**NTNU – Trondheim**
Norwegian University of
Science and Technology

# Optimization on Matrix Manifolds with Applications to Blind Source Separation

Vegard Berg Kvernelv

Master of Science in Physics and Mathematics
Submission date: June 2013
Supervisor: Elena Celledoni, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

# Problem Description

Study how concepts from optimization theory generalizes to manifolds, in particular matrix manifolds, and consider how this can be applied to blind source separation problems.

# Summary

The objective of this thesis have been to study optimization techniques on matrix manifolds. In particular the generalization of steepest descent and Newton's method have been taken into consideration. The methods were implemented for two blind source separation techniques, sparse representation and independent component analysis, and compared to established methods. Both techniques assumes a linear relation between recorded signals and source signals. Independent component analysis assumes statistically independent sources, and an equal number of source signals and recorded signals. Sparse representation can assume more sources than recorded signals, but requires the sources to have a sparse nature.

First it was studied how geometrical principles from Euclidean geometry can be generalized to smooth manifolds, and in particular Riemannian manifolds. This provided an understanding of the geometry of matrix manifolds in terms of concepts from familiar Euclidean spaces, such as $\mathbb{R}^n$.

It was seen that for the sparse representation problem, performing Newton's method had lower complexity than what one might expect, but more work is required in order to determine how to ensure convergence. Steepest descent showed promising results, but relatively small step sizes was necessary in order to achieve stable convergence.

For indepdendent component analysis problems, steepest descent and Newton's method outperformed an implementation of the established FastICA method when it comes to accuracy. The Hessian of the objective function had a block-diagonal form, causing the Newton's method to have a relatively low complexity, which makes the method attractable.

# Sammendrag

Målet med denne oppgaven har vært å studere optimeringsmetoder på matrisemangfoldigheter. Mer konkret er generalisering av "steepest descent" og Newtons metode betraktet. Metodene ble implementert for to "blind source separation"-teknikker, glissen signalrepresentasjon og uavhengig komponentanalyse, og ble sammenlignet med etablerte metoder. Begge teknikkene antar en linear sammenheng mellom kildesignalene og de målte signalene. Forskjellen ligger i at uavhengig komponentanalyse antar like mange kilder som målte signaler, samt statistisk uavhengige kilder, mens glissen signalrepresentasjon kan anta flere kilder enn målte signaler, men det antas at kildene har en glissen natur.

Først var det sett på hvordan geometriske prinsipper fra Euklidsk geometri generaliseres til glatte mangfoldigheter, og spesielt Riemannske mangfoldigheter. Dette ga en forståelse for geometrien av matrisemangfoldigheter ved hjelp av forståelsen av konspetene i Euklidske rom, slik som $\mathbb{R}^n$.

Det ble observert at for glissen representasjonsproblemet hadde Newtons metode lavere kompleksitet enn man kan forvente, men mer arbeid er krevd for å kunne si noe sikkert om konvergens. "Steepest descent" viste lovende resultater, men relativt små steg måtte taes for å oppnå stabil konvergens.

For uavhengig komponentanalyse var ytelsen av Newtons metode og "steepest descent" bedre enn en implementering av den etablerte FastICA-metoden, når det kom til nøyaktighet. Hessianen av objektfunksjonen hadde en blokk-diagonal struktur, noe som medfører at Newtons metode har en relativt lav kompleksitet og gjør den attraktiv for slike problemer.

# Preface

This thesis is written as the final part of my Master's degree from the Norwegian University of Science and Technology(NTNU) in Applied Physics and Mathematics, with specialization in Industrial Mathematics. It is written for the course TMA4910-Numerical Mathematics, Master Thesis, in the spring of 2013.

The thesis have been inspired to some extent by the work I did on my specialization project in the Fall of 2012, which was partially a study of Independent Component Analysis.

I would like to thank my supervisor, Elena Celledoni, for helpful discussions, constructive suggestions, proof reading and taking the time to get involved in the content of this thesis.

*Vegard Berg Kvernelv*, June 2013, Trondheim

# Contents

# Chapter 1

# Introduction

The general form of an optimization problem is to minimize (or maximize) a function, the objective function, under given constraints:

$$\min_{x \in \Omega} f(x), \quad \text{such that } c_i(x) = 0, c_j(x) \geq 0, i \in \mathcal{E}, j \in \mathcal{I}$$

The constraints are referred to as the equality and inequality contraints, respectively. $\Omega$ is some field over the real numbers, typically $\mathbb{R}^n$ or $\mathbb{R}^{n \times p}$. Different forms of the objective function or the constraints lead to various subfields of optimization theory, such as linear and quadratic programming, least-squares problems and unconstrained optimization. Within these subfields, many of the algorithms are developed to take advantage of a particular geometry of the problem. The simplex method, for instance, iterates from vertex to vertex in a hyper-polygon. Another subclass of problems is considered in this thesis, namely manifold constraints. The general optimization formulation may in some cases be directly translated to an optimization problem on a manifold, if the manifold is defined in terms of equality and/or inequality contraints. The Stiefel manifold, which will be presented in section 2.2.1, is one such example. In other cases the feasible domain may not be explicitly defined, such as in the case of the Grassmann matrix manifold. In these cases, the domain of the objective function is restricted to the manifold instead:

$$\min_{x \in \mathcal{M}} f(x)$$

This can be viewed as a generalization of unconstrained optimization, but the space does not behave as straightforwardly as say $\mathbb{R}^n$. Hence geometrical concepts popularly used in optimization theory, such as gradients, directional derivaties and Hessians, have to be studied in the more general context of a manifold.

The intuitive advantage of this approach is that it provides a different, and perhaps more natural point of view. Instead of focusing on the purely algebraic formulation of constraints, attention is brought to the geometry of the problem. Manifolds provide an understanding of abstract geometry on the terms of Euclidean

spaces. Concepts such as gradients, Hessians and straight lines can be defined on a manifold (under certain conditions), may can be understood just as in $\mathbb{R}^n$.

Amongst the applications where the geometrical structure stems from physical contraints are robotics and kinetics, where rotation matrices play an important role. A similar application is modelling of the human spine(Adler et al.[4]). Image processing is another popular application, for instance for multi-view images. The perhaps less intuitive manifolds appears in, amongst others, theoretical physics and signal processing. Blind source separation is an example of the latter that will be studied in this thesis.

The outline for the thesis is as follows. Chapter 2 starts with the generalization of some of the familiar geometrical concepts in optimization theory, before discussing how these apply to matrix manifolds in particular. The multi-dimensional Rayleigh quotient is studied as an application. Chapter 3 presents the blind source separation problem and introduces the two techniques independent component analysis and sparse representation. It is shown how a steepest descent method and a Newton method can be applied to the problem, by using the tools from chapter 2. Chapter 4 presents the construction of the numerical tests. Chapter 5 presents and discusses the results.

For this thesis, Lee[21] has been the primary source for the introduction to manifolds and do Carmo[9] for details on Riemannian manifolds. The adaption to matrix manifolds has been covered by Absil et al.[1] and a more practically oriented approach by Edelman et al.[11]. For the applications to blind source separation, papers by Hyvarinen et al.[17, 18, 19] for the background of independent component analysis, and the paper by Kreutz-Delgado et al.[20] for background on sparse representation. The book by Comon et al.[7] has also been useful for details on this topic.

# Chapter 2

# Matrix Manifold Optimization

## 2.1 Manifolds

The reason manifolds are of interest in optimization theory is their applicability to abstract geometry. Their ability to provide a geometrical understanding of higher dimensionsional surfaces, curves, volumes and more is the key to better understand how to develop smart algorithms, not only within optimization theory, but to other problems with a complicated structure as well.

A rigorous introduction to manifolds will not be given here since a wide range of literature exists on the topic. The most central elements from an optimizational point of view will however be repeated here to emphasize which geometrical concepts are of particular interest, and to introduce the terminology used throughout this thesis. It will always be assumed that the manifolds considered are sufficiently smooth.

The tangent space of a manifold, $\mathcal{M}$, at a point $x$ is denoted $T_x\mathcal{M}$ (for a definition see e.g. [21]). The collection of all points and their respetive tangent spaces is called the tangent bundle and is denoted by $T\mathcal{M}$. Let $f$ be a real-valued function on $\mathcal{M}$. The generalization of the directional derivative at $x \in \mathcal{M}$ in direction $z \in T_x\mathcal{M}$ is a real-valued function:

$$\mathrm{D}_z f(x) = \frac{\mathrm{d}f(\gamma(t))}{\mathrm{d}t}\Big|_{t=0},$$

where $\gamma$ is a curve on $\mathcal{M}$, satisfying $\gamma(0) = x, \dot{\gamma}(0) = z$.

A Riemannian metric is a collection of inner products, $\{\langle\cdot,\cdot\rangle_x\}_{x\in\mathcal{M}}$, such that $\langle\cdot,\cdot\rangle_x$ defines an inner product on $T_x\mathcal{M}$. In general, an inner product is an additional structure to a vector space. Since a manifold is not automatically a vector

space, the inner product is defined point-wise on the tangent spaces of $\mathcal{M}$ instead. A manifold equipped with a Riemannian metric is a Riemannian manifold, and the field of Riemannian manifolds is referred to as Riemannian geometry.

With the notion of an inner product, the normal space of a manifold at $x$ is defined as the set of vectors orthogonal to $T_x\mathcal{M}$. It is denoted $(T_x\mathcal{M})^\perp$. For a submanifold $\mathcal{M}$ of $\mathbb{R}^n$, a tangent vector of $\mathbb{R}^n$ (which is a vector in $\mathbb{R}^n$), can be decomposed into its component on $T_x\mathcal{M}$ and $(T_x\mathcal{M})^\perp$. Accordingly, orthogonal projection onto the tangent space is defined as taking $\xi \in \mathbb{R}^n$ to its component on the tangent space. Projection onto the normal space is defined through the relation $\mathrm{P}_x\xi + \mathrm{P}_x^\perp\xi = \xi$.

The tools to define the notion of a gradient are now in place. For a real-valued function, $f : \mathbb{R}^n \to \mathbb{R}$, the gradient satisfies $\mathrm{D}_z f(x) = (\text{grad } f(x))^T z$, where D denotes the directional derivative, here taken with respect to the vector $z$. On a Riemannian manifold, the gradient at the point $x$ is analogously defined as the (unique) tangent vector satisfying

$$\langle \text{grad } f(x), z \rangle = \mathrm{D}_z f(x) \quad \forall z \in T_x\mathcal{M},$$

where $\langle \cdot, \cdot \rangle$ denotes the Riemannian metric at $x$. Note that the gradient will depend on the choice of metric. Also observe that, just as in $\mathbb{R}^n$, grad $f(x) = 0 \iff \mathrm{D}_z f(x) = 0 \; \forall \; z \in T_x\mathcal{M} \iff x$ is a stationary point of $f$.

For a submanifold, $\mathcal{M}$, of $\mathbb{R}^n$, the gradient at $x$ is the orthogonal projection of the gradient from $\mathbb{R}^n$ onto $T_x\mathcal{M}$ ([1], section 3.6.1).

A method that utilizes all of the tools mentioned above is the method of steepest descent. In $\mathbb{R}^n$ it is based on the relatively simple concept of following the direction in which the objective function decreases most rapidly. The method is iterative, and in its conceptually simplest form it attempts to find the minimizer of the function in the direction of steepest descent: $\min_\alpha f(x_k - \alpha \text{ grad } f(x_k))$, and use this as the next iterate. On a manifold the principle is the same, but since addition is not well-defined on an arbitrary manifold, moving to the next iterate (or evaluating the objective function along a direction for that matter) will have to be redefined.

### 2.1.1 Geodesics

Iterative methods in $\mathbb{R}^n$ are typically moving from one point to another at every iteration (or adding a vector to a point to arrive at another point). On a manifold in general, this notion of "moving from one point to another" is not as trivial. The concept of geodesics is therefore introduced. A geodesic curve, $\gamma : (a, b) \to M$, is a generalization of a straight line in $\mathbb{R}^n$ in the sense that it is the smooth curve that minimizes the distance between two points, $x$ and $y$, defined by $\int_a^b \|\dot{\gamma}(t)\|^{1/2} \mathrm{d}t$, $\gamma(a) = x, \gamma(b) = y$. $\|\cdot\|$ is the norm induced by the Riemannian metric. Other equivalent definitions exist as well, for instance as zero-acceleration curves.

An extensive study of geodesics for matrix manifolds in particular may be found in the article by Edelman, Arias and Smith[11]. When it comes to iterative methods, the idea is to use geodesics to move on the manifold. Given a current iterate, $x_k$, the next iterate is found by first picking a direction in which it is reasonable to move, then move along the geodesic in that particular direction until a reasonable next iterate is found. In the case of minimizing an objective function, a reasonable direction may be the negative gradient direction (that is, the steepest descent direction), and a reasonable estimate may be the minimizer along that direction. This makes sense because following the geodesic is the shortest path, in the Riemannian metric, to any of the points chosen as the successive iterate. As will be illustrated later, determining a geodesic may be computationally infeasible. The notion of a retraction is therefore introduced to provide an alternative that approximates the behavior of the geodesics, but may be computationally less expensive. It also provides a more general analytical framework than geodesics alone.

### 2.1.2 Retractions

A retraction can be understood as an approximation of a geodesic. In fact, if $\mathcal{M}$ is a submanifold of a Euclidean space, it can be shown that the distance between the geodesic and an arbitrary retraction is $\mathcal{O}(t^2)$ (see reference [2]). If geodesics are tedious to compute, there may exist retractions of lower complexity, at the cost of accuracy. A retraction is a smooth mapping, $R : T\mathcal{M} \to \mathcal{M}$, such that when restricted to $T_x\mathcal{M}$, it satisfies

i) $R_x(0_x) = x$, where $0_x$ denotes the origin of $T_x\mathcal{M}$

ii) For every tangent vector $z \in T_x\mathcal{M}$, the curve $\gamma_z : t \mapsto R_x(tz)$ satisfies $\dot{\gamma}(0) = z$.

It is assumed that the domain of $R_x$ is the whole $T_x\mathcal{M}$, but the general definition is that i) and ii) hold on an open ball containing $0_x$. One of the earliest definitions of the term can be found in the article by Shub from 1986[24]. If moreover

iii) $\left[\frac{\mathrm{D}}{\mathrm{d}t}\dot{R}(t\xi)\right]_{t=0} = 0, \quad \forall \xi \in T_x\mathcal{M},$

the retraction is termed a second-order retraction[1].

The exponential map (which generally should not be interchanged with the matrix exponential function) is a particular case of a second-order retraction. For every $\xi \in T_x\mathcal{M}$, there is a unique geodesic satisfying $\gamma(0) = x$ and $\dot{\gamma}(0) = \xi$ (see e.g. do Carmo[9], chapter 3.2). The exponential map is defined as evaluating this geodesic at $t = 1$. It may be observed that this is a second-order retraction.

Retractions and geodesics provide a natural way of linearizing functions defined on manifolds. If $f$ is a real-valued function on $\mathcal{M}$ and $R$ a retraction, then its extension $\hat{f} = f \circ R$ at a point $x$ is a real-valued function on the Euclidean space

$T_x\mathcal{M}$. In particular, from the definition of a retraction, one has that

$$\hat{f}(0_x) = f(x)$$
$$\text{grad } \hat{f}(0_x) = \text{grad } f(x),$$

As mentioned, one of the most important reasons for introducing retractions is to have a framework for performing analysis that is wider than requiring geodesics. Loosely speaking, any numerical method will only be as accurate as its worst approximation. Hence, it makes sense to have a way of allowing less accurate, but more efficient update schemes if the rest of the method relies on rough estimates.

#### Steepest Descent on Manifolds

Finally, the steepest descent method on manifolds is presented. It was argued how generalizing from $\mathbb{R}^n$ to manifolds was not straight forward since addition is not well-defined. By using geodesics (or retractions) one may translate the algorithm to

1. $\alpha^* = \arg\min_{\alpha} f\left(R_{x_k}(-\alpha \text{ grad } f(x_k))\right)$
2. $x_{k+1} = R_{x_k}(-\alpha^* \text{ grad } f(x_k))$

Other versions of steepest descent exist as well. $\alpha^*$ may for instance be chosen according to some Armijo-Wolfe-like conditions instead, in case step 1 is hard to solve. The algortihm provides an important practical implication, which is that most algorithms will consider a minimization problem on the tangent space at $x_k$, find an appropriate next iterate, then retract the solution to keep it on the manifold.

The manifolds considered in depth in this thesis are either $\mathbb{R}^{n\times p}$ or submanifolds of $\mathbb{R}^{n\times p}$. The next section is an introduction to matrix manifolds, and how the concepts introduced here specialize.

## 2.2 Matrix Manifolds

The set of real $n\times p$ matrices is a manifold. This means that the concepts introduced in the previous section may be applied when studying matrices and in particular optimization techniques on matrices. In addition, for submanifolds $\mathbb{R}^{n\times p}$, many of the concepts are naturally inherited from $\mathbb{R}^{n\times p}$. The projection of the gradient, as mentioned in section 2.1, is one such example.

In the introduction it was mentioned how matrices in various ways can describe the geometrical properties of the feasible domain. Some examples are given here to get a clearer idea of what that means. The matrix manifolds are fairly well-known, so the proofs that they are in fact manifolds are not included.

**Example 1.** *The (orthogonal)* ***Stiefel manifold****,* $\mathrm{St}_n(p)$*, is the set of* $n \times p$ *matrices with orthonormal columns, that is,*

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

*Special cases are* $p = 1$ *and* $p = n$ *which are, respectively, the hypersphere,* $S^{n-1}$*, and the Lie group of orthogonal matrices,* $O(n)$*.*

**Example 2.** *The* ***non-compact Stiefel manifold****,* $\mathbb{R}_*^{n\times p}$*, is the set of* $n \times p$ *matrices with linearly independent columns. The special case* $n = p$ *is the general linear group,* $GL_n$*, which is the group of invertible matrices.*

**Example 3.** *The* ***Grassmann manifold****,* $\mathrm{Grass}_n(p)$*, is the set of all* $p$*-dimensional subspaces of* $\mathbb{R}^N$*. For instance if* $f : \mathrm{St}_n(p) \to \mathbb{R}$*, and in addition* $f(X) = f(XQ)$ *for any orthogonal matrix* $Q$*, the optimization problem may be interpreted as a problem on the Grassmann manifold.*

**Example 4.** *The* ***3D Rotation group****, SO(3), is the set of* $3 \times 3$ *orthogonal matrices of determinant 1. It represents rotation in* $\mathbb{R}^3$*, preserving length and orientation.*

**Example 5.** *The* ***fixed rank manifold****,* $\mathcal{M}_r(n \times p)$*, is the set of* $n \times p$ *matrices of rank* $r$*. Its dimension is* $(n + p - r)r$ *[3]*

The common denominator of all of the above examples, which is also one of the primary reasons for considering matrices in the manifold context, is that the constraints they impose are naturally presented using matrices instead of a set of equality constraints. Let for instance $W(t)$ be a curve on $\mathrm{St}_n(p)$. Differentiate the constraint with respect to $t$:

$$\frac{\mathrm{d}}{\mathrm{d}t}[W(t)^T W(t)]\Big|_{t=0} = W_0^T \dot{W}(0) + \dot{W}^T(0) W_0 = \frac{\mathrm{d}}{\mathrm{d}t}(I)\Big|_{t=0} = 0,$$

hence the velocity of the curve through $W_0$ must satisfy $W_0^T \dot{W}(0) \to$ skew-symmetric. In other words, the tangent space of the Stiefel manifold is the set $T_Z \mathrm{St}_n(p) = \{Z \in \mathbb{R}^{n\times p} : X^T Z = -Z^T X\}$. The normal space is the set of matrices that can be written as $N = X\Omega$, for a symmetric matrix $\Omega$. The projection of a matrix $A \in \mathbb{R}^{n\times p}$ onto the tangent space is $\mathrm{P}_X A = X\mathrm{skew}\,(X^T A) + (I - XX^T)A$. Details may be found in the article by Edelman et al. [11]. The gradient of a function, $f$, is readily found, as shown in section 2.1, by projection from the embedding Euclidean space:

$$\mathrm{grad}\, f(X) = \mathrm{P}_X \nabla f(X) = X\mathrm{skew}\,(X^T \nabla f(X)) + (I - XX^T)\nabla f(X).$$

Note that a different metric yields a different projection operator.

### 2.2.1 Advantages of retractions

A simple differential equation may illustrate more explicitly the reason to study matrices as manifolds. Let $X = X(t)$ be an $n \times p$ matrix and consider

$$\dot{X} = AX, \quad A = -A^T, \quad X(0) = X_0.$$

The solution to this equation is $X(t) = \exp[At]X_0$ (seen by e.g. insertion), where exp is the regular exponential function for square matrices. The following holds for the solution,

$$X^T X = X_0^T \exp[A^T t] \exp[At] X_0 = X_0^T \exp[-At] \exp[At] X_0 = X_0^T X_0,$$

thus every element of $X^T X$ is invariant in time. Solving this numerically though, does not guarantee that this property is preserved. Implemented with a forward Euler scheme of step size $h$, one may observe that

$$X_{k+1}^T X_{k+1} = X_k^k (I - hA)(I + hA) X_k = X_k^T (I - h^2 A^2) X_k.$$

Hence, an error depending on $A$ and $h$ is introduced in the supposedly invariant quantity for each time step. For this problem, and others, it may be of greater importance to have a feasible solution, than an accurate solution. Consider for instance the hypothetical optimization problem

$$\min_{Q \in O(n)} f(Qz),$$

where $z$ is some known $n$-dimensional vector and $f$ is a real-valued function.The objective function assumes that regardless of the argument matrix, $Q$, the 2-norm of $Qz$ is preserved[1]. If a matrix violates the constraints, the model that was assumed to formulate the optimization problem could become invalid.

The two examples above hopefully illustrate why it is important to have an analytice framework to evaluate the cost of preserving geometrical structure before accuracy.

### Geodesic versus Retraction

As mentioned in the previous section, the computational complexity of geodesics may be too high and alternative maps from the tangent bundle to the manifold are considered instead. A particular example can be found for geodesics on $\mathrm{St}_n(p)$ with the canonical metric, which are given by

$$\gamma(t) = [\gamma(0), Q] \exp\left( t \begin{bmatrix} \gamma(0)^T \dot{\gamma}(0) & -R^T \\ R & 0 \end{bmatrix} \right) \begin{bmatrix} I \\ 0 \end{bmatrix},$$

where $Q$ and $R$ are determined by the compact qr-decomposition of $(I - \gamma(0)\gamma(0)^T)\dot{\gamma}(0)$. The proof may be found in REF:edelman. In addition to a qr-decomposition, a computation of the matrix exponential is needed. However, the complexity is at worst of the same order as qr-decomposition which is $\mathcal{O}(np^2)$ (see for instance the article by Moler and Van Loan[22] for details on efficient exponential computations).

[1] It is a property of the orthogonal group that $\|Qz\|_2 = \|z\|_2$ for arbitrary $Q \in O(n)$.

An alternative is to use a retraction based on the Cayley transformation, studied in the article by Wen[25]. Consider the curve on $\mathrm{St}_n(p)$,

$$Y(t) = \left(I + \frac{t}{2}B\right)^{-1}\left(I - \frac{t}{2}B\right)W, \tag{2.1}$$

for some $W \in \mathrm{St}_n(p)$ and $B$ any skew-symmetric matrix. The curve satisfies $Y(0) = W$, $\dot{Y}(0) = -BW$ (see mentioned article for the proofs). If for instance $B$ can be written as $XG^T - GX^T$ for some $G \in \mathbb{R}^{n\times p}$, then $\dot{Y}(0)$ is the projection of $G$ onto the tangent space at $X$, using the canonical metric[2]. Moreover, if $G = \nabla f(X)$ (where $\nabla f(X)_{ij} = \partial f/partial X_{ij}$), then $Y(t)$ is a curve along the steepest descent direction. Furthermore, by writing $U = [\nabla f_W, W]$ and $V = [W, -\nabla f_W]$ and utilizing the Sherman-Morrison-Woodbury formula(see e.g. Demmel[8], question 2.13) for inverting matrices, the curve is rewritten as

$$Y(t) = W - tU\left(I + \frac{t}{2}V^TU\right)^{-1}V^TW,$$

which only requires the inversion of a $2p\times 2p$-matrix ($\mathcal{O}(p^3)$ flops), as opposed to the $\mathcal{O}(np^2)$-flops required for the qr-decomposition mentioned above. This illustrates how the use of a retraction over the geodesic can be beneficial.

## 2.3 Second-order Geometry and Newton's Method

A well-known method of optimization theory is Newton's method for finding zeroes of a vector field. An important special case is when the vector field is the gradient of some real-valued function, which corresponds to finding the function's stationary points.

In $\mathbb{R}^n$, Newton's method may be derived from multi-variable Taylor expansion. Let $F = [f_1(x), f_2(x)...f_p(x)]^T$ be a vector field on $\mathbb{R}^n$. Approximate the vector field around the current iterate by,

$$F(x_k + h) \approx F(x_k) + J(x_k)h,$$

where $J$ is the Jacobian of $F$, i.e., $J_{ij} = \partial f_i/\partial x_j$. The successive iterate, $x_{k+1}$, is found by setting the approximation (i.e. the left hand side) to zero and solve for $h$. This yields the iteration scheme $x_{k+1} = x_k + h = x_k - J^{-1}(x_k)x_k$. Note that in the special case $F = \mathrm{grad}\, g(x)$, $J$ becomes the Hessian matrix of $g$.

There are several ways to interpret the Euclidean Hessian matrix. By definition, it is the collection of the second-order partial derivatives of $g$ in matrix format. Alternatively, it can be viewed as a linear operator that takes a vector $z$ to the directional derivative of all the components of the gradient in direction $z$. It also

[2]The definition of the canonical metric is $\langle Z_1, Z_2\rangle_{\mathrm{canon}} = \mathrm{tr}(Z_1(I - \frac{1}{2}XX^T)Z_2)$, and the induced projection is $\mathrm{P}_W Z = Z - WZ^TW$. See the article by Edelman et al.[11] for details.

satisfies being the quadratic form defined by $\frac{\mathrm{d}^2}{\mathrm{d}t^2}\Big|_{t=0} g(x+tz)$, that is the concavity of $g$ at $x$, in direction $z$[11].

### 2.3.1 The Hessian operator

For manifolds, the Hessian is considered as a linear operator. The intuitive purpose of the Hessian is to provide information on how the gradient at a point behaves in tangent directions to the manifold. In standard literature on differential geometry, the Hessian is usually not covered in great depth since its use is argueably most seen within optimization theory. The two most used definitions are as a linear map from the tangent space to itself, or as a bilinear real-valued operator defined by a quadratic form similar to the one mentioned above. The view that is taken throughout this thesis is the same as in Absil et al.[1]. In a Euclidean space, the Hessian is defined as the mapping Hess $f(x) : T_x\mathcal{M} \to T_x\mathcal{M}$, such that

$$\text{Hess } f(x)[z] = \mathrm{D}_z \text{grad } f(x) = \lim_{t\to 0} \frac{\text{grad } f(x+t\eta_x) - \text{grad } f(x)}{t},$$

The gradient of a submanifold of a Euclidean space is found by projection onto the tangent space. Let $\mathcal{E}$ denote the ambient Euclidean space, and $\mathcal{M}$ the manifold. Moreover, denote the projection operator $\mathrm{P}_x : \mathcal{E} \to T_x\mathcal{M}$. The Hessian of a submanifold of a Euclidean space is defined similar to the gradient as

$$\text{Hess } f(x)[z] = \mathrm{P}_x \left(\mathrm{D}_z(\mathrm{P}_x \text{grad } f(x))\right),$$

where it is emphasized that $\mathrm{D}_z$ is the directional derivative in the Euclidean space, applied to the projected gradient[3]. For submanifolds of $\mathbb{R}^{n\times p}$, the projection operator is an $np \times np$ matrix. It is then clear that the Hessian of matrix submanifold follows from the product rule,

$$\begin{aligned} \text{Hess } g(x)[z] &= \mathrm{P}_x \left[\mathrm{D}_z(\mathrm{P}_x \nabla g(x))\right] \\ &= \mathrm{P}_x \left[\mathrm{P}_x(\mathrm{D}_z \nabla g(x)) + (\mathrm{D}_z P_x)\nabla g(x)\right] \\ &= \mathrm{P}_x \partial^2 g(x)[z] + \mathrm{P}_x(\mathrm{D}_z \mathrm{P}_x)\nabla g(x)), \end{aligned} \tag{2.2}$$

where the parentheses in the last expression are used to emphasize the differentiation of the projection operator. The operator $\mathrm{D}_z$ is the directional derivative in $\mathbb{R}^{n\times p}$. The last term is in the technical report by Absil et al.[3] shown to be related to the Weingarten map, this will not be studied further here. $\partial^2 g(x)[z]$ is the Hessian in $\mathbb{R}^{n\times p}$ (the Hessian matrix multiplied by the vector $z$). This expression is of often practical when operating on matrix manifolds. It requires the computation of the Euclidean Hessian, gradient, projection operator and differentiated projection operator. It is evident by the expression that obtaining the Hessian requires taking into account how the projection operator changes on the manifold, in addition to straight forward projecting the Euclidean Hessian.

[3]The expression comes from the definition of the Hessian through affine connections(see e.g. do Carmo [9]) which are not discussed here. The definition is justified by proposition 5.3.2[1].

For completeness, an alternative definition of the Hessian is included. Edelman et al.[11] define the Hessian by the quadratic form

$$\text{Hess } g(x)[z,z] = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\Big|_{t=0} g(\gamma(t)), \quad \gamma(0) = x, \dot{\gamma}(0) = z,$$

where $\gamma(t)$ is a geodesic. The Hessian operator is determined by the polarization identity, which given any quadratic form provides a bilinear form associated with it[4]. This expression does however require knowledge of the geodesic.

### 2.3.2 Local approximations on the manifold

By using retractions, it is possible to introduce a model approximation for the objective function on the manifold. Consider the function $\hat{f}_x = f \circ R_x : T_x\mathcal{M} \to \mathbb{R}$. Since it is defined on a Euclidean space, its Taylor series is defined. The second-order approximation is

$$\begin{aligned}
\hat{m}_x(z) &:= \hat{f}_x(0_x) + \langle \text{grad } \hat{f}_x(0_x), z\rangle + \frac{1}{2}\langle z, \text{Hess } \hat{f}_x(0_x)[z]\rangle \\
&= f(x) + \langle \text{grad } f_x(x), z\rangle + \frac{1}{2}\langle z, \text{Hess } \hat{f}_x(0_x)[z]\rangle \\
&= f(x) + \langle \text{grad } f(x), z\rangle + \frac{1}{2}\langle z, \text{Hess } f(x)[z]\rangle,
\end{aligned}$$

where the first equality holds by the properties of a retraction. The last equality is only true for second-order retractions(proposition 5.5.5[1]).

The idea was to have a model on the manifold that approximates the objective function. If $x$ is the original point, the approximation is achieved by taking another point, $y \in \mathcal{M}$, through the inverse of the retraction to the tangent space at $x$. On the tangent space a Taylor expansion is known and it thus holds that

$$|\hat{f}_x(R_x^{-1}(y)) - \hat{m}_x(R_x^{-1}(y))| \leq C\|R_x^{-1}(y)\|^{r+1}, \tag{2.3}$$

with $r$ equals 1 or 2, depending on whether $R$ is a first or second order retraction. However, by definition, $\hat{f}_x = f \circ R_x$ such that $\hat{f} \circ R_x^{-1} = f$, and $\hat{m}_x \circ R_x^{-1}$ is an $r$-order approximation of $f$, that is defined on the manifold. It is also an approximation that can be expressed using the function value, the gradient and the Hessian of $f$ at $x$. In particular, if $y$ comes from retract a tangent vector, $z$, then the error is given by the norm of the tangent vector,

$$\begin{aligned}
|f(R_x(z)) - \hat{m}_x(z))| &= |f(R_x(z)) - (f(x) + \langle \text{grad } f(x), z\rangle + \frac{1}{2}\langle z, \text{Hess } f(x)[z]\rangle)| \\
&\leq C\|z\|^{r+1}
\end{aligned}$$

Furthermore, proposition 7.1.3[1] states that this is equivalent to $|f(y) - \hat{m}_x(R_x^{-1}(y))| \leq C^*(\text{dist}(x,y))^{r+1}$. In other words, $\hat{m}_x \circ R_x^{-1}$ defines an approximation of $f$ on the

[4]For a general quadratic form $Q(v)$, the bilinear form is $B(u,v) = \frac{1}{4}(Q(u+v) - Q(u-v))$

manifold itself, and its error is bounded by the distance measured on the manifold.

The message of this section is to show that by using retractions, it is possible to describe something that on a manifold is analogous to Taylor expansion in Euclidean space, and to justify why using retractions is a proper alternative to geodesics.

### 2.3.3 Newton's method on manifolds

Just as in $\mathbb{R}^n$, Newton's method on a manifold seeks to find a tangent vector, $\eta$, such that Hess $f(x)[\eta] = -\text{grad } f(x)$. Updating the current iterate with this solution presents the same challange as with the steepest descent method. A retraction is therefore used to map the Newton vector to the manifold. The algorithm is presented as algorithm 1.

**Data**: Retraction $R$, real-valued (objective) function, $f$
**Result**: $x^*$, such that grad $f(x^*) = 0$
**Input**: Initial guess $x_0$, Hess $f(\cdot)[\cdot]$, grad $f(\cdot)$
**while** *not converged* **do**
  Solve for $\eta_k$:
  $$\text{Hess } f(x_k)[\eta_k] = -\text{grad } f(x_k)$$
  Update iterate:
  $$x_{k+1} = R_{x_k}(\eta_k)$$
  Compute convergence criterion
**end**

**Algorithm 1:** Newton's method for real-valued functions on manifolds.

The previous section provides an important way of understanding Newton's method. The Newton vector is the solution to an approximate optimization problem on the tangent space, where the error in the approximation is given by equation (2.3). In particular, it is noteworthy that for a Newton vector with large norm, the error grows cubically. On the other hand, it becomes a very good approximation close to a true stationary point (which is analogous to Newton's method in $\mathbb{R}^n$). It is emphasized again that the bound is after the retraction has been taken into account, and not the truncation error from the Taylor expansion on the tangent space.

Due to this understanding of the Newton's method, it becomes clear that it may be necessary to consider for instance trust-region methods, or an initial gradient search to ensure that the model equation sufficiently approximates the objective function.

### 2.3.4 The multi-dimensional Rayleigh quotient

To verify the necessity of projecting the Hessian correctly, and to test the quadratic convergence of Newton's method, the generalized Rayleigh quotient is considered. For a square matrix, $A$, the multi-dimensional Rayleigh quotient is defined as

$$\rho(X) = \mathrm{tr}(X^T A X), \quad X \in \mathrm{St}_n(p).$$

It holds that $X$ is a stationary point if and only if the columns of $X$ are eigenvectors of $A$ (Absil[1], section 4.8.1). To test convergence rate, a symmetric positive definite matrix is generated by defining $D = \mathrm{diag}(1, 1.01, \ldots, 1.99, 2)$. The spacing was chosen such that the method requried several steps before it converged up to machine precision. The next section discusses in depth how the spacing affect convergence rate. Next, an orthogonal matrix, $Q$, is computed from the qr-decomposition of a random $100 \times 100$ matrix. Defining $A = QDQ^T$ thus yields a summetric positive definite matrix whose eigenvalues are known. To achieve convergence for Newton's method, it is necessary to be close enough to the minimum. This is ensured by taking sufficiently many steps with a steepest descent method.

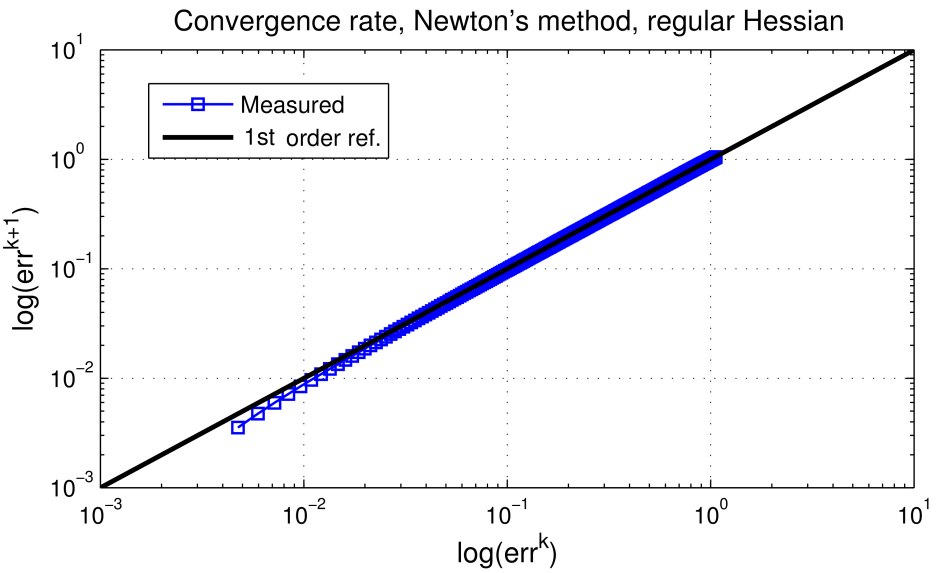


Figure 2.1: Convergence rate for the Euclidean Hessian, $\partial^2\rho(X)$. $\sim$350 iterations performed. Only first order convergence is observed

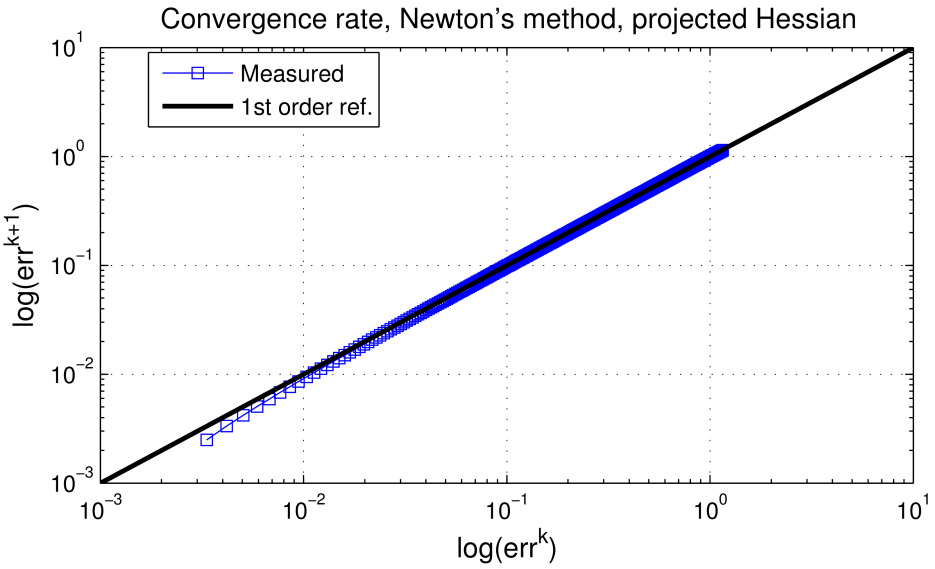


Figure 2.2: Convergence rate for the projected Hessian, $\mathrm{P}_X \partial^2 \rho(X)$. $\sim$350 iterations. Same as the plain Hessian, only first order convergence.

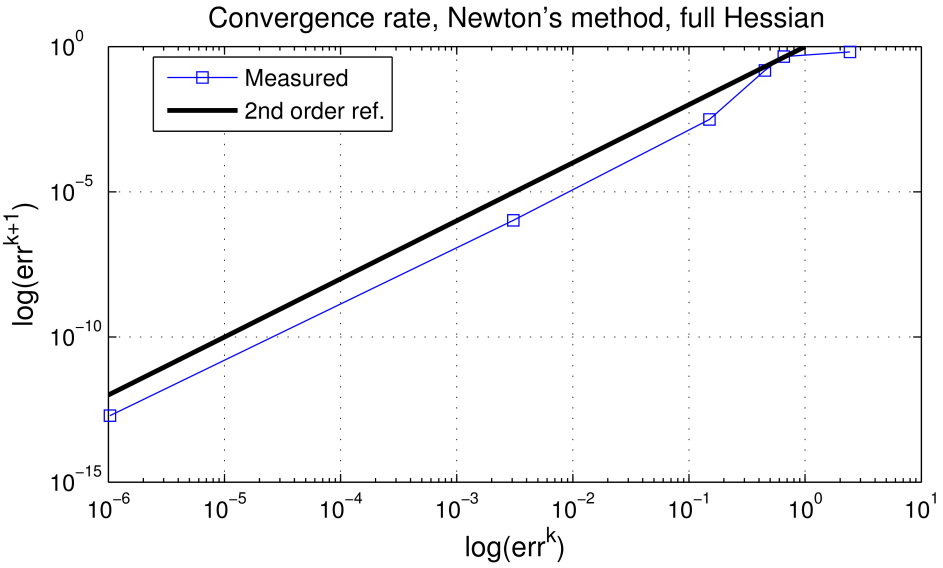


Figure 2.3: Convergence rate for the full Hessian, $\mathrm{P}_X \partial^2 \rho(X) + \mathrm{P}_X D_Z \mathrm{P} \nabla \rho(X)$. Less than 10 iterations to machine precision accuracy.

Three ways to compute the Hessian operator have been considered. Without

taking into account the manifold structure, one may utilize the Euclidean Hessian, which is $AZ$, and project the solution after the Newton vector is found. Alternatively, one may project the Hessian onto the tangent space using the projection operator (see below). The final option is using equation (2.2), which comes from considering the Riemannian manifold structure.

The projection operator, $\mathrm{P}_x : \mathbb{R}^{n\times p} \to T_x\mathrm{St}_n(p)$, applied to a matrix $B \in \mathbb{R}^{n\times p}$ is

$$\mathrm{P}_X B = X\mathrm{skew}\,(X^T B) + (I - XX^T)B,$$

see for instance Edelman et al.[11]. Alternatively write out skew, collect the terms and rewrite as

$$\mathrm{P}_X B = B - X\mathrm{sym}\,(X^T B). \tag{2.4}$$

Taking the directional derivative thus yields

$$\begin{aligned}(\mathrm{D}_Z \mathrm{P}_X B) &= \lim_{t\to 0} \frac{\mathrm{P}_{X+tZ}B - \mathrm{P}_X B}{t} \\ &= Z\mathrm{sym}\,(X^T B) + X\mathrm{sym}\,(Z^T B).\end{aligned}$$

Moreover, projecting the directional derivative of the projection becomes

$$\begin{aligned}\mathrm{P}_X(\mathrm{D}_Z \mathrm{P}_X B) =& Z\mathrm{sym}\,(X^T B) + X\mathrm{sym}\,(Z^T B) \\ &- X\mathrm{sym}\,(X^T Z\mathrm{sym}\,(X^T B)) - X\mathrm{sym}\,(X^T B)\end{aligned} \tag{2.5}$$

Using $\nabla\rho(X) = AX$ and $\mathrm{Hess}\,\rho(X)[Z] = AZ$, equation (2.2) becomes

$$\mathrm{Hess}\,\rho(X)[Z] = AZ - X\mathrm{sym}\,(X^T AZ) - Z(X^T AX) + X\mathrm{sym}\,(X^T Z(X^T AX)).$$

The convergence rate is determined by running the three versions of Newton's method from the same initial point until a stationary point is reached, then use this as the reference when running the same computations again. The error is computed for each iteration and plotted with logarithmic scales as seen in figures 2.1, 2.3 and 2.3. It is clear that it is necessary to use the Riemannian manifold structure to achieve quadratic convergence and that only a few iterations were needed in order to achieve machine precision. The methods were implemented by explicitly forming the Hessian matrix and solve the system directly using Matlab.

## 2.4 Convergence Analysis

The analytical tools that comes with the notion of retractions are as mentioned important. This section discusses some of the most general results from Absil et al.[1]. The purpose is to present some of the results available, not to develop new convergence proofs. A few of the theorems are therefore included without the proof. The main theorem of this section is quite technical, so an attempt on specializing it to a steepest descent method is made. This is primarily to address what factors are of significance when considering optimization methods on manifolds.

### 2.4.1 Convergence to stationary points

In $\mathbb{R}^n$, steepest descent is a particular case of a line-search method, that is $x_{k+1} = x_k + \alpha_k \eta_k$ with $\eta_k = \nabla f(x_k)$ and a suitable scalar $\alpha_k$. Convergence to a stationary point is achieved if the Wolfe conditions are satisfied and the gradient is sufficiently smooth (see e.g. Nocedal and Wright[23]). As addition is not generally well-defined on a manifold, the manifold will be equipped with a retraction. The line-search method is generalized to the form

$$x_{k+1} = R_{x_k}(\alpha_k \eta_k). \tag{2.6}$$

It can be shown that convergence to a stationary point is achieved under conditions that resemble the $\mathbb{R}^n$ requirements. The proof is rather tedious, but due to the importance of the theorem, it will be included without the proof. First, a few definitions are required. A gradient-related sequence is defined as a collection of tangent vectors $\{\eta_k\}$, where $\eta_k \in T_{x_k}\mathcal{M}$, such that the following holds:

$$\lim_{k\to\infty} \sup_{k\in\mathcal{K}} \langle \text{grad } f(x_k), \eta_k \rangle < 0, \quad \eta_k \text{ bounded}, \tag{2.7}$$

for any subsequence, $\{x_j\}_{j\in\mathcal{K}}$, of $\{x_k\}$ that converges to a non-stationary point of $f$. Note that the argument of (2.7) is the directional derivative of $f$ at $x_k$ in direction $\eta_k$, thus an interpretation is that the directions, $\eta_k$, should always be descent directions, unless $\{x_j\}_{j\in\mathcal{K}}$ has converged.

Absil et al.[1] shows convergence for a more general method where $x_{k+1}$ need not be determined by a retraction. Instead, the function value at the next iterate should decrease "sufficiently". This is similar to one of the Wolfe conditions in $\mathbb{R}^n$ stating that $x_{k+1}$ should satisfy

$$f(x_{k+1}) \leq f(x_k) + c\nabla f(x_k)^T p_k,$$

that is, the next iterate should ensure that the function value is not only less than (or equal to) the previous iterate, but should also take into account the directional derivative in the search direction. To generalize this further, the Armijo point is defined. Let $x \in \mathcal{M}$ and $\eta \in T_x\mathcal{M}$ and let $\alpha > 0, \beta, \sigma \in (0,1)$. The Armijo point is $\eta^A = t^A\eta = \beta^m \alpha\eta$, where $m$ is determined as the smallest (non-negative) integer satisfying

$$f(x) - f(R_x(\beta^m \alpha\eta)) \geq -\sigma\langle \text{grad } f(x), \beta^m \alpha\eta\rangle. \tag{2.8}$$

$t^A$ is the Armijo step size. Note that the Armijo point is a general definition, not defined on the terms of any algorithm. Also, $\beta^m$ decreases with increasing $m$. Given $\alpha$ and $\beta$, the Armijo step size is in a sense the largest down-scaling of $\alpha\eta$ such that $\beta^m\alpha\eta$ is a sufficient descent direction. A re-writing of (2.8) yields

$$f(R_x(\beta^m\alpha\eta)) \leq f(x) + \sigma \mathrm{D}_{\beta^m\alpha\eta} f(x).$$

Thus, since $\sigma > 0$, $m$ must ensure that the directional derivative is non-positive to satisfy the expression. The Armijo point will determine the convergence requirement for the method. Theorem 4.3.1[1] states (and contains the proof) that

a method whose search directions are gradient based, and whose iterates, $\{x_k\}_k$, satisfy

$$f(x_k) - f(x_{k+1}) \geq c(f(x_k) - f(R_{x_k}(t_k^A \eta_k))), \tag{2.9}$$

for $c \in (0, 1)$, converges to a stationary point of the function $f$. The last criteria, together with the definition in (2.8), implies the necessary condition

$$f(x_k) - f(x_{k+1}) \geq -c\sigma \mathrm{D}_{t_k^A \eta_k} f(x_k),$$

which replicates the form of the $\mathbb{R}^n$ Wolfe condition. In particular, $x_{k+1} = R_{x_k}(t_k^A \eta_k)$, satisfy this condition. A retraction based update, $x_{k+1} = R_{x_k}(\alpha_k \eta_k)$, is also satisfactory if $\alpha_k$ is chosen such that $f(R_{x_k}(\alpha_k \eta_k)) \leq f(R_{x_k}(t_k^A \eta_k))$ (e.g. as a minimizer along $\eta_k$).

The theorem does only guarantee convergence to a stationary point, not necessarily a minimum. This should be kept in mind when performing the analysis.

### 2.4.2 Convergence rate

A useful theorem that gives a bound on the convergence rate of the line search method is theorem 4.5.6[1]. The proof is again tedious, but the theorem itself provides insight into which factors determine convergence rate. The theorem is stated next.

**Guaranteed Convergence Rate** (Theorem 4.5.6[1]). *Let $\eta_k = -\text{grad } f(x_k)$ and assume the algorithm converges to a point $x^*$, which is guaranteed to be a stationary point. Further, let $\lambda_{\min}$ and $\lambda_{\max}$ be the smallest and largest eigenvalue of the Hessian of $f$ at $x^*$. Assume $\lambda_{\min} > 0$, such that $x^*$ is a local minimizer. Then there exists an integer $K$ such that for $k \geq K$,*

$$f(x_{k+1}) - f(x^*) \leq (r + (1 - r)(1 - c)) \left(f(x_k) - f(x^*)\right), \tag{2.10}$$

*where $r$ is given in the interval $(r^*, 1)$, with*

$$r^* = 1 - \min\left\{2\sigma\bar{\alpha}\lambda_{\min}, 4\sigma(1 - \sigma)\beta\frac{\lambda_{\min}}{\lambda_{\max}}\right\}.$$

A qualitative analysis of the theorem follows. Let $c \approx 1$. This corresponds to each iteration satisfying $f(x_{k+1}) \leq f(R_{x_k}(t_k^A \eta_k))$. Choosing the exact minimizer at each step is sufficient to achieve this. Equation (2.10) then becomes

$$f(x_{k+1}) - f(x^*) \leq r\left(f(x_k) - f(x^*)\right) \tag{2.11}$$

$\sigma, \alpha$ and $\beta$ are related to the Armijo point. Consider $\sigma = 0$ such that the Armijo point is determined by $f(x) \geq f(R_x(\eta^A))$. This yields a rather trivially defined Armijo point, which does not contain a lot of information. On the other hand, consider $\sigma = 1$, that is $f(x) \geq f(R_x(\eta^A)) - \mathrm{D}_{\eta^A} f(x)$. For a concave function, this would yield $\eta^A = 0$ (seen by visualizing such a one-dimensional function).

For other functions, a steeper descent than the direction derivative at $x$ in direction $\eta$ is required to obtain a non-zero Armijo point. Hence, $\sigma$ should have a moderate value between $(0, 1)$. It will be assumed that choosing 0.5 can be done without loss of much generality. $\alpha$ may be interpreted as a scaling of $\eta$ that decides an upper bound on the Armijo point. It is hard to determine a general value to this variable. Since $\alpha = 1$ corresponds to setting $\eta$ as the maximum Armijo point, it is assumed reasonable that $\alpha = \omega(1)$ (in $\mathcal{O}$-notation). In particular, if $\alpha \gg \frac{1}{\lambda_{\max}}$ (and $\sigma$ and $c$ as discussed) the convergence interval from above is determined by

$$r^* = 1 - \beta\frac{\lambda_{\min}}{\lambda_{\max}} = 1 - \beta\kappa^{-1}$$

Of course when using the exact minimizer, plausible values for $\sigma$ are determined by the shape of the function. However, it is assumed that for any $x$ and direction $\eta$, setting $\sigma = 0.5$ will always include the exact minimum. The main purpose of this section is anyway to consider which parameters play the most important roles when considering the convergence of the methods. In this case, if the lower bound on $r$ is attained, and if steepest descent with exact line search is performed, then equation (2.11) becomes

$$\frac{f(x_{k+1}) - f(x^*)}{(f(x_k) - f(x^*)} \leq \beta\kappa^{-1}.$$

It is thus clear that the Hessian has significant impact on the bound on the convergence rate of the objective function.

In this thesis, the exact minimizer (or very close estimates) will be used in most cases, hence the parameters will not be studied further. Finally it will be noted that Newton's method also fits into this framework if the Hessian at every iteration is positive definite[5], because this ensures that the sequence of Newton vectors is indeed a gradient related sequence. This is seen directly by inserting the Newton stesps into the definition of a gradient-related sequence,

$$\langle \eta_k, \text{grad } f(x_k)\rangle = \langle \text{Hess } f(x_k)^{-1}(\text{grad } f(x_k)), \text{grad } f(x_k)\rangle.$$

If the Hessian is not positive definite, it can be modified (for instance by adding a positive multiple of the identity operator) to ensure that the Newton vectors are a gradient-related sequence. The ideas presented so far in this section are illustrated by an example using the multi-dimensional Rayleigh quotient.

### 2.4.3 Rayleigh quotient example

Convergence of the steepest descent algorithm applied to the multi-dimensional Rayleigh quotient is studied:

$$f(X) = \frac{1}{2}\text{tr}(X^T AX), \quad X \in \text{St}_n(p),$$

[5]Note that the Hessian is by default symmetric (self-adjoint).

and let $A$ be a symmetric positive definite matrix with distinct eigenvalues. The eigenvalues of the Hessian operator may be obtained by vectorizing the equation. Vectorization refers in this thesis exclusively to stacking the columns of a matrix on top of one another ($\text{vec}(Z)$ is the column vector $[Z_{1,1}, Z_{2,1}, \ldots, Z_{n,1}, Z_{1,2}, \ldots, Z_{n,p}]$). The eigenvalue decomposition of $A$ will be ordered such that the $p$ first eigenvalues are the same as $\Lambda$. From equation (2.2), the Hessian operator at a point where $AX = X\Lambda$ is

$$\text{Hess } f(X)[Z] = AZ - Z\Lambda - X\Lambda X^T Z,$$

which can be seen by insertion and using the projection operator and its directional derivative from section 2.3.4 (equation (2.4) and (2.5), respectively). It follows that,

$$A = \hat{X}\hat{L}\hat{X}^T, \quad \hat{X} = \begin{bmatrix} X, X_\perp \end{bmatrix}, \quad \hat{L} = \begin{bmatrix} \Lambda & 0_{p\times(n-p)} \\ 0_{(n-p)\times p} & L \end{bmatrix}$$

Finally, the vectorization of the Hessian is (with $\otimes$ denoting the Kronecker matrix product)

$$\begin{aligned} \text{vec}(\text{Hess } f(X)[Z]) &= [I_p \otimes \hat{X}\hat{L}\hat{X}^T - \Lambda \otimes I_n - I_p \otimes X\Lambda X^T]\text{vec}(Z) \\ &= (I_p \otimes \hat{X})[I_p \otimes \hat{L} - \Lambda \otimes I_n - I_p \otimes \hat{\Lambda}](I_p \otimes \hat{X}^T)\text{vec}(Z) \\ &= (I_p \otimes \hat{X})[I_p \otimes (\hat{L} - \hat{\Lambda}) - \Lambda \otimes I_n](I_p \otimes \hat{X}^T)\text{vec}(Z), \end{aligned}$$

where $\hat{\Lambda}$ is the $n \times n$ diagonal matrix retrieved by appending $n - p$ zeroes to the diagonal of $\Lambda$. The eigenvalues of the Hessian operator are thus (since $(I_p \otimes \hat{X}^T)$ is an orthogonal matrix),

$$\{\lambda_i - \lambda_j\}_{i=p+1:n, j=1:p} \cup \{-\lambda_i\}_{i=1:p}. \tag{2.12}$$

Note that by restricting $Z$ to the tangent space, not all of these necessarily applies. Also note that the $\lambda_i$s are not ordered by magnitude, but rather such that $\lambda_1, \ldots, \lambda_p$ corresponds to the eigenvalues at the stationary point $X$. However, if $\lambda_1, \ldots, \lambda_p$ do corresponds to the $p$ largest eigenvalues of $A$, the eigenvalues of the Hessian are all negative (since the $\lambda_i$s are distinct), and $X$ is thus a minimum. Without loss of much generality, it will in the following be assumed that the smallest eigenvalue is 1. The condition number of the stationary point is thus the largest eigenvalue divided by the "gap" from the eigenvalues corresponding to $X$ and the rest.

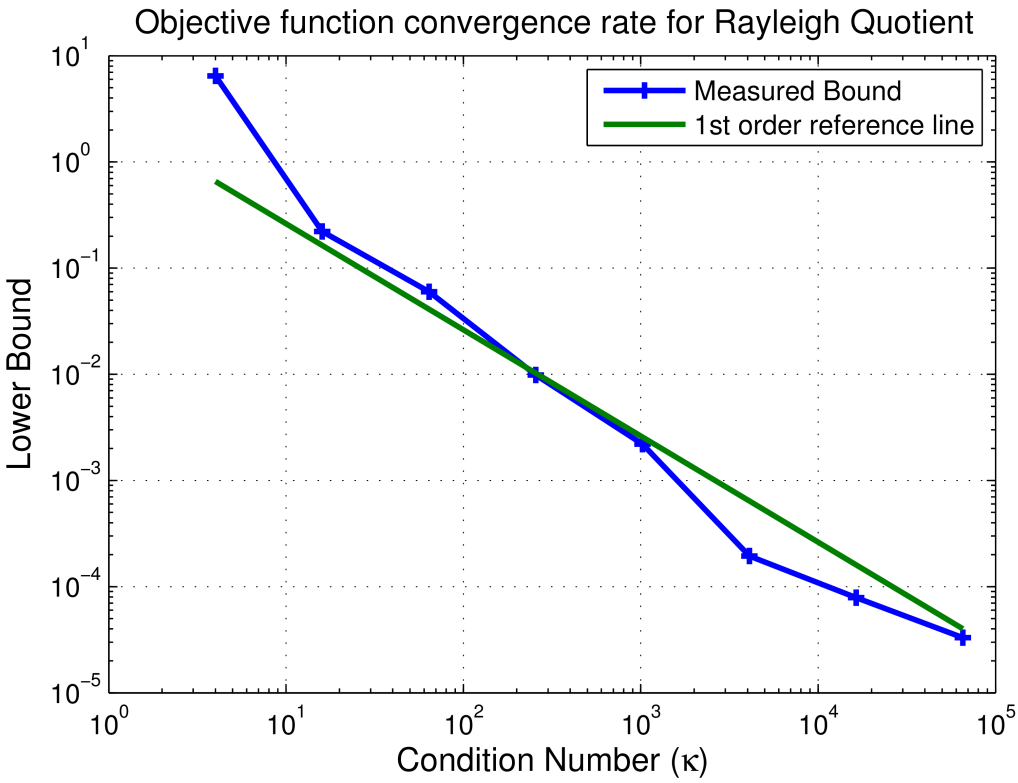


Figure 2.4: Convergence bound on the objective function for steepest descent and its dependence on condition number of the Hessian matrix. It is clear that a higher condition number for the Hessian matrix implies a slower convergence rate.

Figure 2.4 illustrates how the convergence bound may depend on the condition number of the Hessian when the objective function is the multi-dimensional Rayleigh quotient. The matrix was constructed in a similar fashion as in section 2.3.4 by first generating a random orthogonal matrix, $Q$, then constructing a diagonal matrix, $N$, and letting $A = QNQ^T$. This way, the eigenvalues of $A$ were easily manipulated, and thus the condition number of the Hessian matrix can be constructed. As seen above, the condition number of the Hessian matrix at the minimum is

$$\kappa = \frac{l_1}{l_{p+1} - l_p},$$

where $l_j$ is the $j$-th eigenvalue when sorted in descending order. By setting $l_p - l_{p+1} = \delta > 0$ and equally spacing the $p$ first and the $n - p$ last diagonal elements of $N$, the eigenvalues are ordered and distinct. The eigenvalues of $A$ are also all positive, thus the considerations above apply.

The $y$-axis in figure 2.4 is the measured minimum value of $(f_{k+1}) - f(x^*))/(f_k -$

$f(x^*)$) when $x_j$ is assumed to be relatively close to the minimum (plotted logarithmically), taken over all $k \geq j$. From the theorem(2.4.2), and the assumption that $c \approx 1$, it holds that

$$\frac{f_{k+1} - f(x^*)}{f_k - f(x^*)} \geq K\kappa^{-1},$$

where $K$ is some positive constant. This implies that a logarithmic comparison between the condition number and the lower bound should exhibit a linear relationship of order 1. This is observed to some extent in the figure. When the condition number lies is between 10 and $10^3$, there is a nice correspondance with the first order line. Towards larger condition numbers (in particular the three last data points), the convergence rate was actually better than expected, but it should be kept in mind that the theoretical bound is a lower bound.

It is interesting that the method performs better for large condition numbers, but the example is indeed illustrational and constructed. More rigorous testing is necessary to see if this is something that holds in the general case. Regardless of the exact order of the convergence rate, the condition number of the Hessian matrix appears to have a significant impact on the convergence rate, with a generally faster convergence for a smaller condition number (which is consistent with what one might expect).

# Chapter 3

# Blind Source Separation

In the introduction signal processing was mentioned as an application of optimization on manifolds. More specifically blind source separation will be considered. Consider the problem of determining a set of source signals, from another set of recorded signals. A typical example is the cocktail party problem. Assuming a group of individuals are gathered in a room and microphones are placed at random locations to record their voices, the objective is to retrieve the voices from the signals recorded by the microphones. The cocktail party problem is merely an intuitive explanation of the problem. Various restriction may be imposed without addressing the issue as a problem of human voices.

One such example is data from MEG (Magnetoencephalography) to remove effects like eye movement and blinking, amongst other activities that highly affect the measured brain signal[19]. Another application is underlying factors in economical data [6]. The return on a security can be assumed to be a sum of several underlying factors, such as seasonal variations, gross domestic product or federal or other tax rates. Blind source separation can be used to determine these factors, and which are the most significant ones.

Independent component analysis, which is a specific blind source separation techniqiue, has been used for text mining[14], where a number of news articles were chosen from certain topics and subtopics. The goal was to determine the topics and subtopics from the given texts. The independent components were assumed to be the subtopics (defined by a set of words), which were mixed together to the resulting articles.

A more detailed description of applications can be found in e.g. [19]. It is important to emphasize the perhaps clearest commonality. In all the examples one is able to measure several quantities that are assumed to result from the same underlying factors. The following sections presents and discuss the manifold approach of two blind source separation techniques: the sparse representation problem and independent component analysis.

## 3.1 Sparse Representation and Dictionary Learning

The problem is defined as finding a sparse solution, $s$, to the underdetermined (overcomplete) linear inverse problem

$$x(t) = As(t), A \in \mathbb{R}^{m \times n}, m \leq n \tag{3.1}$$

It is often assumed, and it will be here, that any $m$ columns of $A$ are linearly independent. The dependence on $t$ may be dropped for notational simplicity. $A$ is often referred to as the dictionary. The signals are dependent on time, meaning that under discrete sampling one may write (3.1) as

$$X = AS, \quad X, S \in \mathbb{R}^{m \times T}, \tag{3.2}$$

where $T$ is the number of samples. Time is defined rather abstractly here and may not refer to actual wall-time, but rather any measurements that can be represented as the $m \times T$-system above. In the literature it looks as if it is never emphasized which of the two representations above are considered. In reference [20] it is assumed that equation (3.1) is sufficient, and this practice is also adopted here. The transition to discrete time sampling is expected to always be clear. A noise-term can also be added to the model. Only independent and identically distributed Gaussian noise will be considered here, it is also the most common assumption.

The exact formulation of the noiseless case is

$$\min \|s\|_0, \quad \text{s.t. } As = x,$$

where $\|\cdot\|_0$ is defined as counting the number of non-zero elements (the 0 stems from the fact that it is the $p$-norm for $p = 0$, if one defines $0^0 = 0$). It has been proved that if $A$ is known, finding the sparsest solution is NP-hard[5], therefore it is reasonable to assume that the same problem with $A$ unknown is at least as hard. Alternative formulations are therefore sought.

Note that both $s$ and $A$ are assumed unknown. Finding these is a daunting task, especially since the norm $\|\cdot\|_0$ is not very suited for numerical computations. Several approximations of sparsity using continuous functions are suggested in the literature(see e.g. reference [16] for a comprehensive list of suggestions). Such measures will in the following be referred to as diversity measures. Diversity is the opposite of sparsity. The FOCUSS-algorithm, first presented by Gorodnitsky et al.[13], uses the $p$-norm for $0 < p \leq 1$, that is $d(x) = \sqrt[p]{\sum |x|^p}$. The $p$-norm is often denoted by $\|x\|_p$ or $l_p$. This will also be the case in this thesis, and $p$ will be set to 1. In a technical report by Donoho[10], it was argued that minimizing $l_1$ leads

to the same sparsity pattern (or support) as using $l_0$. The choice of diversity measure is revisited later, after some of the most popular methods have been presented.

Assuming one such measure have been chosen, one may view the optimization problem as the unconstrained problem over $A$ and $s$:

$$\min_{A,s} \frac{1}{2}\|As - x\|_2^2 + \lambda d(s), \tag{3.3}$$

where $\lambda$ is a scalar that for now will be interpreted as a weight parameter. An alternative derivation of equation (3.3) can be done from a Bayesian perspective[20] when $A$ is assumed to be known. In fact, if the sources satisfy being hypergeneralized Gaussians[1] and the model is assumed to contain Gaussian noise, $\lambda$ can be interpreted as the signal-to-noise ratio. The Bayesian perspective will not be emphasized in this thesis, therefore $\lambda$ will for now be understood as a weight or penalty parameter.

Some algorithms consider the case where $A$ is known and $s$ is sought. As mentioned earlier such problems are NP-hard, and approximate solutions are sought instead. The formulation is basically the same as (3.1), except that $A$ is removed from the argument list. Some of the algorithms in the literature are (Orthogonal) Matching Pursuit, Basis Pursuit and FOCUSS. The latter is considered in this thesis for comparison to the manifold-based methods.

If the dictionary is unknown, learning algorithms are implemented instead. Commonly, the solution is found by a 2-step iterative procedure. For every iteration, it is first assumed that the dictionary is given, and the signals are found using e.g. one of the methods mentioned above. Next, the dictionary is updated to adapt to the changes that occurred in the previous step. The procedure is continued until some convergence criterion is met. The two steps are referred to as sparse coding and codebook update (or dictionary update)[5]. Some of the algorithms (and references to literature) for the codebook update are:

- Maximum Likelihood Estimates, maximize $P(A|x)$(see [5] and references therein)
- Method of Optimal Directions (MOD)[12]
- Maximum a posteriori probability[20], $P(A|x) \propto P(x|A)P(A)$
- K-SVD[5] (K-means, column-wise update)

### 3.1.1 The manifold adaption

Observe that the signals may be scaled and ordered arbitrarily,

$$As = AP\Lambda\Lambda^{-1}Ps = \tilde{A}\Lambda^{-1}Ps = \tilde{A}\tilde{s},$$

[1] A hypergeneralized Gaussian distribution have a probability density function on the form $P(x) = Z^{-1}\exp(-\gamma d(x))$, where $Z$ is a normalization parameter. $\gamma$ is a scalar, and $d(x)$ must satisfy $d(x) = d(|x|)$ and $d(0) = 0$.

for any diagonal matrix $\Lambda$ of non-zeros and perturbation matrix $P$. It therefore makes sense to impose additional constraints on $A$. Instead of just assuming $A \in \mathbb{R}^{m\times n}$, two other manifold structures are considered:

$$\mathcal{A}_F = \{A \in \mathbb{R}^{m\times n} : \|A\|_F = 1\}, \quad \mathcal{A}_C = \{A \in \mathbb{R}^{m\times n} : \|a_i\|_2 = 1/\sqrt{n}\}, \tag{3.4}$$

where $a_i$ are the columns of $A$, and $\|\cdot\|_F$ denotes the Frobenius norm. $\|\cdot\|_2$ denotes the regular 2-norm. Note that $\mathcal{A}_C \subset \mathcal{A}_F$. $\mathcal{A}_F$ is in a sense the $(nm-1)$-dimensional sphere when $A$ is vectorized. Let $\vec{A}$ and vec$(A)$ both denote the vectorized version of $A$. Due to the sphere-analogy, the projection operator of $\mathcal{A}_F$ is

$$\mathrm{P}_{\vec{A}} = (I_{nm} - \vec{A}\vec{A}^T),$$

which applied to a vectorized matrix yields,

$$\mathrm{P}_{\vec{A}}\vec{Z} = \vec{Z} - (\vec{Z}^T\vec{A})\vec{A},$$

thus the matrix version of the applied projection operator becomes

$$\mathrm{P}_A Z = Z - \mathrm{tr}(Z^T A)A.$$

Moreover, $(A + Z_A)/\|A + Z_A\|_F$ is a retraction on $\mathcal{A}_F$ (properties i) and ii) are readily verified by insertion), but it is not the geodesic. The geodesic that satisfy $\gamma(0) = A$ and $\dot{\gamma}(0) = Z_A$ is

$$\gamma(t) = A\cos(\|Z_A\|_F \cdot t) + \frac{Z_A}{\|Z_A\|_F}\sin(\|Z_A\|_F \cdot t),$$

analogous to the geodesic on the sphere. However, sine and cosine are sensitive for large values, meaning that if this geodesic is applied it is likely that a renormalization has to be carried out to numerically ensure $\gamma(t) \in \mathcal{A}_F$. For this reason, $(A + Z_A)/\|A + Z_A\|_F$ is used in the following, and since it is a retraction, it is known from the previous analysis that it is a proper alternative to the geodesic.

Kreutz-Delgado et al.[20] reported that the optimization on $\mathcal{A}_C$ showed best performance, followed by $\mathcal{A}_F$, and then $\mathbb{R}^{m\times n}$ (at least for the FOCUSS-based method, which will be presented in section 3.1.3).

If the problem is attempted solved by the two steps mentioned, an option is to use steepest descent on the dictionary update (as done by Kreutz-Delgado et al.[20]). Following the discussion from section 2.1, the gradient can be computed by applying the projection operator to the Euclidean gradient. Let $\phi$ denote the objective function,

$$\phi(A, S) = \frac{1}{2T}\|AS - X\|_F^2 + \lambda d(S). \tag{3.5}$$

The division by 2 is for computational convenience. $A, S$ and $X$ are as defined in equation (3.2). Dividing by $T$ ensures that the the first term is the average residual error, in the 2-norm, sampled at every time step. The diversity measure is assumed

to apply column-wise, meaning that if the $p$-norm is used as a diversity measure, $d(S)$ is the sum of the $p$-norm over the individual columns. Let $\nabla_A$ denote taking the gradient of $\mathbb{R}^{n\times p}$ with respect to $A$. It follows that

$$\begin{aligned}\nabla_A\|AS-X\|_F^2 &= \nabla_A(\mathrm{tr}((AS-X)^T(AS-X))\\ &= \nabla_A(\mathrm{tr}(S^TA^TAS-2X^TAS+X^TX)\\ &= \nabla_A(\mathrm{tr}(ASS^TA^T-2X^TAS)\\ &= 2ASS^T-2XS^T,\end{aligned}$$

thus, the gradient is given by

$$\nabla_A\phi = \frac{1}{T}(ASS^T-XS^T). \tag{3.6}$$

Note that for a zero-mean signal, $SS^T/T$ approximates the covariance matrix of the signals given by $S$. In case of a different matrix structure, the appropriate gradient is readily found by projection.

A different approach is considered next. The sparse representation problem was introduced as a problem depending on both $A$ and $s$. It is therefore natural to try and consider it as equally dependent on both $A$ and $s$. The Cartesian product of two manifolds can also be given a manifold structure[21], and this will be utilized here. Consider the minimization of $\phi$ defined in (3.5), over the manifold $\mathcal{A}\times\mathcal{S}$. For now $\mathcal{S}=\mathbb{R}^{n\times T}$, but it may be of interest to consider other possible manifold structures on the signals. This is not within the scope of this thesis. Recall that $T$ is the number of time samples. In a similar manner as above, one may show that

$$\nabla_S\phi = \frac{1}{T}(A^TAS-A^TX)+\lambda\nabla_S d. \tag{3.7}$$

Note that this requires the gradient of the diversity measure. For descriptive purposes, it is possible to consider an element of the manifold $\mathcal{A}\times\mathcal{S}$ as the vector $[\mathrm{vec}(A)^T,\mathrm{vec}(S)^T]^T$, and accordingly it is natural to identify an element of the tangent space as $[\mathrm{vec}(Z_A)^T,\mathrm{vec}(Z_S)^T]^T$. The tangent space of the Cartesian product of two manifolds manifold at a point, is the Cartesian product of the tangent spaces at that point(see e.g. Lee[21], proposition 3.14). Hence, the projection operator for the vectorized interpretation is

$$\mathrm{P}_{\vec{A}\vec{S}} = \begin{bmatrix}\mathrm{P}_{\vec{A}} & 0\\ 0 & \mathrm{P}_{\vec{S}}\end{bmatrix} = \begin{bmatrix}\mathrm{P}_{\vec{A}} & 0\\ 0 & I_{nT}\end{bmatrix}$$

It should be emphasized that a practical implementation of the manifold, and operations on its elements, need not operate with vectorized matrices, but for illustrational purposes, this notation is used. It will often be clear how for instance projection operators, gradients or Hessians should be implemented using the pure matrix structure instead. With the projection operator and the gradient defined, steepest descent is readily implemented, either with a constant step size or using a more adaptive procedure like exact line search.

## Hessian Computations

Regardless of the manifold, the Hessian is found by first computing the Euclidean Hessian. Using vector notation, it is

$$\partial^2\phi(\vec{A},\vec{S}) = \begin{bmatrix} J_{\vec{A}}(\nabla_{\vec{A}}\phi) & J_{\vec{S}}(\nabla_{\vec{A}}\phi) \\ J_{\vec{A}}(\nabla_{\vec{S}}\phi) & J_{\vec{S}}(\nabla_{\vec{S}}\phi) \end{bmatrix} = \begin{bmatrix} J_{\vec{A}}(\nabla_{\vec{A}}\phi) & J_{\vec{S}}(\nabla_{\vec{A}}\phi) \\ J_{\vec{S}}(\nabla_{\vec{A}}\phi)^T & J_{\vec{S}}(\nabla_{\vec{S}}\phi), \end{bmatrix}$$

where $J$ denotes the Jacobian matrix with respect to its subscript. The last equality follows either direct calcularion, or the fact that the Euclidean Hessian is symmetric. Consider first the upper left block:

$$\begin{aligned} J_{\vec{A}}(\nabla_{\vec{A}}\phi) &= \frac{1}{T} J_{\vec{A}}((SS^T \otimes I_m)\vec{A} - \mathrm{vec}(XS^T)) \\ &= \frac{1}{T}(SS^T \otimes I_m) \end{aligned}$$

The upper right block can be determined by

$$\begin{aligned} J_{\vec{S}}(\nabla_{\vec{A}}\phi)Z_{\vec{S}} &= \frac{1}{T} J_{\vec{S}}((SS^T \otimes I_m)\vec{A} - \mathrm{vec}(XS^T))\mathrm{vec}(Z_S) \\ &= \frac{1}{T}\mathrm{vec}[\mathrm{D}_{Z_S}(ASS_T - XS^T)] \\ &= \frac{1}{T}\mathrm{vec}[A(SZ_S^T + Z_S S^T) - XZ_S] \\ &= \frac{1}{T}(I_n \otimes (AS - X)\sigma + S \otimes A)\mathrm{vec}(Z_S), \end{aligned}$$

where $\sigma$ is the perturbation operator (or matrix) such that $\mathrm{vec}(Z_S^T) = \sigma\mathrm{vec}(Z_S)$[2]. Hence,

$$J_{\vec{S}}(\nabla_{\vec{A}}\phi) = \frac{1}{T}(I_n \otimes (AS - X)\sigma + S \otimes A),$$

Finally the lower right block is

$$\begin{aligned} J_{\vec{S}}(\nabla_{\vec{S}}\phi) &= J_{\vec{S}}((I_T \otimes A^T A)\vec{S} - \mathrm{vec}(A^T X) + \lambda\nabla_{\vec{S}} d(\vec{S})) \\ &= I_T \otimes A^T A + \lambda J_{\vec{S}}(\nabla_{\vec{S}} d(\vec{S})), \end{aligned}$$

which for large time samples is block diagonal when the diversity measure is chosen as will be discussed in section 3.1.2.

Section 2.3 noted that computing the Hessian on a manifold had to take into account the variation of the projection operator. Consider the projection of the Euclidean Hessian (the first term in equation (2.2)):

$$\mathrm{P}_{\vec{A}\vec{S}}\partial^2\phi(\vec{A},\vec{S})\vec{Z} = \begin{bmatrix} \mathrm{P}_{\vec{A}} & 0 \\ 0 & I_{nT} \end{bmatrix} \begin{bmatrix} J_{\vec{A}}(\nabla_{\vec{A}}\phi) & J_{\vec{S}}(\nabla_{\vec{A}}\phi) \\ J_{\vec{S}}(\nabla_{\vec{A}}\phi)^T & J_{\vec{S}}(\nabla_{\vec{S}}\phi), \end{bmatrix} = \begin{bmatrix} \mathrm{P}_{\vec{A}} J_{\vec{A}}(\nabla_{\vec{A}}\phi) & \mathrm{P}_{\vec{A}} J_{\vec{S}}(\nabla_{\vec{A}}\phi) \\ J_{\vec{S}}(\nabla_{\vec{A}}\phi)^T & J_{\vec{S}}(\nabla_{\vec{S}}\phi), \end{bmatrix}.$$

[2] It can be interpreted as the $n^2 \times n^2$ identity matrix with its rows permuted as $[1:n] \otimes \mathbf{1}_n + \mathbf{1}_n \otimes [0:n-1]n$. $[i:j]$ is defined as the vector $[i, i+1 \ldots j-1, j]$.

Furthermore, the second term of equation (2.2) is

$$\begin{aligned}
\mathrm{P}_{\vec{A}\vec{S}}(\mathrm{D}_{\vec{Z}}\mathrm{P}_{\vec{A}\vec{S}})\nabla\phi &= \mathrm{P}_{\vec{A}\vec{S}}\begin{bmatrix}\mathrm{D}_{\vec{Z_A}}\mathrm{P}_{\vec{A}} & 0_{mn\times nT}\\ 0_{nT\times mn} & \mathrm{D}_{\vec{Z_S}}\mathrm{P}_{\vec{S}}\end{bmatrix}\begin{bmatrix}\nabla_{\vec{A}}\phi\\ \nabla_{\vec{S}}\phi\end{bmatrix}\\
&= \mathrm{P}_{\vec{A}\vec{S}}\begin{bmatrix}\mathrm{D}_{\vec{Z_A}}\mathrm{P}_{\vec{A}}\nabla_{\vec{A}}\phi\\ 0_{nT\times 1}\end{bmatrix}\\
&= \begin{bmatrix}\mathrm{P}_{\vec{A}}\mathrm{D}_{Z_{\vec{A}}}\mathrm{P}_{\vec{A}}\nabla_{\vec{A}}\phi\\ 0_{nT}\end{bmatrix}
\end{aligned}$$

For $\mathcal{A}_C$, the projection operator is $\mathrm{P}_{\vec{A}} = I - \vec{A}\vec{A}^T$ and $\mathrm{D}_{\vec{Z_A}}\mathrm{P}_{\vec{A}} = -(\vec{A}\vec{Z}_A^T + \vec{Z}_A\vec{A}^T)$. Furthermore, $\mathrm{P}_{\vec{A}}\mathrm{D}_{\vec{Z_A}}\mathrm{P}_{\vec{A}} = -\vec{Z_A}\vec{A}^T$. The matrix representation of the Hessian is thus

$$\text{Hess }\phi(\vec{A},\vec{S}) = \begin{bmatrix}\mathrm{P}_{\vec{A}}J_{\vec{A}}(\nabla_{\vec{A}}\phi) & \mathrm{P}_{\vec{A}}J_{\vec{S}}(\nabla_{\vec{A}}\phi)\\ J_{\vec{S}}(\nabla_{\vec{A}}\phi)^T & J_{\vec{S}}(\nabla_{\vec{S}}\phi)\end{bmatrix} - \begin{bmatrix}(\vec{A}^T\nabla_{\vec{A}}\phi)I_{mn} & 0\\ 0 & 0\end{bmatrix} \tag{3.8}$$

Observe that due to the projection of the upper right block of the matrix, and not the lower left block, the matrix itself is no longer symmetric, which appears to contradict the fact that the Hessian should be symmetric. However, when viewed as an operator on the tangent space at $A$, one may verify that the operator is indeed self-adjoint.

### 3.1.2 Diversity measures

The choice of diversity measures is not studied in great depth in this thesis since the $l_1$ measure is assumed to suffice. However, since it is such an essential part of the optimization formulation, it is important to be wary of the limitations that follow from the choice of measure.

The goal of the diversity measure is obviously to encourage sparse solutions. Hurley and Rickard[16] suggests six sparseness properties that a diversity measure should be able to handle. The study is not directly related to the sparse representation problem, but how one may determine if (and to what extent) a data set is sparse.

Gorodnitsky[13] suggests that the gradient of the diversity measure can in many cases be factorized as

$$\nabla_s d(s) = \alpha(s)\Pi(s)s,$$

where $\alpha(s)$ is a positive scalar function of $s$ and $\Pi(s)$ is a symmetric positive definite matrix. For the $l_1$-norm, the gradient is the sign function, $\text{sgn}(s)$, thus $\alpha(s) = 1$, $\Pi(s) = \text{diag}(|s|^{-1})$. The FOCUSS algorithm takes explicit use of the factorization, but the factorization is not useful for the gradient-based ones. The gradient is discontinuous whenever $s_i = 0$, which may be unfortunate for numerical methods. However since these discontinuities appear almost nowhere, it will be assumed that this choice is valid even when using gradient and Hessian-based methods. The Hessian of the $l_1$-norm is undefined for any signals that are 0. In the implementation

the value is set to 0, which is not ideal, but since numerical values are assumed to rarely be exactly 0, it is assumed to be a good enough ad-hoc fix. It also provides a much more efficient implementation, since $J_{\bar{S}}(\nabla_{\bar{S}}\phi)$ becomes $I_T \otimes A^T A$, which applied to $\text{vec}(Z_S)$ becomes $A^T A Z_S$.

The lack of smoothness of the $l_1$-norm provides challenges for numerical methods, but it will be assumed that only minor modifications are necessary in order to make gradient-based methods converge. The elements of the Hessian are all zeroes, except if $s_i = 0$, where it is undefined. Arguing again that $s_i$ being identically zero is a rare case leads to the assumption that setting the Hessian to all zeroes, regardless of the values of $S$, is sufficient for a practial implementation of Newton's method. It does in particular lead to a very efficient implementation. Smoother diversity measures could also be studied for such methods, but it is not within the scope of this thesis.

A final remark is that the diversity measure will sometimes be referred to as dependent on the vector $s$, and sometimes on the matrix $S$. To avoid confusion, it is clarified here that a diversity measure is evaluated on a vector. If the notation $d(S)$ is used, it means summing up the diversity of all the columns of $S$. Potentially, one may use the average diversity instead, but since the weighting parameter of sparsity, $\lambda$, is considered as a penalty parameter, division by $T$, or any other constant, will only alter how $\lambda$ should be chosen. By assuming this property on the diversity measure, the Jacobian of $d$ for vectorized $S$ becomes block-diagonal, implying that the lower right block of the Hessian of $\phi$, $B_4$, is block-diagonal as well. This greatly improves the efficiency of computing the Hessian.

### 3.1.3 FOCUSS

The FOCUSS algorithm is designed to solve the sparse representation problem when the matrix $A$ is known. It is an acronym for Focal Underdetermined System Solver. Consider equation (3.3), and let $A$ be known, thus not an argument of the objective function. Setting the gradient with respect to $s$ to 0 yields the generally non-linear equation for $s^*$,

$$A^T(As^* - x) + \lambda \nabla_s d(s^*) = 0,$$

which is shown[13] to be equivalent to

$$s^* = \Pi^{-1}(s^*)A^T(\alpha(s^*)\lambda I + A\Pi^{-1}(s^*)A^T)^{-1}x, \tag{3.9}$$

when the factorization of the gradient of $d$, introduced above, is used. This becomes the basis for a fixed point iteration which is the FOCUSS algorithm. In an article by He et al.[15], it was shown that the FOCUSS algorithm is a quasi-Newton method, and that it possesses certain superlinear convergence properties, at least for $0 < p < 1$.

When implemented on the unknown dictionary case, it was suggested[20] that one may alternate between using the FOCUSS algorithm and a steepest descent method with constant step size on the dictionary to ensure convergence. The constant step size is often referred to as the learning rate. Note that the FOCUSS algorithm operates on one time sample at a time. It can therefore make sense to update a batch of the time samples before updating the dictionary, instead of updating all the time samples before considering the dictionary.

### 3.1.4 Complexity

To get an idea of the efficieny of the methods, it is necessary to take into consideration complexity and potential run time. Steepest descent requires the computation of $\nabla_A \phi$ and $\nabla_S \phi$, where the first one is given by equation (3.6), and takes $\mathcal{O}(n^2T + mn^2 + mnT) = \mathcal{O}(n^2T)$ flops to compute[3]. $\nabla_S \phi$ is given by equation (3.7). By doing $A^T(AS)$ instead of $(A^TA)S$, it can be done in $\mathcal{O}(3nmT)$ flops. The gradient of the diversity measure is not included, but due to the relatively simple form, its complexity is negligible. The overall complexity for computing the gradient is thus

$$\mathcal{O}(4mnT + n^2T).$$

If a manifold structure is imposed on $A$ as well, a retration has to be performed as well. For $\mathcal{A}_F$, a retraction can be to divide $A$ by its Frobenius norm, which is a relatively cheap procedure. The cost of the retraction is therefore considered negligible.

The FOCUSS-based algorithm uses the gradient with respect to $A$ as well, but this is not the most complex step. The iteration given by (3.9) requires the inversion of an $m \times m$ matrix, which has to be performed independently for each time sample. There may be some matrix structure to take advantage of, considering the relatively simple form it takes. However, performing this step on all time samples is a priori $\mathcal{O}(Tm^3)$. The gradient is computed every $T_N$ time steps, hence the total cost is

$$\mathcal{O}\left(m^3T + n^2T \cdot \left(\frac{T}{T_N}\right)\right)$$

In a practical case, $T_N$ will be relatively high, such that the ratio $T/T_N$ is a constant that can be excluded, thus only the first term is necessary to represent the complexity.

For Newton's method, it helps to consider the four blocks of the Hessian,

$$\text{Hess } \phi(A,S)[Z_A, Z_S] = \begin{bmatrix} \mathrm{P}_A B_1 & \mathrm{P}_A B_2 \\ B_2^T & B_4 \end{bmatrix} \begin{bmatrix} Z_A \\ Z_S \end{bmatrix}$$

Computing the matrix-vector product means computing $\mathrm{P}_A(B_1 Z_A + B_2 Z_S$ and $B_2^T Z_A + B_4 Z_S$. Note that the sizes of $B_1, B_2$ and $B_4$ are respectively $mn \times mn$, $mn \times nT$ and $nT \times nT$. The projection operator is relatively cheap, hence the a

[3]The last equality follow from the assumption $m < n$ and $T > n$.

priori complexity is $\mathcal{O}((mn)^2 + 2mn^2T + (nT^2))$, and that is without taking into consideration that the matrix has to be formed. However, since $B_4$ is sparse and the matrix-vector product can be computed by $A^T AZ_S$, the last term of the complexity is instead $2mnT$. In a similar fashion, $B_1 Z_A$ can be computed in $\mathcal{O}(n^2T)$, which actually is more, but takes into account forming the matrix. Moreover, $B_2 Z_S$ is done in $\mathcal{O}(mnT+n^2T)$, and $B_2^T Z_A$ is $\mathcal{O}(mnT+n^2T)$. Hence, the overall complexity of computing the matrix-vector product is,

$$\mathcal{O}(2n^2T + 2Tmn) = \mathcal{O}(n^2T),$$

which is much lower than compared to a dense Hessian matrix-vector product which would be $\mathcal{O}((nm + nT)^2) = \mathcal{O}(n^2T^2)$. Ideally, an iterative method can be implemented that requries few steps ($\ll nm+nT$) to converge. Alternatively the system can be solved directly by block-wise inversion, which would also be relatively cheap.

In conclusion, computing the gradient and applying the Hessian is of the same complexity. Furthermore, performing a Newton step is (approximately) as complex as a steepest descent up to a constant that depends on the required number of iterations to solve the Newton equation. FOCUSS is, at least for the problem sizes in this thesis, $m$ times more complex than steepest descent and Newton's method.

## 3.2 Independent Component Analysis

Independent Component Analysis(ICA) is another case of blind source separation. The two main aspects of what separates it from the sparse representation problem is that the sources are required to be statistically independent and that $n = m$. It is also assumed that the variance of $s$ is 1, but this is merely a restriction from the scale and perturbation invariance of $s$ (as mentioned in the previous section). In the literature, $n < m$ and $n > m$ are also considered, but $n > m$ is basically the sparse representation problem with the independence of signals assumption. To have a clear distinction between ICA and sparse representation, ICA will in this thesis therefore always imply that $n = m$. It will also be assumed that $s$ is a zero-mean signal without loss of generality (in case of a nonzero mean, a simple pre- and postprocessing step is added).

Due to the independence of signals, pre-whitening the measured signal is a useful tool. This is done by computing the eigenvalue decomposition of the covariance matrix of the measured signals,

$$\mathrm{E}(xx^T) = \frac{1}{T-1} XX^T = VDV^T,$$

then premultiply by $VD^{-1/2}V^T$ to obtain independent signals. Division by $T-1$ is to ensure a biased estimator. Let $\tilde{x}$ denote the pre-processed signals,

$$\tilde{x} = VD^{-1/2}V^T x \implies \mathrm{E}(\tilde{x}\tilde{x}^T) = VD^{-1/2}V^T \mathrm{E}(xx^T) VD^{-1/2}V^T = I_n.$$

A reformulation of the blind source separation problem is thus

$$\tilde{x} = VD^{-1/2}V^T x = VD^{-1/2}V^T As, \tag{3.10}$$

Since $s$ contains statistically independent signals, it must hold that

$$\mathrm{E}(\tilde{x}\tilde{x}^T) = (VD^{-1/2}V^T A)(VD^{-1/2}V^T A)^T = I,$$

hence $VD^{-1/2}V^T A$ must be an orthogonal matrix, $W$, and (3.10) becomes

$$\tilde{x} = Ws \iff s = W^T\tilde{x}.$$

The blind source separation formulation is thus rewritten as a problem on the orthogonal group.

In the previous problem, finding $s$ was based on the assumption that the signals were sparse, and an objective function was formed to minimize the error, but also take into account the sparsity of the solution. Now however, the signals are explicitly given by $W^T\tilde{x}$, hence any objective function will exclusively depend on $W$, and not $s$. A different assumption is therefore imposed on the signals.

The central limit theorem states that the probability distribution function of a linear combination of random variables, is a priori closer to a Gaussian distribution than the individual signals. The goal of the objective function is therefore to measure non-Gaussianity, and accordingly the optimization formulation attempts to maximize the non-Gaussianity of $s$. Such an objective function is referred to as a contrast function.

### 3.2.1 Contrast functions

The quantification of non-Gaussianity is not straightforward. The sought contrast function should map a random variable to $\mathbb{R}$. Random variables have probability distributions, hence it makes sense to consider how various variables can be seperated based on their distributions. The entropy of a random variable is defined as

$$H(y) := -\int f(y)\log f(y)dy,$$

where $f$ is the probability density function, and it can be shown that no random variables have a larger entropy than a gaussian variable. It therefore makes sense to define *negentropy*:

$$J(y) = H(y_{gauss}) - H(y),$$

where $y_{gauss}$ is a Gaussian variable with the same mean and variance as $y$. The property of $J$ is now that for a Gaussian variable it is zero and positive for (all) other distributions. From a statistical point of view, this is optimal [17]. It is however computationally difficult to compute, hence approximations are made. One such approximation is obtained by choosing some appropriate functions, $G_i$s, and let

$$J(y) \approx \sum_{i=1}^{k}[E(G_i(y)) - E(G_i(\zeta))]^2.$$

Here $\zeta$ is a gaussian variable with the same mean and variance as $y$. In the following, $y$ will have zero mean and unitary variance. Depending on any a priori knowledge about the independent components, one may adapt the choice of the $G_i$s. For simplicity, only $k = 1$ is considered in this thesis. When it is clear from the context, $G$ may be referred to as the contrast function (since it defines the approximation function). Hyvarinen and Aapo[17] gives a rigorous explanation of how to choose the function in practice, at least when implemented using the FastICA-algorithm. However, several of the ideas apply to the problem regardless of how the implementation is done. They argue that the following should be kept in mind when considering the choice of contrast,

1. If the ICA model is correct and $G$ is sufficiently smooth and even, then the set of local maxima for the approximation, $J_G(w)$, includes the $i$th row of the orthogonal matrix $W$.

2. If $G$ is on the form $G(u) = k_1 \log f_i(u) + k_2 u^2 + k_3$. Where $f_i$ is the density function of $s_i$, then the trace of the asymptotic variance of $w$ is minimized. This can be considered as a criteria to optimize the efficiency of the estimator (of negentropy).

3. If $G$ is bounded, or at least a function that grows slowly when $u$ becomes large, it will be robust against outliers.

Point 2) is the optimal choice with respect to the variance property mentioned. The main idea behind point 2) is that for a finite number of samples there exists a best approximation to negentropy. Therefore, the approximation function should preferrably be close to as good as the optimal choice. It is reasonable to assume that a function behaving similarly (with respect to $u$) to $k_1 \log f_i(u) + k_2 u^2 + k_3$ is such a function.

Next are two particular points to consider. The first is computational complexity ($G$ and its derivatives should be easy to compute). The second is that with a fixed choice of contrast function, signals with different distributions are weighted differently, such that certain maxima are "more important" than others. If for instance only one signal is sought (the optimization is carried out on $\mathrm{St}_n(1)$), certain basins of attraction can be exceedingly large such that a random initial guess will in almost all cases converge to the same maxima. This can however be taken advantage of, if information on the signals is available beforehand. The functions that were suggested are the following three:

1. $G_1 = \frac{1}{a_1} \log \cosh(a_1 u)$
2. $G_2 = -\frac{1}{a_2} \exp(-a_2 u^2/2)$
3. $G_3 = \frac{1}{4} u^4$

Experimentally, $a_2 = 1$ and $1 \leq a_1 \leq 2$ have proven to be convenient choices[17]. Having these three functions to choose from, the following is summarized:

- $G_1$ is a good general-purpose contrast function

- $G_2$ may be better if the independent components are known to be super-Gaussian (which are often according to the same article) or when robustness is important.
- $G_1$ and $G_2$ may be approximated in case they are too computationally expensive.
- $G_3$ requires practically no presence of outliers

Due to its computational and conceptual simplicity, kurtosis will be used in the experiments later.

Searching for the least Gaussian signals is just one way of approaching the ICA problem. Minimization of mutual information is another measure that can be used. Other contrasts are not studied here, but an interesting common property of constrast functions is that the Hessian matrix becomes block-diagonal, with $n \times n$ sized blocks. The next section shows what this means for the particular choice of $G_3$. As will hopefully be clear, the block-diagonal structure is caused by the fact that it is the *distribution* of the time sampling of the signal that determines the value of the contrast function. In other words, reordering the columns of the $n \times T$ matrix $\tilde{X}$ does not affect the value of the contrast function. The linear relationship between $\tilde{x}$ and $s$ is also necessary.

### 3.2.2 Manifold adaption

It was mentioned earlier how $s$ is uniquely determined by $W$. Assuming then that a contrast function, $\phi$, has been chosen, the optimization formulation is

$$\max_{W \in O(n)} \phi(W^T \tilde{x}),$$

where $O(n)$ is the orthogonal group. It should be noted that if only a few of the signals are of interest, the optimization may be performed over $\mathrm{St}_n(p)$ instead. In particular, if any statistical properties of the required signal(s) such as expected kurtosis are known, then a clever choice of contrast, and optimization over $\mathrm{St}_n(p)$ could prove very effective. In the following, it will be assumed that all the signals, and thus an orthogonal matrix, are sought.

The tools from the previous chapter applies readily. Steepest descent (or ascent, depending on how the implementation is done) is for instance one method that is well defined. A retraction is also needed, and several exists in the literature. QR-factorization, the Cayley-transform related retraction mentioned in section 2.2.1 and the explicit geodesic to mention a few. The projection operator is given by

$$\mathrm{P}(Z) = W\mathrm{skew}\,(W^T Z)$$

For the computation of the gradient, consider $\phi(s) = 1/2 \sum_i [\mathrm{E}(g(s_i)) - C]^2$ as the contrast function. With a slight abuse of notation, let $y_i = y_i(t)$ denote signal $i$ of the vector $y(t)$, while $Y_i$ is the row vector containing the time samples of $y_i(t)$. For a measured signal, the expected value is replaced by the mean of the signal. Let $g(S_i)$ denote the $T$-dimensional column vector containing the sampling of $g(s_i)$. The contrast function is written

$$\phi(S) = \frac{1}{2} \sum_i \left[ \frac{1}{T} \mathbf{1}_T^T g(S_i) - C \right]^2,$$

where $\mathbf{1}_T$ is the vector of length $T$ containing only ones. Before computing the gradient of the expression, note that each term in the above sum depends on one, and only one, of the signals and thus only one of the columns of $W$. The gradient is therefore found term-by-term, through differentiation with respect to the appropriate column in $W$. Let $z = w_i$ (column $i$ of the matrix $W$), such that $S_i = w_i^T \tilde{X}$ and compute

$$\begin{aligned} \nabla_z \left[ \frac{1}{T} \mathbf{1}_T^T g(z^T \tilde{X}) - C \right]^2 &= 2 \left[ \frac{1}{T} \mathbf{1}_T^T g(z^T \tilde{X}) - C \right] \left( \nabla_z \left[ \frac{1}{T} \mathbf{1}_T^T g(z^T \tilde{X}) - C \right] \right), \\ &= 2 \left[ \frac{1}{T} \mathbf{1}_T^T g(z^T \tilde{X}) - C \right] \left( \frac{1}{T} \tilde{X} g'(z^T \tilde{X}) \right) \end{aligned} \tag{3.11}$$

where the first factor is a scalar, and the second factor is an $n$-dimensional vector. Writing out the entire gradient thus yields,

$$\nabla_W \phi = \tilde{X} G'(W) D(W),$$

where $D(W) = \mathrm{diag}(\frac{1}{T} \mathbf{1}_T^T g(w_i^T \tilde{X}) - C)$ and the $i$th column of $G'(W)$ is $g'(w_i \tilde{X})$. In a similar fashion, the Hessian may be computed. Again, due to the lack of interaction between the columns of $W$ in the contrast function, the Hessian need only be computed "locally". Thus it is sufficient to compute the Jacobian of (3.11) with respect to $z$,

$$J_z(\nabla_z(...)) = \frac{1}{T^2} (\tilde{X} g'(z^T \tilde{X}))(\tilde{X} g'(z^T \tilde{X}))^T + \frac{d(z)}{T} \tilde{X} \mathrm{diag}(g''(z^T \tilde{X})) \tilde{X}^T, \tag{3.12}$$

with $d(z) = \frac{1}{T} \mathbf{1}_T^T g(z^T \tilde{X}) - C$. Vectorizing $W$ and viewing the Hessian as an $n^2 \times n^2$ matrix yields a block diagonal matrix,

$$\mathrm{Hess}\, \phi(\mathrm{vec}(W)) = \begin{bmatrix} J_z(\nabla_z(...))|_{z=w_1} & & & \\ & J_z(\nabla_z(...))|_{z=w_2} & & \\ & & \ddots & \\ & & & J_z(\nabla_z(...))|_{z=w_n} \end{bmatrix}$$

Applying the Hessian is thus of lower complexity, $\mathcal{O}(n^3)$, than for a potentially dense Hessian (which is $\mathcal{O}(n^4)$). This makes iterative Krylov subspace methods

suitable, for instance when solving the Newton equation. Such methods need not form the matrix either, but can apply (3.12) directly. However, this is the Euclidean Hessian. The projection operator onto the tangent space of the orthogonal group must also be taken into account.

The projection operator is as mentioned

$$\mathrm{P}_W A = W\mathrm{skew}\,(W^T A) = \frac{1}{2}(A - WA^T W), \quad \mathrm{P}_{\vec{W}}\vec{A} = \frac{1}{2}(\vec{A} - (W^T \otimes W)\sigma(\vec{A})),$$

where $\sigma$ is again the perturbation operator such that $\mathrm{vec}(Z^T) = \sigma(\mathrm{vec}(Z))$. Moreover, the directional derivative and its projection are

$$(\mathrm{D}_Z\mathrm{P})A = -\frac{1}{2}(ZA^T W + WA^T Z), \quad \mathrm{P}_W(\mathrm{D}_Z\mathrm{P})A = -\frac{1}{2}\left(\mathrm{sym}\,(A^T W) + \mathrm{sym}\,(WA^T)\right)$$

By vectorizing all the tools used above, it is possible using equation (2.2) to compute an explicit Hessian matrix. However, in a practical application it is more convenient to use it as a linear operator, and use a Krylov subspace solver.

### 3.2.3 The FastICA algorithm

The arguably most common method to solve the ICA problem, is the FastICA algorithm proposed by Hyvarinen and Oja[18] in 1997. The method finds the components one-by-one, followed by a re-orthogonalization procedure to ensure that the constraint is fulfilled. The derivation of the method is based on constrained optimization theory and the Karush-Kuhn-Tucker conditions. Let $\phi = 0.5\mathrm{E}[(g(w^T\tilde{x}) - C)^2]$ be the one-component contrast. The derivation is not included here, since it is readily available in the literature. The algorithm is to repeat:

1. $y_k \leftarrow \mathrm{E}[\tilde{x}g'(w_k^T\tilde{x})]$
2. $w_{k+1} \leftarrow w_k - \mu[y_k - (w_k^T y_k)w_k]/[\mathrm{E}(g''(w_k^T\tilde{x}) - w_k^T y_k]$
3. $w_{k+1} \leftarrow w_{k+1}/\|w_{k+1}\|$.

$\mu$ is a learning parameter, where $\mu = 1$ correspond to a full Newton step at every iteration. It is introduced to be able to stabilize the solution in case the second-order approximation is poor, and thus the Newton step inaccurate. When searching for more than one of the signals, a re-orthogonalization have to be performed as well. This is in principle close to performing a projection and a retraction on the Stiefel manifold, but the implementation can be done in a variety of ways.

# Chapter 4

# Experimental Design

This chapter describes how the numerical experiments in this thesis were performed. Two hypothetical test problems were studied to hopefully provide insight into how to specialize any of the methods to other problems. The parameters involved are also discussed, some of which have been set according to what has been done previously in the literature, and some of which had to be studied in depth. Measures of performance are also discussed, but first the algorithms are restated for completeness and for an easier comparison.

## 4.1 Implementation

### Sparse Representation

Three implementations were done for the sparse representation problem. A steepest descent method, Newton's method and the FOCUSS-based method described in Kreutz-Delgado et al.[20] for comparision. Initially, the $\mathcal{A}_F$ structure was assumed. Using $\mathbb{R}^{m\times n}$ or $\mathcal{A}_C$ instead would only require minor alterations, as discussed in section 3.1.

The FOCUSS-based method was presented in section 3.1.3. Choosing $l_1$ as diversity measure yields the following two steps to alternate:

1. $s_{k+1}(t) = \Pi_k^{-1} A_k^T(\lambda I + A_k \Pi_k^{-1} A_k^T)^{-1} x(t)$, for $t = 1 \dots T$
2. $A_{k+1} = A_k - \mu(\nabla_A \phi(A_k, S_k) - \mathrm{tr}(\nabla_A \phi(A_k, S_k)^T A_k) A_k)$

$\Pi_k^{-1} = \Pi_k^{-1}(s_k(t)) = \mathrm{diag}(|s_k(t)|)$, but the dependence on $s_k$ is dropped for readibility. For clarity it is noted that $s_k(t)$ refers to iteration number $k$ and time step $t$(that it column number $t$ in $S$). The notation comes from the factorization of the gradient from section 3.1.2. Step 1 can be done batch-wise. That is, update the dictionary for every $T_N$ updates of $s$, instead of running all the time samples. The dictionary update follows from the manifold structure. For all the experiments, $T_N$

was set to 100 (corresponding to 10% of the total number of time samples).

The steepest descent method was implemented using the projected gradient and a simple retraction, as described in section 2.1

$$\begin{aligned}
&\nabla_A \phi(A_k, S_k) = \frac{1}{T}(A_k S_k S_k^T - X S_k^T) \\
&A_{k+1} = A_k - h(\nabla_A \phi(A_k, S_k) - \operatorname{tr}(\nabla_A \phi(A_k, S_k)^T A_k) A_k) \\
&S_{k+1} = S_k - h\left(\frac{1}{T}(A_k^T A_k S_k - A_k^T X) + \lambda \operatorname{sgn}(S_k)\right) \\
&A_{k+1} \leftarrow A_{k+1} / \|A_{k+1}\|_F
\end{aligned} \tag{4.1}$$

Note that $h$ need not be constant, but can be chosen adaptively.

Newton's method was implemented when assuming the manifold $\mathcal{A}_F$ only. The two steps of the Newton method are given in algorithm 1, but repeated here for completeness. First, solve

$$\text{Hess } \phi(A_k, S_k)[Z_A, Z_S] = -\text{grad } \phi(A_k, S_k),$$

for $Z_A \in T_A \mathcal{A}_F$ and $Z_S \in \mathbb{R}^{n \times T}$. The solution $(Z_A, Z_S)$ is referred to as the Newton vector. Next retract the solution onto the manifold, that is

$$A_{k+1} = \frac{A_k + \mu_N Z_A}{\|A_k + \mu_N Z_A\|_F}, \quad S_{k+1} = S_k + \mu_N Z_S,$$

where $\mu_N$ is a potential step size reduction to ensure stability. The same step size reduction is used in $S$ and $A$. Using a different reduction may make more sense, but this was not investigated here. The Hessian is given in equation (3.8), and it was implemented both as an operator (a function call in Matlab) and by forming the matrix explicitly. When solving the system iteratively, the built-in function minres was used. It requires a self-adjoint operator (which the Hessian by definition is), but does not require positive-definiteness such as the conjugate gradient method. It was chosen because it gave low relative residual errors, and the solution satisfied for the most part being on the tangent space. To ensure that the Newton vector was on the tangent space, a projection was performed on $Z_A$ before the retraction step.

## Independent Component Analysis

For the ICA problem, three methods were implemented, steepest descent, Newton's method and FastICA. The contrast used was $\phi(s) = (\mathrm{E}[g(s) - C])^2$, with $g(s) = 1/4 s^4$, such that $C = 3/4$ (since $C$ is defined as the expected value of $g$ for a standard Gaussian variable). The one-component FastICA algorithm with learning rate thus becomes

1. $y_k \leftarrow \mathrm{E}[\tilde{x}(w_k^T \tilde{x})^3]$

2. $w_{k+1} \leftarrow w_k - \mu_F[y_k - (w_k^T y_k)w_k]/[\mathrm{E}(3(w_k^T \tilde{x})^2 - w_k^T y_k]$

3. $w_{k+1} \leftarrow w_{k+1}/\|w_{k+1}\|$.

The decorrelation was done after all the iterations had completed and performed in a similar fashion as the pre-whitening procedure. By using the eigenvalue decomposition $VDV^T = \mathrm{E}[\tilde{W}\tilde{W}^T]$, where $\tilde{W}$ is the matrix of column vectors from the above iterations, the solution is determined by

$$W = VD^{-1/2}V^T\tilde{W}.$$

This decorrelation scheme was suggested in the article by Hyvarinen and Oja[19].

The steepest descent implementation, referred to as GradICA (gradient-based ICA), was implemented with the same contrast function as FastICA. With the notation introduced in section 3.2, the algorithm is to repeat

1. Compute Euclidean gradient: $\nabla_W \phi(W_{k+1}) = \tilde{X}G'(W_k)D(W_k)$

2. Project gradient onto tangent space: $\nabla_W \phi(W_{k+1}) \leftarrow \mathrm{P}_{W_k} \nabla_W \phi(W_{k+1})$, with $\mathrm{P}_W : \mathbb{R}^{n\times n} \to T_W O(n)$ defined as $\mathrm{P}_W(B) = W\mathrm{skew}\,(W^T B)$.

3. Apply retraction for next iterate: $W_{k+1} = R_{W_k}(h\nabla_W \phi(W_{k+1}))$, with $R_W : T_W O(n) \to O(n)$ defined as $R_W(Z) = \exp(ZW^T)W$.

The step size parameter $h$ may be constant or chosen adaptively.

Newton's method was implemented by explicitly forming the block diagonal Hessian, defined in section 3.2 by equation (3.12). This was to ensure consistency between theory and implementation. A practical large-scale implementation could use an iterative solver with the Hessian implemented as an operator instead. The Newton equation was then solved directly by Matlab. The next step is found by applying a retraction (the same as for steepest descent) to the Newton vector. For stability, a step size reduction parameter, $\mu_H$, was introduced, such that the entire Newton step need not be taken at each iteration. In the following, Newton's method, with or without Hessian modification, will be referred to as HessICA.

## Test Problem 1, Complete Dictionary

The first problem set up was replicated from Kreutz-Delgado et al.[20]. For $n = m$, both sparse representation and ICA are valid methods, hence an attempt at a broad comparison was made. $n$ was set to 20, and 1000 time samples were used. $A$ was generated with standard Gaussian distributed values, and scaled to have Frobenius norm 1. $S$ was generated by first uniformly constructing a random sparse support with 4 non-zeros at each time step. The values were generated as Gaussian variables. Any entry with absolute value lower than 0.1 were then removed to have a more distinct sparsity pattern.

### Test Problem 2, Overcomplete Dictionary

The second test problem is specifically oriented towards dictionary learning and sparse coding. The problem was generated in a similar fashion as above, except that the dimensions were different. The dimensions were $n = 40, m = 20, r = 7$. $T$ was still 1000. The dictionary was again scaled to have Frobenius norm 1.

## 4.2 Parameters and Practical Considerations

### Steepest Descent

Initial empirical tests indicated that the choice of algorithmic parameters is not straightforward. It was therefore important to determine if there are any stability issues related to the choice of parameters. In particular, the gradient method required rigorous testing. A minimum requirement to a method should be that if the initial guess is sufficiently close to the true solution, then convergence is ensured. In particular, if $\lambda = 0$, that is no penalty on sparsity, then the error should get close to zero. The first test was therefore to see if the original solution was perturbed slightly (by Gaussian noise with variance 0.05), did the method converge when $\lambda = 0$? The tests were performed on problem 2 first.

The penalty term was considered next. Kreutz-Delgado et al.[20] suggested using a dynamical value, with maximum value $2 \cdot 10^{-3}$. For the method to work on arbitrary data, it would be ideal to be able to choose $\lambda$ according to a desired sparsity level. It was therefore interesting to study the effect of $\lambda$, and if it could be set to a constant.

Since the initial data is already sparse, one should expect that the sparsity is constant for moderate $\lambda$. However, if $\lambda \to \infty$, then the error is not weighted at all, and the optimal solution is $S \to 0$. It is therefore necessary to determine an upper bound for a reasonable $\lambda$. The term $\lambda d(S)$ will perturb the stationary points of $\|AS - X\|_F$, so the problem will be to find a $\lambda$ such that the original $A$ and $S$ still are a solution (or at least very close to a solution) of the optimization formulation. If this is not the case, the initial guess is not close to a stationary point, thus convergence may not be expected.

The idea behind the tests was to find a $\lambda$ that caused divergence. This value would then be assumed as the maximum bound when the original values are not available as initial guess.

### ICA Methods

One of the advantages of ICA is that when the contrast function is chosen, there are few parameters left to tune. GradICA requires a step size. For the tests runned in this thesis, a constant step size is assumed to suffice. Since GradICA

is only applied to the first test problem, the step size parameter was tuned to ensure convergence on that specific example. $h = 0.01$ turned out to be a good choice.

FastICA has as mentioned a step size parameter, or learning rate, even though it attempts a Newton step at each iteration. The step size was set to 0.5, which ensured convergence to a stationary point (but not necessarily a maximum), for the first test problem.

The Hessian operator was also implemented for the ICA problem. Just as for the other methods, it was of interest to see if a step size parameter had to be introduced in order to ensure convergence. It was also studied if global convergence could be assured by introducing a Hessian modification to ensure positive definiteness. Only a multiple of the identity matrix was considered as Hessian modification.

### Initial Data

After having evaluated performance close to the original solution, it was necessary to test for global convergence. In this case, that means choosing initial data independent of the original $A$ and $S$, and tune $\lambda$ and $h$ to achieve optimal performance. Three ideas were tested out:

1. Choose $A$ and $S$ randomly, perhaps with a scaling of $S$
2. Choose $A$ randomly and $S = A^+X$, $A^+$ being the Moore-Penrose pseudoinverse of $A$
3. Let $A = I_n$ and $S = X$ (only valid for the test problem 1, when $n = m$)

If a manifold structure was assumed on $A$, this was also taken into account. For instance for 2), $A$ would be enforced onto the manifold before the pseudoinverse was computed. For the ICA methods, the initial data was always chosen as random orthogonal matrices.

## 4.3 Measures of Performance

To measure performance, one may address the objective function. It is a decent measure when comparing methods that start from the same initial point, and are based on the exact same objective. It does not however work well to compare problems to which both ICA and sparse representation apply. In addition to using the objective function, measures of performance based on the problem statement can be used. Two such measures are presented. The first one is sparsity. The sparsity measure that has been implemented is sometimes referred to as the $l_0^\varepsilon$ measure. Before taking the number of non-zero elements divided by the number of elements, all elements with absolute value lower than $\varepsilon(\max_{ij} |S_{ij}|)$ are set to zero. In these experiments, $\varepsilon = 0.05$ was used.

The second measure was root mean square error(RMSE), which is basically the square root of the first term of the objective function from equation (3.2)

$$\text{RMSE}(S^\varepsilon) = \frac{1}{\sqrt{T}}\|AS^\varepsilon - X\|_F,$$

where $\varepsilon$ emphasizes that the solution is "sparsified" before computing the RMSE, thus separating it slightly from the objective function. The objective function was also used, not as a performance measure, but as a way of verifying that the objective was decreasing. Run time was not investigated since it depends to a great extent on how the implementation is done. Complexity is however important, it will be assumed that an effecient implementation reflects the complexity of the method.

## Convergence Rate

Convergence rate were measured in some of the cases. In general, if a method converges to $x^*$, its convergence rate is given by the maximum $r$ such that,

$$\frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^r} \leq C, \tag{4.2}$$

where $C$ is some bounded constant. Convergence rate can be measured for both $A$ and $S$, and it is assumed that when considering $A$, it is sufficient to consider the norm on $\mathbb{R}^{m\times n}$ instead of having to computing the distance on the manifold.

$x^*$ is set to the value of either $A$ or $S$ when the algorithm terminated. That is, every test case has to be run twice to first identify the stationary point, then to compute the convergence rate towards it. When doing the analysis it should be kept in mind that not all final solutions were stationary points. This is discussed after the presentation of the results.

In a similar manner, one may also consider the convergence rate of the objective function. That is,

$$\frac{f(x^{k+1}) - f(x^*)}{(f(x^k) - f(x^*))^r} \leq M.$$

# Chapter 5

# Results and Discussion

## 5.1 Local Convergence

Before attempting to achieve global convergence for the sparse representation problems, local convergence was studied. That is, choose the initial dictionary and source signal as the ones used to generate $X$, but add a noise term such that they are close to, but not exactly the original solution. The ICA problems proved to be less dependent on initial data and converged for most randomly generated initial matrices. The penalty term, $\lambda$, and the step size parameter for steepest descent were studied first.

### Evaluating $\lambda$ and $h$

The first tests were done to find suitable $\lambda$ for both FOCUSS and the steepest descent method. A step size parameter, $h$, was also needed for steepest descent. It is a necessary criterion that if the initial data is close to the original data, convergence should be assured. As mentioned in the previous chapter an upper bound on $\lambda$ must be sought for as well.

| $\lambda$ | RMSE | $\Delta$Sparsity | $\Delta A$ | $\Delta S$ |
|---|---|---|---|---|
| 0 | 3.033E-2 | 0% | 0.8198 | 8.65E-12 |
| 1E-3 | 3.028E-2 | 0% | 0.8198 | 8.225E-8 |
| 1E-1 | 4.053E-3 | -0.37% | 0.8194 | 6.2852E-4 |

Table 5.1: Performance for close initial data. $h = 0.01$ and 500 iterations. $\Delta A$ is the difference between the final $A$ from the initial one, measured in the same way as in figure 5.1. $\Delta S$ is measured similarly. The original sparsity was 16.073%.

The perturbation was with Gaussian noise of variance 0.05. In hindsight, it may have been unfortunate to use the same variance when perturbing $A$ and $S$. Since the Frobenius norm of $A$ is 1, the elements of $A$ are in general small, such

that 0.05 variance makes a significant difference. This is the reason why $\Delta A$ is so large in table 5.1. However, it appeared that the optimization of $A$ was very efficient such that the performance was not affected to a great extent by this.

By increasing $\lambda$, the change in $S$ increases. This is consistent with what was expected. Increasing $\lambda$ should mean increasing the importance of sparsity. Sparsity is exclusively determined by $S$, hence it is reasonable that increasing the penalty also increased the difference in $S$ from the initial state. $h = 0.05$ was also tested, but the solution did not appear to converge, which led to setting $h = 0.01$. In the tests it was also seen that decreasing $h$, but keeping the product of $h$ and the number of iterations constant produced the same solution. If this holds in the general case, it means that decreasing $h$ will not increase accuracy, but it must be low enough so that the method is stable. For now, this will be the assumed conjecture on step size.

From table 5.1 it is clear that $\lambda = 0$ and $\lambda = 1E-3$ are both good candidates. However $\lambda = 0.1$ encourages even sparser solutions than the original solution and may thus be too large ($\Delta$Sparsity is the change in sparsity from the initial state). By iterating further, the sparsity decreases and the RMSE increases. The effect is the same as for $\lambda = 1$ which is discussed next, where it is more accentuated.

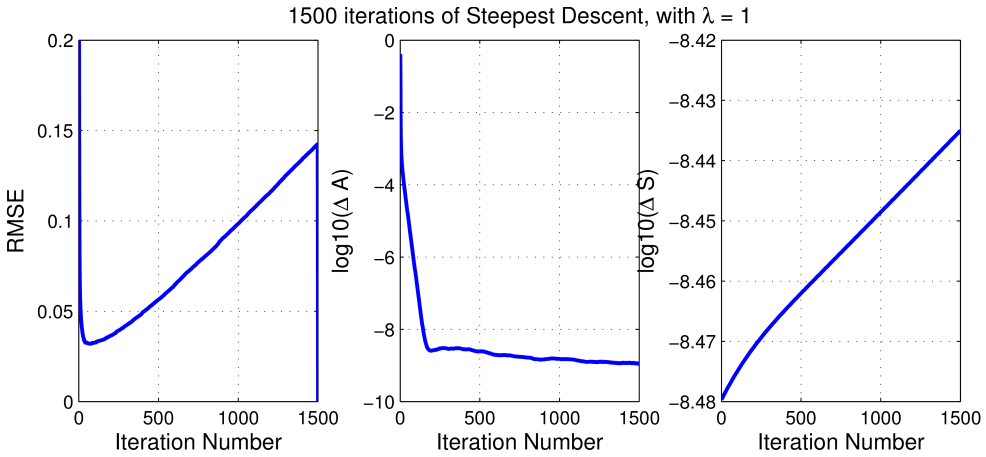


Figure 5.1: Closer study of steepest descent performance. For iteration number $k$, $\Delta A(k)$ is defined as $1 - \frac{\mathrm{tr}(A_k^T A_{k+1})}{\|A_k\|_F \|A_{k+1}\|_F}$, and similarily for $S$, such that a low $\Delta A$ implies a small change in $A$ from iteration $k$ to $k+1$, and vice versa.

Figure 5.1 gives an idea of what happens when $\lambda$ is too large. Initially the RMSE decreases relatively fast, but after around 100 iterations it increases again. At its lowest, it is approximately the same as for the lower values of $\lambda$. The other two plots show the change in $A$ and $S$. The values on the $y$-axes are not comparable to eachother, but it is clear that the rapid decrease in RMSE is due to the

adaptability of $A$ (as mentioned above). For $\lambda \geq 1$, the behavior is the same except that it is magnified. The same test was run with $h = 10^{-3}$ instead of $h^{-2}$, but the results were practically the same.

In practice, it should be possible to stop the iterations close to the minimum RMSE, at least for this value of $\lambda$. When the initial data is not sparse as in this case, it will be important that $\lambda$ is large enough such that the solution to the optimization formulation is actually sparse, but small enough such that the gradient with respect to $S$ is not entirely dominated by the diversity term.

Empirical tests were done to evaluate the behavior for $\lambda$ larger than $10^{-1}$. For the following results, $10^{-1}$ is set as the practical upper bound for $\lambda$. As mentioned, the initial decrease of RMSE is attributed to the adaptability of the dictionary, and not the signals themselves. Recall the gradient with respect to $S$,

$$\nabla_S \phi(A, S) = \frac{1}{T}(A^T AS - A^T X) + \lambda \operatorname{sgn}(S).$$

The elements of the second term, the gradient from the diversity measure, can only take three values, $\pm\lambda$ and 0. The latter being rare since it requires an element of $S$ to be exactly 0. This means that an exact stationary point is hard to find since every discrete value of $\lambda \nabla_S d(S)$ must be negated by the smooth gradient of the error term. Note that the gradient from the error term is the residual of the least squares equation ($A^T Ax = A^T b$) divided by $T$. Intuitively, this can be understood as for large $\lambda$, it is necessary to have a large residual error in the least squares approximation at a stationary point. Also, when $\lambda$ becomes large, the updating of the dictionary is not able to keep up with the rate at which the values of $S$ decrease. This will not be analyzed further, but the discrete nature of the diversity gradient is noted as a potential cause for not finding true stationary solutions.

To summarize the initial tests on the gradient method, it is important that $\lambda$ is chosen such that the solution becomes sparse enough (that is, $\lambda$ is not too small), but also so that when approaching a stationary point, the gradient with respect to $S$ is affected by both the error term and the diversity term. The convergence of $A$ is rather efficient in comparison, but it cannot achieve arbitrarily low RMSE for any $S$. In conclusion, there appears to be a fine balance between $\lambda$ and $h$ that needs to be investigated further when switching to other initial conditions. Due to the lack of flexibility of the diversity gradient, it will also be important to be able to stop the iterations on some other criteria than $\nabla_S \phi = 0$. Another remedy may be to consider more flexible parameters. Two suggestions are a more dynamic choice of $\lambda$ and a different step size for $A$ and $S$ (that is, the two gradient flows defined by the steepest descent method depend on different time parameters).

## Newton's Method

As mentioned Newton's method was also implemented for the sparse representation problem. However, consistent convergence was hard to achieve. Using Gaussian

noise as perturbation of the initial data, with variance $5 \cdot 10^{-7}$ for $A$ and $5 \cdot 10^{-5}$ for $S$, did not cause consistently decreasing RMSE, even with $\lambda = 0$. Step size reduction was attempted as well, with limited success. It was observed that there may have been some issues with solving the Newton equation. Forming the matrix explicitly and using direct solve in Matlab caused the solution to not be obtained on the manifold. This is generally not a problem since a post-projection can be performed afterwards, but since the forming of the matrix was very slow, this method was not investigated in great depth. Performing 15 iterations took typically 8-10 seconds with minres, but several minutes when forming the entire matrix. Solving the system using minres had its issues as well. Using 2000 iterations, which is a significantly higher number of iterations than was hoped for, the relative residual was typically of order $10^{-3}$, which is relatively inaccurate

The Rayleigh quotient example studied in section 2.3 required getting relatively close to the stationary solution to achieve convergence. Considering that this linear system is large, $20800 \times 20800$ (somewhat sparse, but nonetheless) compared to $2000 \times 2000$, it is not unreasonable that the initial state must be very close in order for a non-modified Newton's method to work.

## Convergence rate

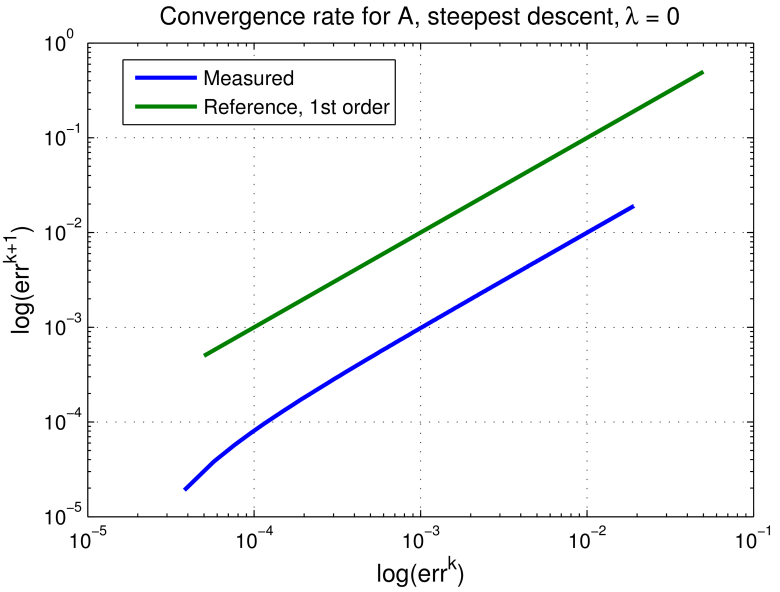


Figure 5.2: $h = 0.01$, 500 iterations. Linear convergence observed. Signs of superlinear convergence towards the final solution.

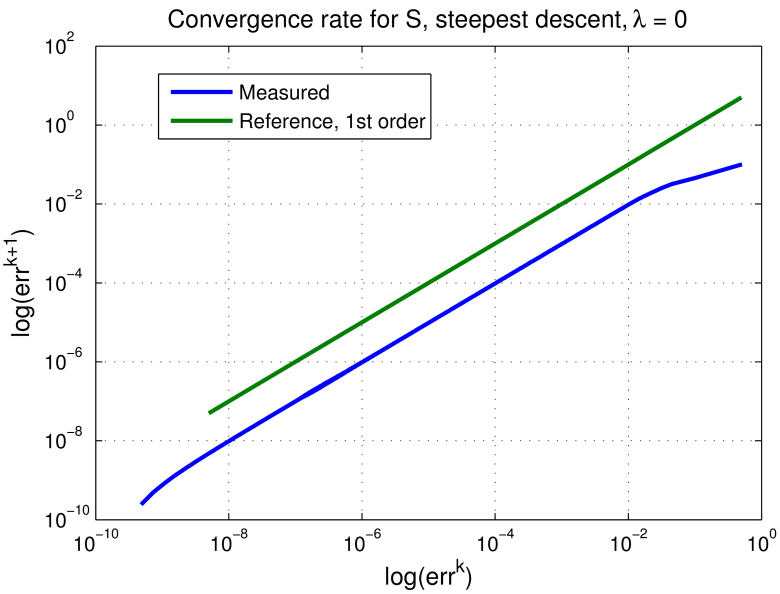


Figure 5.3: $h = 0.01$, 500 iterations. Linear convergence for $S$.

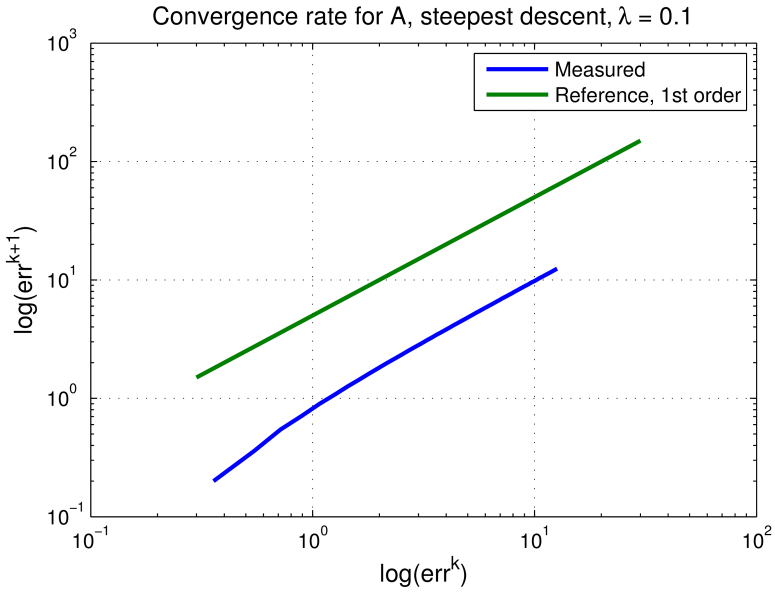


Figure 5.4: $h = 0.01$, 500 iterations. Linear convergence for $A$.

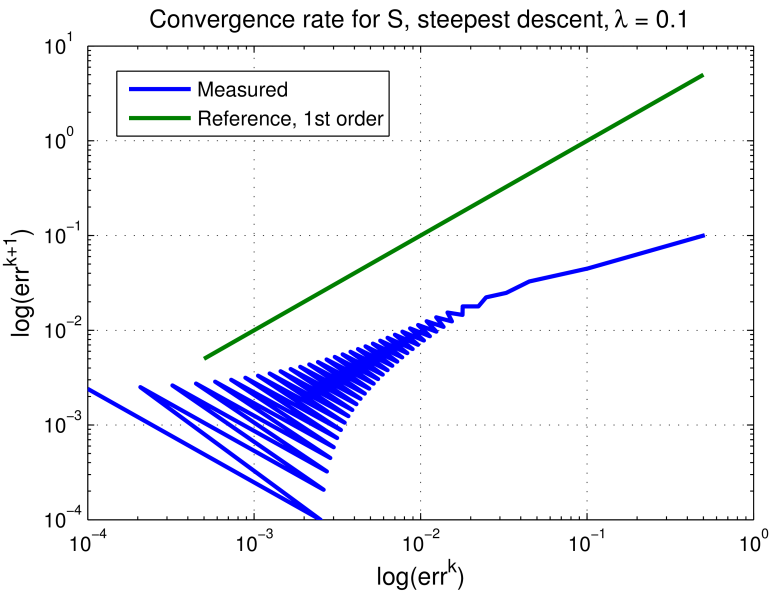


Figure 5.5: $h = 0.01$, 500 iterations. Problematic behavior towards to the final iteration, stationary point unlikely.

Another measure of performance mentioned was convergence rate. Figure 5.2 and 5.3 show the measured convergence rate for $h = 0.01$ and $\lambda = 0$. The behavior is as expected linear. Figure 5.4 and 5.5 on the other hand, show convergence rate when $\lambda = 0.1$. The iterations were stopped when minimum RMSE was reached since a definite stationary point was not reached. Observe that $A$ exhibit linear convergence, but $S$ does not. Initially (the north east part of figure 5.5) the convergence rate is consistent, but lower than one. When getting closer to the final solution, the convergence rate goes out of control. This could be related to the fact that the solution is compared to iteration number 70 (which corresponds to lowest RMSE), and considering how much $S$ seemingly oscillates, it is not unreasonable to observe sublinear convergence at the beginning. This will not be spectulated on any further, but should be kept in mind. It is noted however, that at least in the beginning, the behavior is at least nice.

The tests were run from the same initial data as for the results in table 5.1. This indicates that $\lambda = 0.1$ is capable of ensuring low RMSE, but also that it is not likely to converge to a stationary point (seen from figure 5.5).

## 5.2 Global Convergence

Both test problems were checked for global convergence. For the first test, a comparison was done with the FOCUSS-based method, steepest descent on the ICA

formulation and FastICA. Parameter tuning turned out to be necessary in order to obtain convergence. Substantial testing was done with regards to the parameters. $\lambda$ was finally set to $5 \cdot 10^{-2}$ and $h = 0.01$, satisfying what was discussed above. The choice of initial data appeared to be of great importance as well. The magnitude of the initial $S$ influenced whether the method would converge or not. Necessary criteria for global convergence were not sought. Letting $A = I$, the identity matrix, and $S = X$ provided satisfactory convergence, at least compared to the other two suggestions from the previous chapter. This was chosen as the initial data for the FOCUSS-based method as well. The initial data for FastICA and GradICA were both chosen as qr-decompositions of random matrices.

It is noted that for this specific problem, it was necessary to tune down the learning rate of the FOCUSS-based method to achieve convergence. In the article by Kreutz-Delgado et al.[20] the dictionary update step size was 1, but neither this nor 0.1 worked here. The learning rate was therefore fixed at 0.01. Considering that the dictionary update of the FOCUSS-based method is basically the same as for the gradient based method, it is reasonable that a lower learning rate is required, considering the analysis above. Also, in the mentioned article, $\lambda$ was adaptive which might have mitigated for a large learning rate.

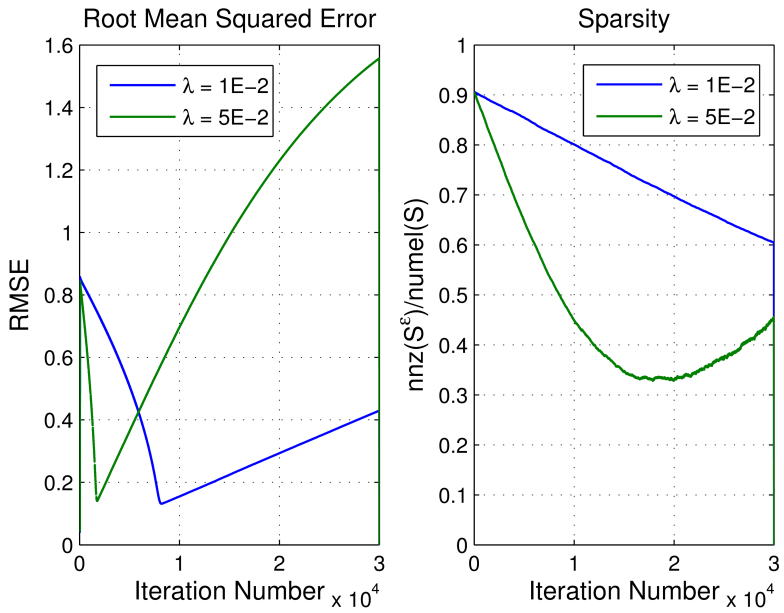


Figure 5.6: Global convergence of RMSE and sparsity for two values of $\lambda$. $h = 0.01$ in both cases. 30000 iterations performed, which is substantial. The form of $\lambda$ =1E-2 resembles a "stretched" version of $\lambda$ =5E-2, which is consistent with the assumption that a lower $h$ does not increase accuracy.

To illustrate the dependence on $\lambda$ and $h$, sparsity and RMSE have been plotted in figure 5.6 for the gradient based method. First of all, they illustrate that a significant number of iterations is necessary for stable convergence. Increasing $h$ to 0.1 turned out to be unstable. By visual inspection, it seems that for RMSE higher than 0.4, the error is too large. So regardless of sparsity criterium, a subjective opinion is that the RMSE should not exceed 0.4 for test problem 1. Note that RMSE is not a relative measure, thus every application needs a separate evaluation of performance. However, it is potent at comparing various methods to one another.

## Full Scale Comparison

| Method | Sparsity | RMSE | Iterations |
|---|---|---|---|
| FOCUSS | 17.8% | 1.26E-1 | 500[1] |
| Steep. Desc. | 62.5% | 3.97E-1 | 5500 |
| GradICA | 31.4% | - | 145 |
| FastICA | 43.7% | - | $6 \cdot 20$[2] |

Table 5.2: Comparison of performance, global convergence. Initial sparsity was approximately 90%. Steep.Desc. is steepest descent on the sparse representation problem, implemented using $\mathcal{A}_F$.

The summary of the comparison of the four methods is shown in table 5.2. Since the ICA methods by construction should satisfy the equation $AS = X$ (after pre-whitening), the RMSE is not computed. The sparsity does however indicate whether the solution is the true solution. For comparison, the original signals had sparsity 18.3%. The number of iterations to convergence is determined subjectively by inspecting the values of RMSE and sparsity. It should give an indicator for when a satisfactory solution is reached. In particular, for steepest descent the iterations were stopped for the lowest sparsity that had an RMSE lower than 0.4.

It is clear that FOCUSS gave the definite best approximation, with a low sparsity, low RMSE and relatively few iterations. Steepest descent required a significant number of iterations, and the sparsity of the solution is moderate at best. It is however promising that the method is able to provide a solution with around 30% lower sparsity than $A = I$ and $S = X$, and still having a (at least visually) acceptable solution.

The main difference between the steepest descent method and FOCUSS, is that FOCUSS uses a fixed point iteration on the signals that to a greater extent encourages sparse signals. Table 5.2 thus emphasizes what has been suggested earlier;

[1] After 100 iterations, the sparsity had already reached 24.8% but lowering RMSE required more. This is likely due to the reduction of the learning rate.

[2] Estimated 6 Newton steps on average per signal. Based in visual inspection of the contrast function for each signal.

that the dictionary gradient search is rather efficient compared to the signals when it comes to decreasing RMSE, at least for a constant $\lambda$.

Another intersting observation is that GradICA performs better than FastICA. FastICA is derived for a component-by-component computation, so it may not be that surprising a method that considers all the signals simultaneously shows better performance. FastICA also seeks stationary solutions, and not necessarily minima. From a closer inspection of the individual signals, it becomes clear that FastICA has indeed converged to a maximum for a few of the signals. On the other hand, the FastICA implementation was done rather straightforward, and several papers in the literature addresses the issues of FastICA converging to maxima. Still, steepest descent is conceptually also a simple method, and it is noteworthy that the performance is relatively good.

## HessICA

As mentioned, Newton's method was implemented for the ICA problem. First of all it was of interest to see if quadratic convergence was obtainable close to a stationary point. A steepest descent (or technically ascent in this case) was performed first, with step size $h = 0.01$. The Newton's method was carried on from there. Quadratic convergence was indeed observed, however the convergence was very rapid and highly dependent on the initial data. From the point initizalied by steepest descent, it took 2-3 steps to reach the stationary point, up to machine precision. As a particular example: from the 128-th step of steepest descent, it took 3 iterations to converge, but if the starting point was the 127-th step, the stationary was not obtained. It could be of interest to study larger dimensions, or manipulate the problem to be harder so that more iterations are required.

Why it is necessary to get so close to the maximum is not investigated further, but as seen in section 2.4, the answer may lie in the eigenvalues of the Hessian at the stationary point. As mentioned a Newton's method with Hessian modification was tested, where the modification was adding a multiple of the identity. This is indeed the simplest choice because a Hessian modification must not only be able to make the Hessian negative definite (since a maximum is sought in this case), but must also satisfy being a linear map from the tangent space to itself. The largest eigenvalue of the Hessian matrix was computed and if it was positive, it was multiplied by $I$ and subtracted from the Hessian matrix. To ensure a negative definite matrix, and not just semi-negative definite, 1 was added to the largest eigenvalue before the subtraction was done. The choice of 1 was merely from taking into consideration the percieved magnitude of the eigenvalues for different iteration steps, and ensuring that the condition number was kept at an acceptable level.

Interestingly, the method with Hessian modification appeared more stable than steepest descent. By using the same initial point, the Hessian modified Newton's method looked to converge faster than steepest descent. However, the convergence

| Method, manifold | RMSE | Sparsity | Iterations |
|---|---|---|---|
| FOCUSS, $R^{m\times n}$ | 0.2276 | 42.4% | 250 |
| FOCUSS, $\mathcal{A}_F$ | 0.2935 | 30.44% | 45 |
| FOCUSS, $\mathcal{A}_C$ | 0.3026 | 23.68% | 20 |
| Steepest Descent, $\mathbb{R}^{m\times n}$ | - | - | - |
| Steepest Descent, $\mathcal{A}_F$ | 1.0427 | 50.93% | 22809 |
| Steepest Descent, $\mathcal{A}_C$ | 1.0810 | 47.48% | 23517 |

Table 5.3: Effect of imposing a manifold structure on $A$. A dash (-) indicates the method did not converge. Iterations for FOCUSS is iteration until RMSE converged, sparsity converged at around 500 iterations for all three manifolds. The results are from test problem 2.

was only linear, which is likely to be caused by the closeness-to-maximum requirement discussed above. More rigorous testing is required in this regard, but it is clear that the concept works, and that quadratic convergence can be achieved close enough to a maximum.

## Implications of Manifold Structure

To test the implications of imposing a manifold structure on $A$, test problem 1 was considered. The FOCUSS-based method and steepest descent were implemented by imposing the three different manifold structures mentioned in section 3.1. The original dictionary was normalized such that it was on the manifold $\mathcal{A}_F$. It was assumed that this does not benefit any method over the other. The initial data were chosen by randomly generating $A_{\text{initial}}$ and enforce it onto the appropriate manifold. $S$ was then set to $S_{\text{initial}} = A^+_{\text{initial}}X$, where $A^+_{\text{initial}}$ is the Moore-Penrose pseudoinverse of $A_{\text{initial}}$. The same initial dictionary was used for all methods (before it was enforced onto the manifold). In hindsight, the initial matrix should probably have been enforced onto $\mathcal{A}_C$ such that all methods could have used the exact same matrix as the initial dictionary. Recall that $A \in \mathcal{A}_C \implies A \in \mathcal{A}_F$ ( $\implies A \in \mathbb{R}^{m\times n}$).

Finding appropriate parameters when evaluating the manifold structure was based on the analysis done in section 5.1. Some adaptation was needed, and the final choice was based on visual inspection of the solutions. For steepest descent, the step size was set to 5E-3. $\lambda$ was set to 0.05 for both problems, and the learning rate of FOCUSS was set to 0.01. It turned out to be hard for steepest descent to find a solution that was both accurate and sparse. With these parameters, the iterations were stopped at the lowest RMSE that also satisfied having a diversity around or below 50%.

From table 5.3 it is clear that FOCUSS in general outperforms steepest descent. The solution for steepest descent is not good, but it resembles the true solution. This is illustrated in figure 5.7, which shows a sample from the matrix $AS$ compared to the original $X$. Using $\mathbb{R}^{m\times n}$ did not work for steepest descent, but this

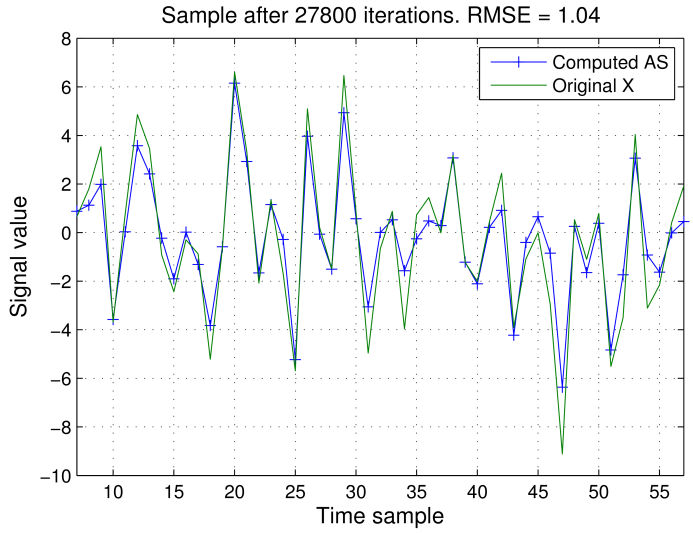


Figure 5.7: Sample of the solution after 22809 iterations on $\mathcal{A}_F$, $h = 0.005$ and $\lambda = 0.05$. The solution is similar, but in general not good enough for all applications, considering this was the best solution achieved.

may be due to how the initial matrix was chosen. Without normalizing the initial matrix to a manifold, the magnitude of the values of $S$ became large (caused by small singular values of $A$, which in turn causes large singular values in $A^+$). This issue was not investigated further, but it is noted that it could be of interest to study some form of pre-processing. Note that the initial signals may be multiplied from the right by a matrix without interfering with the manifold structure of $A$: $(XM) = A(SM)$, such that the expected order of magnitude of the elements of $S$ can to some extent be controlled beforehand.

When it comes to the impact of the manifold structure, a better diversity appears to be achieved from using $\mathcal{A}_C$. This is consistent with what was reported in the article by Kreutz-Delgado[20] et al. The RMSE was slightly higher, but since both the diversity and the number of iterations (at least for FOCUSS) was lower, it is assumed to be the generally better choice.

## 5.3 Concluding Discussion and Further Work

Finally, the potential of taking a manifold approach to an optimization problem is evaluated. The steepest descent method was studied and showed promising results for convergence locally, where the step size had to be kept low enough to ensure stability. A practical implementation may choose the step size in an adaptive

manner. Global convergence was harder to achieve, and the search for a suitable dictionary appeared more efficient than updating the signals. This may explain why the FOCUSS-based algorithm performs so much better; the fixed point iteration for the signals is much more convenient for finding sparse signals.

The sparsity penalty term, $\lambda$, was also studied. It was argued that a too large $\lambda$ led to problems for the steepest descent method, and especially how hard it can be to reach an exact stationary solution due to discrete nature of $\nabla_S d(S) = \text{sgn}\,(S)$. An idea may be to enforce some constraints on $S$ or to consider smoother diversity measures, the latter probably being the most interesting for any further work. Alternatively, choosing $\lambda$ adaptively have been extensively studied in the literature, and should be considered also here. FOCUSS can be studied to better understand why the fixed point iteration on the signals works so well.

For the dictionary learning problem, no clear advantage was seen by optimizing over both $A$ and $S$. Considering the discussed adaptability of $A$ compared to $S$, it may not be much to gain on doing the update in one step. Also, a reduction of step size was implemented in most of the cases, indicating that something like a trust-region method may be necessary in a practical implementation, with an initial radius that is relatively small.

The results from the manifold-based methods on the ICA problems was promising, and it could be interesting to see how they perform on a more realistic case. It may also be of interest to test the performance on other measures of convergence, since there are some clear disadvantages to using kurtosis, especially sensitivity to outliers. For HessICA in particular it may be interesting to consider other Hessian modifications, such as BFGS or SR1, which may be adapted to the manifold implementation[1].

Newton's method for sparse representation was implemented, but trouble was met when trying to achieve even local convergence. It may be important to better understand the steepest descent method and the objective function, especially the Hessian of the diversity measure, before attempting to achieve local convergence. This provides another argument for considering smoother diversity measures when implementing with manifolds. In general it is interesting to study the methods that rely on second-order information, due to the reduced complexity of applying the Hessian.

# Chapter 6

# Conclusion

Chapter 2 discussed how to interpret an optimization problem through manifolds, and concepts from Riemannian geometry were introduced to provide a geometrical understanding of methods on manifolds. The multi-dimensional Rayleigh quotient was studied since its stationary points are well known. Chapter 3 described two blind source separation techniques, and how the optimization can be performed using matrix manifolds. The FOCUSS algorithm and the FastICA methods were presented for comparison.

The relatively low complexity that was seen in chapter 3, especially for Newton's method, is one the properties that makes the methods attractable for blind source separation problems. It was seen how the steepest descent method depends on the choice of initial data as well as other parameters. The method showed potential by achieving local convergence, but global convergence still require some work, especially with respect to achieving sparser solutions. The manifold-based methods implemented for independent component analysis worked very well and outperformed FastICA, at least up to the performance measures used here. Overall, most of the methods required a step size reduction in order to converge, indicating that a practical implementation should be done using some sort of trust-region method.

Further work would be a more rigorous study of the steepest descent method to better understand the geometry of the sparse representation problem, and hopefully be able to implement a working Newton's method. For independent component analysis it is of interest to see if the manifold-based methods presented here can perform as well on a more advanced problem, and perhaps further study the potential of Newton's method with Hessian modification.

# Bibliography


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

[2] P. Absil and J. Malick, *Projection-like retractions on matrix manifolds*, SIAM Journal on Optimization, 22 (2012), pp. 135–158.

[3] P.-A. Absil, R. Mahony, and J. Trumpf, *An extrinsic look at the Riemannian Hessian*, Tech. Rep. UCL-INMA-2013.01-v1, U.C.Louvain, February 2013.

[4] 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, 22 (2002), pp. 359–390.

[5] M. Aharon, M. Elad, and A. Bruckstein, *K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation*, Signal Processing, IEEE Transactions on, 54 (2006), pp. 4311–4322.

[6] S.-M. Cha and L.-W. Chan, *Applying independent component analysis to factor model in finance*, in Intelligent Data Engineering and Automated LearningâĂŤIDEAL 2000. Data Mining, Financial Engineering, and Intelligent Agents, Springer, 2000, pp. 538–544.

[7] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*, Academic press, 2010.

[8] J. Demmel, *Applied Numerical Linear Algebra*, Miscellaneous Bks, Society for Industrial and Applied Mathematics, 1997.

[9] M. do Carmo, *Riemannian Geometry*, Mathematics (Birkhäuser) theory, Birkhäuser Boston, 1992.

[10] D. L. Donoho, *For most large underdetermined systems of equations, the minimal l1-norm near-solution approximates the sparsest near-solution*, tech. rep., Comm. Pure Appl. Math, 2004.

[11] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. MATRIX ANAL. APPL, 20 (1998), pp. 303–353.

[12] K. Engan, S. Aase, and J. Hakon Husoy, *Method of optimal directions for frame design*, in Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, vol. 5, 1999, pp. 2443–2446 vol.5.

[13] I. F. Gorodnitsky, J. S. George, and B. D. Rao, *Neuromagnetic source imaging with focuss: a recursive weighted minimum norm algorithm*, Electroencephalography and clinical Neurophysiology, 95 (1995), pp. 231–251.

[14] S. Grant, D. Skillicorn, and J. R. Cordy, *Topic detection using independent component analysis*, in Proceedings of the 2008 Workshop on Link Analysis, Counterterrorism and Security (LACTS'08), 2008, pp. 23–28.

[15] Z. He, S. Xie, and A. Cichocki, *On the convergence of focuss algorithm for sparse representation*, CoRR, abs/1202.5470 (2012).

[16] N. P. Hurley and S. T. Rickard, *Comparing measures of sparsity*, CoRR, abs/0811.4706 (2008).

[17] A. Hyvarinen, *Fast and robust fixed-point algorithms for independent component analysis*, Neural Networks, IEEE Transactions on, 10 (1999), pp. 626–634.

[18] A. Hyvärinen and E. Oja, *A fast fixed-point algorithm for independent component analysis*, Neural computation, 9 (1997), pp. 1483–1492.

[19] ——, *Independent component analysis: algorithms and applications*, Neural networks, 13 (2000), pp. 411–430.

[20] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, *Dictionary learning algorithms for sparse representation*, Neural Comput., 15 (2003), pp. 349–396.

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

[22] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM review, 45 (2003), pp. 3–49.

[23] J. Nocedal and S. Wright, *Numerical Optimization*, Springer series in operations research and financial engineering, Springer Science+Business Media, LLC., 2006.

[24] M. Shub, *Some remarks on dynamical systems and numerical analysis*, Proc. VII ELAM.(L. Lara-Carrero and J. Lewowicz, eds.), Equinoccio, U. Simón Bolıvar, Caracas, (1986), pp. 69–92.

[25] Z. Wen and W. Yin, *A feasible method for optimization with orthogonality constraints*, Mathematical Programming, (2013), pp. 1–38.