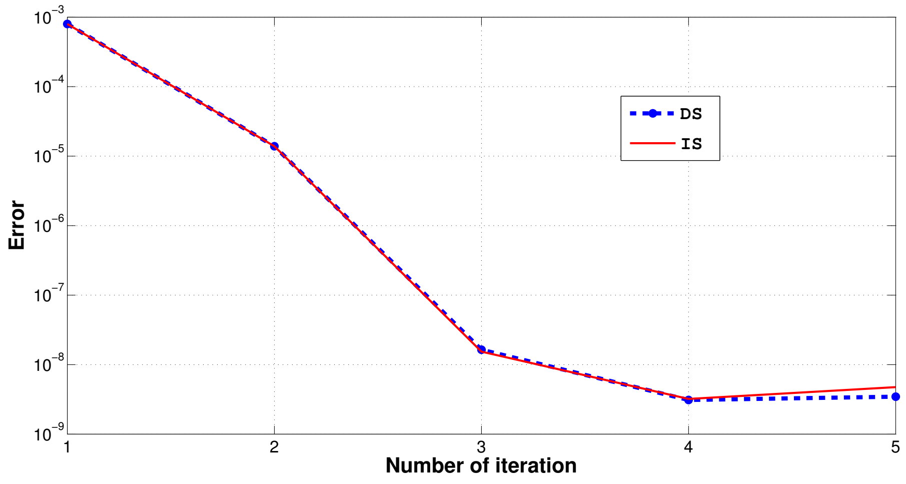

Figure 4.13: Error convergence plot of Newton algorithm using direct solver(DS) and iterative solver PCG(IS)(OS = 2)

first 30 movies and finally the last 30 remaining movies are rated with 9 times more probability of those 30 first movies (see [27]). Besides, each person rates between 10 to 40 movies, uniformly at random. The Figures 4.14 and 4.15 show the residual and error convergence plot of the Newton algorithm for both DS and IS. The convergence result using iterative solver is better for both stagnation and error possibly due to increasing of the condition number of $Hess_Z \hat{J}(0, 0, G)$. Using direct solver, though we do not reach the tolerance of residual, the error is almost equal to the error obtained by iterative solver.

According to the Figure 4.15, we see the quadratic convergence rate as expected using the Newton algorithm.

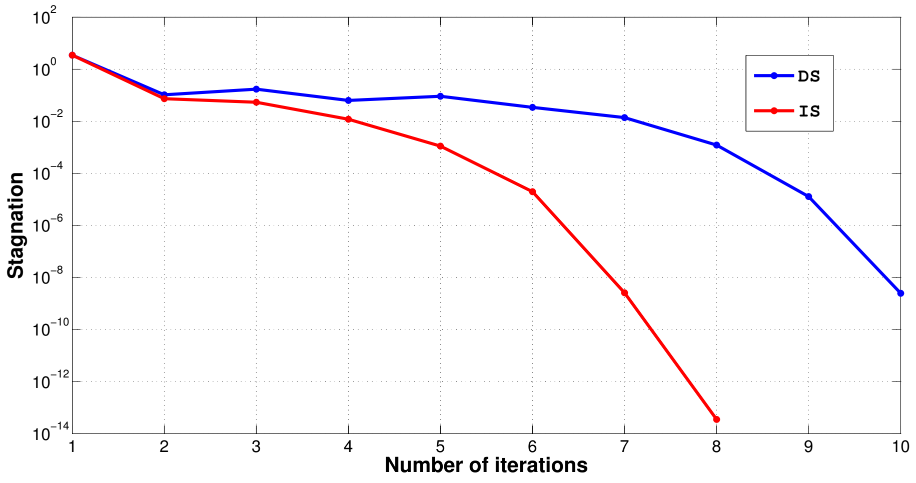

Figure 4.14: Stagnation convergence plot of Newton algorithm using direct solver(DS) and iterative solver PCG(IS)(biased sampling)

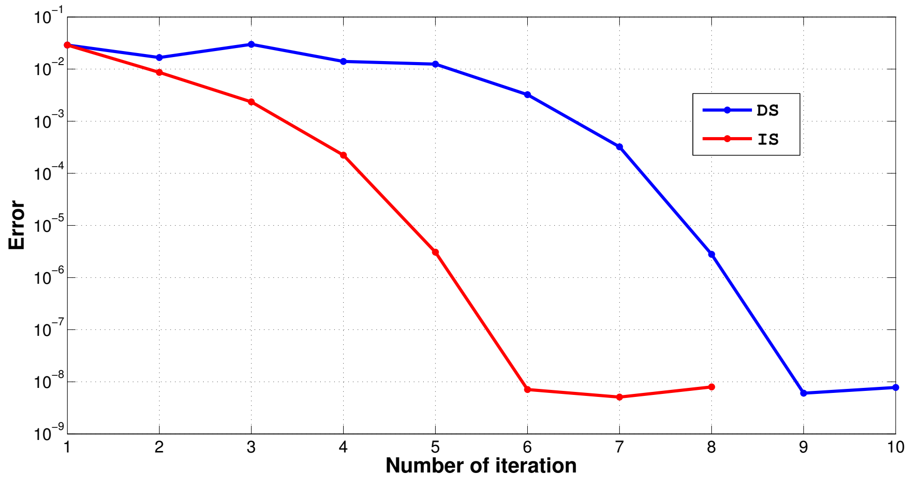

Figure 4.15: Error convergence plot of Newton algorithm using direct solver(DS) and iterative solver PCG(IS)(biased sampling)

Chapter 5

Conclusion

Using the chart based Newton algorithms instead of the steepest descent Riemannian algorithms in the matrix completion problem shows an incredible drop of number of iterations but the computational cost of chart based Newton iteration is very high due to the lack of a low-rank representation of Hessian operator of objective functional. However, the geometric description using local charts and the associated algorithms should be extended to other low-rank tensor formats with applications to high dimensional PDEs, high order tensor completion and other related problems.

Bibliography

[1] P. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds.* Princeton University Press, 2009.

[2] R. L. Adler, J.-P. Dedieu, J. Y. Margulies, M. Martens, and M. Shub, "Newton's method on riemannian manifolds and a geometric model for the human spine," *IMA Journal of Numerical Analysis*, vol. 22, no. 3, pp. 359–390, 2002.

[3] B. Mishra, K. A. Apuroop, and R. Sepulchre, "A riemannian geometry for low-rank matrix completion," *arXiv preprint arXiv:1211.1550*, 2012.

[4] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre, "Fixed-rank matrix factorizations and riemannian low-rank optimization," *Computational Statistics*, vol. 29, no. 3-4, pp. 591–621, 2014.

[5] E. J. Candes, T. Strohmer, and V. Voroninski, "Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming," *Communications on Pure and Applied Mathematics*, vol. 66, no. 8, pp. 1241–1274, 2013.

[6] P.-A. Absil and K. A. Gallivan, "Joint diagonalization on the oblique manifold for independent component analysis," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 5, pp. V–V, IEEE, 2006.

[7] J. Lee, *Introduction to Smooth Manifolds.* Graduate Texts in Mathematics, Springer, 2003.

[8] M. Billaud-Friess, A. Falco, and A. Nouy, "On the principle bundle structure of matrix manifolds." Preprint.

[9] B. Vandereycken, "Low-rank matrix completion by riemannian optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1214–1236, 2013.

[10] P.-A. Absil, L. Amodei, and G. Meyer, "Two newton methods on the manifold of fixed-rank matrices endowed with riemannian quotient geometries," *Computational Statistics*, vol. 29, no. 3-4, pp. 569–590, 2014.

[11] P.-A. Absil, C. G. Baker, and K. A. Gallivan, "Trust-region methods on riemannian manifolds," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 303–330, 2007.

[12] P.-A. Absil and J. Malick, "Projection-like retractions on matrix manifolds," *SIAM Journal on Optimization*, vol. 22, no. 1, pp. 135–158, 2012.

[13] E. Polak, *Optimization: Algorithms and Consistent Approximations.* Applied Mathematical Sciences, Springer, New York, 1997.

[14] J. H. Manton, "A framework for generalising the newton method and other iterative methods from euclidean space to manifolds," *Numerische Mathematik*, vol. 129, no. 1, pp. 91–125, 2015.

[15] J. H. Manton, "Differential calculus, tensor products and the importance of notation," *arXiv preprint arXiv:1208.0197*, 2012.

[16] F. Chinesta, R. Keunings, and A. Leygue, *The Proper Generalized Decomposition for Advanced Numerical Simulations: A Primer (SpringerBriefs in Applied Sciences and Technology).* Springer, 2013.

[17] E. J. Candès and T. Tao, “The power of convex relaxation: Near-optimal matrix completion,” *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[18] J.-F. Cai, E. J. Candès, and Z. Shen, “A singular value thresholding algorithm for matrix completion,” *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[19] D. Goldfarb and S. Ma, “Convergence of fixed-point continuation algorithms for matrix rank minimization,” *Foundations of Computational Mathematics*, vol. 11, no. 2, pp. 183–210, 2011.

[20] K.-C. Toh and S. Yun, “An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems,” *Pacific Journal of Optimization*, vol. 6, no. 615-640, p. 15, 2010.

[21] L. Balzano, R. Nowak, and B. Recht, “Online identification and tracking of subspaces from highly incomplete information,” in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pp. 704–711, IEEE, 2010.

[22] W. Dai, O. Milenkovic, and E. Kerman, “Subspace evolution and transfer (set) for low-rank matrix completion,” *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3120–3132, 2011.

[23] R. H. Keshavan, A. Montanari, and S. Oh, “Matrix completion from a few entries,” *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[24] K. Lee and Y. Bresler, “Admira: Atomic decomposition for minimum rank approximation,” *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4402–4416, 2010.

[25] S. Chatterjee *et al.*, “Matrix estimation by universal singular value thresholding,” *The Annals of Statistics*, vol. 43, no. 1, pp. 177–214, 2015.

[26] R. H. Keshavan and S. Oh, “A gradient descent algorithm on the grassman manifold for matrix completion,” *arXiv preprint arXiv:0910.5260*, 2009.

[27] N. Boumal, *Optimization and estimation on manifolds.* PhD thesis, Université catholique de Louvain, feb 2014.